Universiteit Antwerpen

Automatic exploration techniques for the numerical continuation of large–scale nonlinear systems

Proefschrift voorgelegd op 19/09/2019 tot het behalen van de graad van doctor in de wetenschappen – wiskunde, bij de Faculteit Wetenschappen aan de Universiteit Antwerpen.

Promotor:
Prof. dr Wim Vanroose

Michiel Wouters

Applied Mathematics
University of Antwerp

Doctoral committee

Chairman:	
Prof. dr Benny Van Houdt	University of Antwerp
Supervisor:	
Prof. dr Wim Vanroose	University of Antwerp
Other members:	
Prof. dr Jacques Tempere	University of Antwerp
Prof. dr Sonja Hohloch	University of Antwerp
Prof. dr Dirk Roose	KU Leuven
dr Jonas Thies	German Aerospace Center (DLR)

Original title: *Automatic exploration techniques for the numerical continuation of large–scale nonlinear systems*

Nederlandse titel: *Automatische exploratie technieken voor de numerieke continuatie van grootschalige niet-lineaire systemen*

Keywords: *Numerical continuation, Automatic exploration, Ginzburg-Landau equation, Newton-Krylov method, singularity, sparse linear algebra*

Published and distributed by Michiel Wouters. The research presented in this thesis was supported by the Department of Mathematics and Computer Science of the University of Antwerp.

Contact information

✉ Applied Mathematics, Dept. Mathematics & Computer Science
University of Antwerp (CMI)
Middelheimlaan 1, 2020 Antwerp, Belgium

`mich_wouters@hotmail.com`

`https://be.linkedin.com/in/michiel-wouters-6a3207185`

"Fly, you fools."

– Gandalf –
The Fellowship of the Ring

Samenvatting

Dynamische systemen vormen een belangrijke klasse van wiskundige modellen, ze laten ons toe om veel fenomenen uit de fysica, chemie, biologie en zelfs economie te bestuderen. Typisch houdt de studie van een dynamisch systeem in dat de evolutie van bepaalde toestanden in de tijd worden gesimuleerd. Vaak convergeren deze toestanden naar zogenaamde stabiele equilibria. Equilibria zijn toestanden van het systeem die constant blijven in de tijd, ze zijn stabiel als ze ook aangehouden blijven onder kleine perturbaties. Niet enkel zijn de toestanden met deze eigenschap zelf interessant om te bestuderen, ook hun afhankelijkheid ten opzichte van fysische parameters, zoals bijvoorbeeld de temperatuur of beginconcentraties van bepaalde stoffen, vormt een belangrijk onderzoeksdomein. Een voorbeeld van een dynamisch systeem dat een aantal keer zal terugkomen in de thesis, is het Ginzburg-Landau model. Dit model is een niet-lineaire Schrödinger vergelijking die bepaalde types van supergeleiding beschrijft. Toestanden van supergeleiding hangen typisch af van de temperatuur en de sterkte van het toegepaste magnetische veld, we zullen in de thesis focussen op deze laatste parameter.

In de praktijk wordt de afhankelijkheid tussen toestand en fysische parameter onderzocht met een wiskundige techniek genaamd numerieke continuatie. Hierbij worden verschillende continue, geconnecteerde, curves van equilibria benaderd met behulp van andere numerieke methoden. De verzameling van zo'n curves wordt ook wel een (geconnecteerd) oplossingslandschap (of bifurcatiediagram) van het dynamisch systeem genoemd. Door equilibria op de curves met elkaar te vergelijken, kan men informatie afleiden omtrent de afhankelijkheid tussen de mogelijke toestanden van het systeem en de fysische parameters. De thesis focust op deze numerieke continuatie techniek, meer bepaald het onderdeel dat bestaat uit automatische exploratie. Hiermee wordt het automatisch genereren van een volledig oplossingslandschap bedoeld, in plaats van slechts één curve. Dit wordt gerealiseerd in twee stappen: bifurcatiepunten (equilibria die een transitie in eigenschappen van het systeem induceren) worden benaderd, en raaklijnen naar nieuwe curves uit deze punten worden geconstrueerd.

De huidige technieken om numerieke continuatie uit te voeren schieten vaak te kort. Zo is een groot deel van de beschikbare software gebaseerd op dichte lineaire algebra, wat hun gebruik op grootschalige systemen niet toelaat. Verder zijn er problemen bij het benaderen van bifurcatiepunten, ten gevolge van een eigenschap die deze equilibria hebben: de Jacobiaan van het systeem,

geëvalueerd in een bifurcatiepunt, kan singulier zijn. Dit brengt problemen met zich mee wanneer bijvoorbeeld de methode van Newton wordt toegepast in een algoritme dat dient om het punt te benaderen. Ook bij het construeren van raaklijnen naar nieuwe curves zijn er problemen. Hoewel ze efficiënt bepaald kunnen worden voor systemen zonder symmetrie, is dit niet het geval als het systeem wel bepaalde symmetrieën bevat.

De thesis tracht een oplossing voor de beschreven problemen te vinden. **Hiervoor zullen we de numerieke algoritmes afleiden die nodig zijn voor de automatische exploratie van (geconnecteerde) oplossingslandschappen, waarbij we ons volledig baseren op ijle lineaire algebra.** De voornaamste toepassing van deze technieken is het Ginzburg-Landau model, dat in detail zal worden beschreven.

Om numerieke continuatie bruikbaar te maken voor grootschalige systemen, zullen we gebruik maken van algoritmes gebaseerd op ijle lineaire algebra, zoals Ritz paren en Krylov methoden. Deze technieken spelen ook een rol binnen een klasse van aangepaste Newton algoritmes die beschreven zullen worden in de thesis. De methode van Newton is een krachtig algoritme dat gebruikt wordt om oplossingen van niet-lineaire vergelijkingen te benaderen, maar slaagt hier mogelijk niet in als de Jacobiaan van de vergelijking slecht geconditioneerd is. We zullen een alternatieve versie van het algortime afleiden, waarbij een opsplitsingsstrategie wordt toegepast op de update vectoren en extra delen hieraan worden toegevoegd. Met behulp van enkele voorbeelden tonen we aan dat de alternatieve methoden ook werken voor niet-lineaire vergelijkingen met een slecht geconditioneerde Jacobiaan. Vervolgens tonen we aan hoe deze nieuwe Newton methoden aangepast kunnen worden voor gebruik binnen numerieke continuatie, met als toepassing het bepalen van bifurcatiepunten.

In het tweede deel van de thesis bespreken we details omtrent numerieke continuatie zelf, waarbij we extra aandacht vestigen op het probleem omtrent het bepalen van raaklijnen naar nieuwe oplossingscurves. We leiden hierbij twee strategieën af, die gebruikt kunnen worden voor het geval waarin het dynamisch systeem bepaalde symmetrieën bevat.

De numerieke technieken die besproken worden in de thesis geven uiteindelijk de aanleiding tot een uitbreiding van het Python pakket PyNCT, een toolbox gebaseerd op Python die gebruikt wordt voor het genereren van geconnecteerde oplossingslandschappen. Het pakket is volledig gebaseerd op ijle lineaire algebra, en slaagt erin de twee stappen die vereist zijn voor automatische exploratie uit te voeren. De thesis wordt afgesloten met enkele voorbeelden van zo'n oplossingslandschappen, waarbij we extra aandacht vestigen op deze van het Ginzburg-Landau model.

Bijdragen:

- Analyse omtrent convergentie van huidige Newton methoden bij toepassing op een niet-lineaire vergelijking met slecht geconditioneerde Jacobiaan.
- Afleiden van aangepaste Newton methoden die wel werken op dit type vergelijkingen.

- Uitbreiding van deze aangepaste Newton methoden met blok eliminatie technieken. Dit is nodig om ze toe te kunnen passen binnen algoritmes voor het benaderen van bifurcatiepunten.
- Afleiden van twee algoritmes die het mogelijk maken om raaklijnen te bepalen, gebruikt in het geval dat het onderliggend dynamisch systeem symmetrieën bevat.
- Uitwerking van details rondom automatische exploratie in de praktijk.
- Beschrijving van pseudo-code voor de onderliggende numerieke methoden.
- Uitbreiding van het Python pakket PyNCT met methoden die het automatisch exploreren van een geconnecteerd oplossingslandschap mogelijk maken.
- Bespreking van enkele oplossingslandschappen, onder andere voor het Ginzburg-Landau model.

Summary

Dynamical systems form an important class of mathematical models, they allow us to study many phenomena in physics, chemistry, biology and even economics. Typically, the study of a dynamical system involves simulating the evolution of certain states over time. Often these states converge to so-called stable equilibria. Equilibria are states of the system that remain constant in time, they are stable if they persist under small perturbations as well. Not only are the states with this property themselves interesting to study, their dependence towards physical parameters, like the temperature or initial concentrations of certain substances, forms an important research domain as well. An example of a dynamical system that appears several times in the thesis is the Ginzburg-Landau model. This model is a nonlinear Schrödinger equation that describes certain types of superconductivity. States of superconductivity typically depend on the temperature and the strength of the applied magnetic field, we will focus on this last parameter in the thesis.

In practice, the dependency between state and physical parameter is investigated with a mathematical technique called numerical continuation. Hereto different continuous, connected, curves of equilibria are approximated using other numerical methods. The collection of such curves is also referred to as a (connected) solution landscape of the dynamical system. By comparing equilibria on the curves, information can be derived about the dependence between the possible states of the system and the physical parameters. The thesis focuses on this numerical continuation technique, more specifically the part that consists of automatic exploration. The automatic generation of a complete solution landscape, instead of just a single curve, is meant by this. It is achieved in two steps: bifurcation points (equilibria that induce a transition in properties of the system) are approximated, and tangent directions to new curves that emanate from these points are constructed.

Current techniques for performing numerical continuation often fall short. For example, a large part of the available software is based on dense linear algebra, which does not allow their use for large-scale systems. Furthermore, there are problems in the approximation of bifurcation points, due to a property of these equilibria: the Jacobian of the system, evaluated in a bifurcation point, is possibly singular. This causes problems when, for example, the Newton method is applied in an algorithm used to approximate the point. There are also problems with the construction of tangent directions to new curves.

Although they can be determined efficiently for systems without symmetry, this is not the case if certain symmetries are contained in the system.

The thesis attempts to find a solution for the described problems. **For this purpose we will develop the numerical algorithms required to automatically explore (connected) solution landscapes, entirely based on sparse linear algebra.** Our main application for these techniques is the Ginzburg-Landau model, which will be discussed in detail.

To make numerical continuation operable for large-scale systems, we will use algorithms based on sparse linear algebra, such as Ritz pairs and Krylov methods. These techniques also play a role within a class of modified Newton algorithms that will be described in the thesis. The Newton method is a powerful algorithm that is used to approximate solutions of nonlinear equations, but might fail if the Jacobian of the equation is ill-conditioned. We will derive an alternative version of the algorithm, whereby a splitting strategy is applied to the update vectors and extra parts are added. We illustrate that the alternative methods also work for nonlinear equations with an ill-conditioned Jacobian by providing some examples. Next, we demonstrate how these new Newton methods can be adapted for use within numerical continuation, with the approximation of bifurcation points as an application.

In the second part of the thesis we discuss details about numerical continuation itself, with extra attention being paid to the problem of determining tangent directions to new solution curves. We derive two strategies that can be used in the case where certain symmetries are contained in the dynamical system.

The numerical techniques discussed in the thesis eventually give rise to an expansion of the Python package PyNCT, a Python-based toolbox that is used to generate connected solution landscapes. The package is entirely based on sparse linear algebra, and is able to execute the two steps required for automatic exploration. The thesis is concluded with some examples of solution landscapes, where we draw extra attention to those of the Ginzburg-Landau model.

Contributions:

- Analysis of convergence of current Newton methods when applied to a nonlinear equation with ill-conditioned Jacobian.
- Derivation of adjusted Newton methods that do work on this type of equations.
- Expansion of these adjusted Newton methods with block elimination techniques. This is necessary for their application within algorithms used to approximate bifurcation points.
- Derivation of two algorithms that make it possible to determine tangent directions, used in the case that symmetries are contained in the underlying dynamical system.
- Elaboration of details regarding automatic exploration in practice.

- Description of pseudo-code for the underlying numerical methods.
- Expansion of the Python package PyNCT with methods that allow for automatic exploration of a connected solution landscape.
- Discussion of some solution landscapes, for example for the Ginzburg-Landau model.

Dankwoord

"It's just a flesh wound."

– The Black Knight –
Monty Python and the Holy Grail

Het einde van het schrijfwerk is bijna daar. Na het beschrijven van supergeleiders, oplossingslandschappen en meer wiskunde dan voor een mens gezond is, is het enige dat me nog tegenhoudt om alles in te dienen dit dankwoord. Misschien is dit zelfs het belangrijkste onderdeel van de thesis - of waarschijnlijk toch het deel dat het meeste gelezen zal worden. En terecht, want alleen was het me zeker niet gelukt om mijn doctoraat tot een goed einde te brengen. Dit dankwoord is aan iedereen gericht die, al dan niet bewust, de voorbije 4 jaar een bijdrage heeft geleverd.

Het spreekt voor zich dat een doctoraat zonder promotor niet zou goedkomen. Dit brengt me bij de eerste persoon aan wie dit dankwoord gericht is: Wim Vanroose. De voorbije 4 jaar hebben Wim en ik wekelijks samen gezeten om mijn onderzoek te bespreken en te brainstormen waar nodig. Zonder zijn kennis en inzicht zouden veel van de technieken uit deze thesis nooit ontwikkeld zijn. Hoewel hij als departementsvoorzitter een drukke periode achter de rug heeft, stond hij toch altijd voor mij en mijn collega's klaar. Ik zou me geen betere promotor kunnen indenken.

De leden van de doctoraatsjury wil ik ook bedanken. Naast hun eigen onderzoek, hebben ze ook de tijd genomen om dit proefschrift te lezen en waar nodig feedback te geven. Ik denk dat een verontschuldiging hier ook op zijn plaats is: sorry voor de lengte van deze thesis. Jurylid prof. dr Jacques Tempère zou ik even expliciet willen vermelden in dit dankwoord. Mijn onderzoek was onderdeel van een samenwerking met het fysica-departement. De meetings die daaruit volgden waren zeer leerrijk, en zorgden voor een beter begrip van de toepassingen. Ook jurylid dr Jonas Thies, en daarbij ook zijn collega's in het DLR, zou ik hier extra willen bedanken. Tijdens onze samenwerking werd ik altijd zeer gastvrij ontvangen, en zijn interesse in het onderzoek heeft zeker ook een motiverend effect gehad.

In je eentje onderzoek doen is natuurlijk maar eenzaam, gelukkig kon ik de voorbije jaren rekenen op de steun van mijn collega's. Met Jacob en Nick heb ik beide 3 jaar op kantoor gezeten. Bedankt voor de leuke gesprekken die we daar regelmatig hadden, me te helpen als ik weer in de problemen zat met

dingen zoals Sisa, en mee te brainstormen over wiskunde. Dit brengt mij ook tot een tweede verontschuldiging: sorry voor mijn geklaag en gezucht van de voorbije jaren, het onderzoek ging jammer genoeg niet elke dag even vlot. Ook mijn andere collega's en oud-collega's uit de toegepaste wiskunde wil ik hier even vermelden: Lynn, Jeffrey, Siegfried, Delphine, Shoaib, Koen, Przemyslaw, Bram, Vincent en Valdemar, bedankt voor de vele steun die ik aan jullie had. Ook een dikke merci aan Sten, een collega uit de fundamentele wiskunde. Als het onderzoek wat minder ging en de nood aan pauze verhoogde, kon ik altijd bij jullie terecht voor een babbel.

Het voorbije academiejaar was niet enkel speciaal omdat het mijn laatste jaar op de universiteit was. Ik heb het dan natuurlijk over 3 november, de dag dat ik met Silke trouwde. De voorbije 10 maanden en zo'n 300 pagina's aan thesis werd me vaak genoeg duidelijk waarom ik haar het ja-woord heb gegeven. Silke stond altijd voor me klaar wanneer het schrijven moeizaam ging, of er een onverwacht wiskundig probleem opdook (en zoals elke wiskundige weet, is hier jammer genoeg geen ontkomen aan). Met een knuffel, een relativerend mopje of zelfs met dinokoeken wist ze me steeds terug op te vrolijken. Ik zou nog meer redenen kunnen geven waarom ik haar niet zou kunnen missen, maar omdat ik weet dat ze niet van geslijm houdt, zal ik het hierbij laten.

Buiten Silke kon ik ook altijd rekenen op de steun van mijn familie. Zonder mijn ouders had ik hier vandaag zelfs niet gestaan, bedankt om tijdens mijn schooltijd in me te geloven, en me de kans te geven om verder te studeren in wiskunde. Ook aan mijn zus, Evelien, heb ik de afgelopen jaren veel gehad. Ik denk dan niet alleen aan mijn tijd als doctoraatstudent, maar ook aan mijn beginperiode op de universiteit. Bedankt om me regelmatig op je kot uit te nodigen om te eten of om samen te gaan fitnessen. Verder mogen ook mijn grootouders, stieffamilie, Rick, Jantje, nonkel Jean en tante May niet in dit dankwoord ontbreken. Bedankt voor jullie steun!

Het is misschien iets dat niet in elk dankwoord gebeurt, maar ook mijn schoonfamilie zou ik hartelijk willen bedanken. Tegen alle clichés in ben ik hen de afgelopen jaren als een tweede familie beginnen zien, en weet ik dat Silke en ik altijd op ze kunnen rekenen. Maar ik denk dat ik nu wel genoeg heb geslijmd, hopelijk zijn er weer wat pluspunten verdiend.

Ook mijn vrienden verdienen een dikke "dank je wel". Het doet me veel dat ze - ondanks de vaak ingewikkelde materie - toch regelmatig interesse toonden in mijn werk, en voor de nodige relativering zorgden op momenten dat het onderzoek minder vlot liep.

Bij relativering denk ik vooral aan de keren dat mijn doctoraat het gespreksonderwerp was binnen de party crew # 1. Alvast bedankt voor de appelsienen, en nee, mijn formule is na 4 jaar nog altijd niet opgelost.

Buiten interesse en relativering, heb ik ook enorm veel gehad aan de nodige ontspanning de afgelopen jaren. Ik heb het dan niet alleen over de voorbije 4 jaar, maar ook de periode daarvoor. Mijn studententijd beschouw ik nog steeds als één van de mooiste periodes van mijn leven. Dit heb ik voornamelijk te danken aan mijn oud-klasgenoten, en andere vrienden die ik tijdens mijn studententijd heb leren kennen. Muurklimmen en boardgames zijn ondertus-

sen vaste hobby's geworden, maar ook gewoon een pintje drinken deed soms wonderen wanneer de wiskunde weer een keer niet wou meewerken. Ook Hans zou ik expliciet willen vermelden. Na meer dan 10 jaar beschouw ik hem nog steeds als één van mijn beste maten.

Hopelijk vonden jullie het niet te erg om vandaag naar een wiskunde presentatie te luisteren, anders laten jullie dit volgende keer op café maar weten en kan ik het hopelijk goed maken met een cola of een pintje.

Tot slot zou ik in dit dankwoord ook mijn leraar wiskunde uit het 3de en 4de middelbaar willen vermelden. Hoewel ik wiskunde altijd een leuk vak vond, is mijn passie ervoor pas echt begonnen tijdens de lessen van Benny Luman. Uitspraken als "Alles is wiskunde" (wat me de inspiratie voor het begin van de inleiding gaf) of "Er zit meer wiskunde in een beukennootje dan in alles van Microsoft samen" herinner ik me nog als gisteren. Na 9 jaar wiskunde te studeren, kan ik ook alleen maar zeggen dat hij gelijk heeft.

Met deze uitspraken zou ik dan ook mijn dankwoord willen afsluiten. Wim, leden van de jury, Silke, collega's, vrienden en familie, allemaal ontzettend bedankt voor jullie steun de afgelopen 4 jaar. Zonder jullie zou het niet gelukt zijn.

Michiel Wouters
Antwerpen, 19 september 2019

Contents

List of notations

Abbreviations

PDE	Partial differential equation
MINRES	Minimal residual method
GMRES	Generalized minimal residual method
NCG	Nonlinear conjugate gradients
IFT	Implicit function theorem
EBL	Equivariant branching lemma
HPC	High performance computing
PyNCT	Python Numerical Continuation Toolbox
AUTO	Software for continuation and bifurcation problems in ordinary differential equations [31]
MATCONT	Continuation software in Matlab [30]
LOCA	Library of Continuation Algorithms [90]

General

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{R}_0$	Real numbers different from zero
$\mathbb{R}_0^+$	Real numbers strictly greater than zero
$\mathbb{C}$	Complex numbers
i	Imaginary unit
$\mathfrak{Re}(c)$, $\mathfrak{Re}(x)$	Real part of a scalar c or vector x
$\mathfrak{Im}(c)$, $\mathfrak{Im}(x)$	Imaginary part of a scalar c or vector x
$\mathcal{C}(V)$	Set of continuous functions from V to V
exp	Exponential function
sgn	Sign function
∇	Gradient operator
Δ	Laplace operator
$\dim(V)$	Dimension of a set V
I	Identity operator or matrix
e_i	ith unit vector
δ_{ij}	Kronecker delta, equals 1 if $i = j$, 0 otherwise.

x^T, A^T, B^T	Transpose of a vector x, matrix A or linear operator B
x^*, A^*, B^*	Conjugate transpose of a vector x, matrix A or linear operator B
$\bar{x}$	Conjugate of a vector x
A^{-1}, B^{-1}	Inverse of a matrix A or linear operator B
$\mathcal{N}(A)$, $\mathcal{N}(B)$	Nullspace of a matrix A or linear operator B
$\mathcal{R}(A)$, $\mathcal{R}(B)$	Range of a matrix A or linear operator B
$\langle x, y \rangle$	Inner product of vectors x and y
$\langle x, y \rangle_2$	Euclidean inner product x^*y
$\|x\|$	Norm of vector x
$\|x\|_2$	Euclidean norm $\sqrt{x^*x}$
D_m	Dihedral group of degree m
C_m	Cyclic group of degree m
S^1	Circle group

Numerical methods

n	Size of the vectors
ε_{mach}	Machine precision
$\mathcal{K}_m(\mathcal{A}, b)$	The m-dimensional Krylov subspace spanned by $\mathcal{A}$ and b
$\mathcal{P}$, P	Preconditioner
K	Deflation matrix
$\mathcal{Q}$	Operator for projecting orthogonal to K
Δt	Time step
ΔX, Δx, Δp	Newton update vectors
λ_j	(Approximate) eigenvalue of a matrix or linear operator

Numerical continuation

$\mathcal{F}$	Function from $V \times \mathbb{R}$ to W (with $V, W \subseteq \mathbb{C}^n$) that describes an equilibrium equation
ψ	Variable ($\in V$) of $\mathcal{F}$ that describes the state of the system
μ	Variable ($\in \mathbb{R}$) of $\mathcal{F}$ that describes the physical parameter of the system
Ψ	The variable (ψ, μ)
n	Size of the state variable ψ
$\mathcal{F}_\psi$, $\mathcal{F}_{\psi\psi}$, ...	First partial derivative of $\mathcal{F}$ to ψ, Second partial derivative of $\mathcal{F}$ to ψ, ...
$\mathcal{D}^j_\psi \mathcal{F}$	jth partial derivative of $\mathcal{F}$ to ψ
$(\psi(s), \mu(s))$	Solution curve of $\mathcal{F}$
$(\psi^{(b)}, \mu^{(b)})$	Bifurcation point of $\mathcal{F}$
$(\dot{\psi}, \dot{\mu})$	Tangent direction to the solution curve in (ψ, μ)
$S_\rho(\psi, \mu)$	The set of points (ψ', μ') that lie within a distance strictly lower than ρ from (ψ, μ)

G	Symmetry group associated with $\mathcal{F}$
Δs	Step length used in numerical continuation
C	Condition function
C_{start}	Start point condition

Superconductors

Ω	Open, bounded region of the Euclidean space
d	Side length of the considered material
$\alpha, \beta, \lambda, \xi$	Material parameters
X	Natural energy space of Ω
Y	Dual space of X
T	Temperature
T_{c_1}, T_{c_2}	Critical temperatures
$\mathbf{H}_0$	External magnetic field
$\mathbf{B}$	Total magnetic field
μ	Strength of the external magnetic field
μ_{c_1}, μ_{c_2}	Critical magnetic field strengths
ρ_C	Density of Cooper pairs
Ψ	Order parameter with $\rho_C = \lvert\Psi\rvert^2$
$\mathcal{E}(\psi)$	Part of the Gibbs energy that depends on the (discretized) order parameter ψ
$\mathcal{K}(\mu)$	Kinetic operator

List of pseudo-code

CHAPTER 1

Introduction

"Before we get started, does anyone want to get out?"
*– **Steve Rogers** –*
Captain America: The Winter Soldier

1.1 Dynamical systems

Though direct applications of mathematics are often invisible in daily routines, life without them has become almost unimaginable. Using mathematical models, a lot of physical, chemical, biological and even economical phenomena can be studied without actually performing experiments. The behaviour of weather, the growth of plant tissue, concentrations of chemical substances, or the price of an economic asset all seem to follow certain rules, which can often be described by mathematical equations. These four examples belong to the type of model called the dynamical system: the time dependence of a certain state is expressed by a mathematical evolution function. Such functions are often defined through a partial differential equation (PDE), as is the case for the dynamical systems considered in this thesis. These partial differential equations have the general form

$$\frac{\partial \Psi(x,t)}{\partial t} = F(\Psi(x,t), \Lambda), \tag{1.1}$$

where $\Psi : \Omega \times \mathbb{R} \to \mathbb{C}$ describes the physical/chemical/... state for each point x (in a certain space Ω) and time $t \in \mathbb{R}$. The function F that appears in the equation is typically nonlinear, possibly containing other partial derivatives like $\frac{\partial \Psi(x,t)}{\partial x}$ or $\frac{\partial^2 \Psi(x,t)}{\partial x^2}$. Often the equation is dependent on certain physical parameters, like the temperature, chemical substance properties, initial concentrations, ... These are denoted by $\Lambda \in \mathbb{R}^k$ ($k \in \mathbb{N}$) in (1.1).

By analysing the corresponding partial differential equation, the evolution of a physical/chemical/... state in time is studied, allowing scientists to make predictions of future behaviour. By studying meteorology models (e.g. the ECMWF model [70]), data of current and past conditions is used to make weather forecasts, up to multiple days. The auxin transport models discussed in [35, 34] allow for the prediction of hormone patterns in a plant tissue, used in

the simulation of the tissue's growth. Certain autocatalytic, oscillating chemical reactions are described by the Brusselator system [2, 106], used to analyse the evolution in concentrations of the different compounds. Prices of multiple types of economical options are determined by studying the appropriate financial model, for example the Black-Scholes equation [58] for European options.

Another example of a dynamical system, which will be the main application discussed in the thesis, are superconductors: materials that exhibit a complete loss of electrical resistivity when certain conditions on the temperature, the magnetic field and other physical properties are met. The states of a superconductor are described mathematically by the Ginzburg-Landau model [37], a type of nonlinear Schrödinger equation. Superconductors have many applications. They are for example of interest when building nanoscale fluxonics devices to use in e.g. SQUIDS [65], RSFQ processors [41] and supercomputers [33, 54, 76].

Before predictions of a system's behaviour can be made, the corresponding partial differential equation needs to be analysed. This usually means that the solution $\Psi(x, t)$ of (1.1) is required. Solving such an equation is however far from trivial. Though for some models it is possible to derive an analytical expression for the solution, this is not true in general. Even if such an expression exists, it might still not be practical to use for large-scale problems due to a high computational and memory cost.

Instead of deriving analytical expressions to solve partial differential equations, the thesis will focus on numerical methods. These do not yield an exact solution, but create approximations up to a certain tolerance. From a given initial guess, better approximates are constructed in an iterative manner. Due to their general low computational cost and memory requirements, they form an interesting alternative. In many applications one would rather compute a good approximation to a solution in a short period of time, instead of an exact one that takes long to compute. Examples of numerical methods, that are used for the analysis of a dynamical system, are the Crank-Nicolson time step scheme and the Newton method. Both methods will be discussed throughout the thesis.

1.2 Equilibria

The evolution in time of a certain state is an interesting topic in many branches of science, but for many applications the *eventual* state, as time progresses, of the dynamical system is especially important. If the state of the system becomes constant over time, we call this eventual state an equilibrium (or steady state). Note that for so-called chaotic systems the behaviour of states over time is entirely unpredictable, and an equilibrium might even never be reached. We will not consider such chaotic systems in the thesis.

We distinguish two different kinds of equilibria: physically stable and unstable ones. An equilibrium is called unstable if it does not persist under small perturbations, these states are usually not physically realizable. Dynamical systems typically settle down at stable equilibria, which do persist under small perturbations.

One class of methods to determine equilibria consists of time stepping the partial differential equation. Given an initial state, these methods simulate its evolution over a set of discrete points in time. By generating many evaluations and comparing the states at the discrete time steps, it is possible to determine whether a state is near an equilibrium. Though time step methods generate both the evolution in time and the eventual state, they are not able to find physically unstable equilibria.

A second class of methods consists of setting the partial derivative $\frac{\partial \Psi(x,t)}{\partial t}$ in (1.1) equal to zero, and solving the nonlinear equation that arises for $\Psi(x,t)$:

$$F(\Psi(x,t),\Lambda) = 0. \tag{1.2}$$

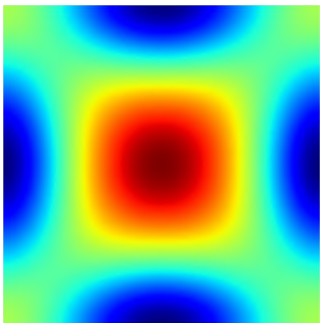
By setting the partial derivative of the state towards the time to zero, we demand the solution of this equation to be constant in time, implying its equilibrium property. One way to solve equations of the form (1.2) is by the Newton method, used to approximate a zero of a function near a given initial guess. Methods of this second class allow for the determination of both physically stable and unstable equilibria.

Though the direct use of unstable equilibria is limited in real situations, they still contain important information on transitions between different equilibrium patterns of a dynamical system. This is motivated by example 1.1 below, where the state of a superconductor is analysed.

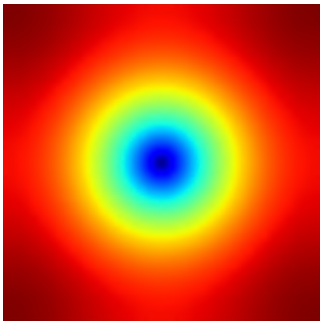
Example 1.1. We consider a square-shaped, 2-dimensional extreme type-II superconductor of side length 3 (measured in units of coherence length) subject to a homogeneous magnetic field (see section 9.5.1 for more details). The strength of the field is fixed at a value that allows the material to adopt two possible stable vortex patterns (this is the value $\mu = 1.3$ in the (dimensionless) Ginzburg-Landau equation, see chapter 2). These two states are shown in figure 1.1: both vortex patterns 1 and 2 are physically stable [92]. The significant parts of their energies (the parts that depend on the order parameter, see chapters 2 and 9) are respectively given by $\mathcal{E}_1 = -0.23387$ and $\mathcal{E}_2 = -0.13271$. We keep the value for the magnetic field strength fixed and want to perform a transition from pattern 1 to pattern 2 through a temporary external perturbation, by supplying additional energy.

It is, however, not sufficient to add the difference in energy between patterns 1 and 2 to the system. Instead, the minimal amount of energy that needs to be supplied is given by the energy difference between pattern 1 and the state presented by pattern 3 (figure 1.1c). This last pattern represents an unstable equilibrium for the same magnetic field strength, with a significant energy part of $\mathcal{E}_3 = -0.14007$. To perform a transition between patterns 1 and 2, we first need to supply energy to the system to reach the barrier implied by pattern 3. Afterwards energy is released to reach the equilibrium described by pattern 2. This is further indicated in figure 1.2. ◇

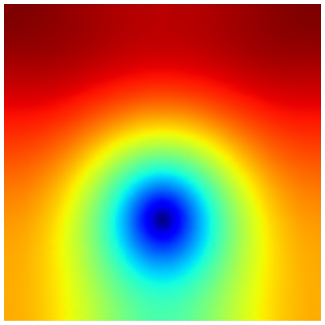
In order to determine both physically stable and unstable equilibria of a dynamical system, we will focus on the second class of methods in this thesis, where a zero of a (typically nonlinear) system (1.2) needs to be approximated.



(a) Pattern 1, with significant energy part $\mathcal{E}_1 = -0.23387$. This pattern is stable.



(b) Pattern 2, with significant energy part $\mathcal{E}_2 = -0.13271$. This pattern is stable.



(c) Pattern 3, with significant energy part $\mathcal{E}_3 = -0.14007$. This pattern is unstable.

Figure 1.1: Some possible vortex patterns for a square-shaped, 2-dimensional extreme type-II superconductor of side length 3 subject to a homogeneous magnetic field with strength $\mu = 1.3$, as described in example 1.1.

A time step scheme, the Crank-Nicolson method, will be discussed as well. It will however only be used to validate results on physical stability.

1.3 Numerical continuation

Equilibria often depend in an intricate way on the physical parameters (the vector Λ in (1.1)) of the dynamical system. Due to its nonlinear nature, a slight perturbation in one of these parameters might induce a huge change in the state, possibly inducing sudden transitions in patterns or physical properties.

This sensitiveness of equilibria to the systems parameters is an important research subject. For example, vortex patterns of superconductors in small nano devices depend in an intricate way on the system parameters and the geometry of the sample. Material scientists and device engineers are designing devices that have an improved critical field and temperatures by exploiting geometrical properties and engineering the material parameters [23, 38]. There

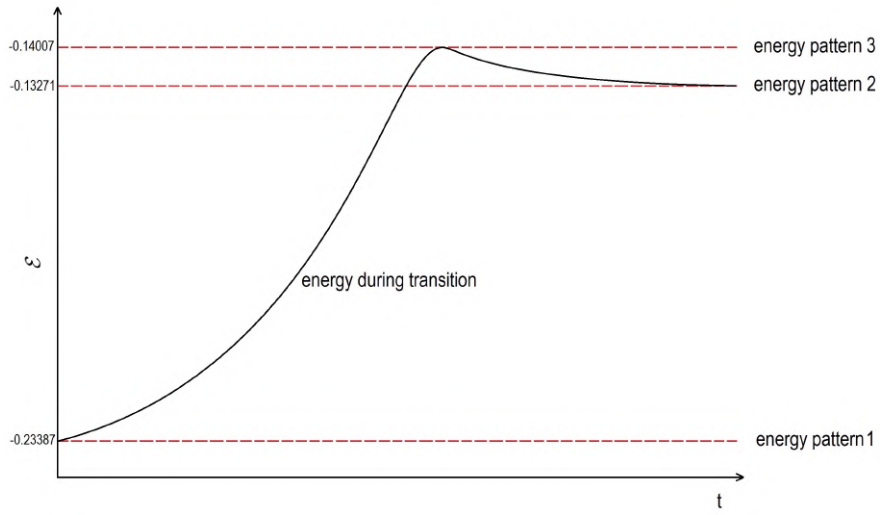


Figure 1.2: Schematic transition between the two stable patterns presented in example 1.1. t represents the time, $\mathcal{E}$ the part of the state's energy that depends on the order parameter. In order to perform a transition between patterns 1 and 2, the energy barrier implied by pattern 3 needs to be crossed.

is wide interest to understand what parameters determine the stability of superconducting states. In particular, the aim is to understand the dynamics of the transitions between vortex patterns and what needs to be changed in the geometry or parameter settings to prevent the system from making a transition that destroys a given pattern.

Under mild conditions on the dynamical system, the sensitivity of its equilibria to physical parameters is described by multiple, interconnected continuous curves (also called branches) of solutions of (1.2). The collection of these curves is called a (connected) solution landscape (or bifurcation diagram) of the dynamical system. To analyse the sensitivity of states towards parameters in a system, the appropriate solution landscape needs to be constructed. An example of a solution landscape is given in figure 1.3, showing interconnected solution curves that correspond to the problem described in example 1.1. Representative solutions for these curves are given in figure 1.4.

Except for the analysis of sensitivity, connected solution landscapes are also helpful when facing problems like the one described in example 1.1. This is further indicated in example 1.2.

Example 1.2. We again consider the set-up of example 1.1, where a transition is discussed between two stable patterns (see figure 1.1) for a fixed magnetic field strength $\mu = 1.3$. An energy barrier implied by an unstable pattern needs to be crossed for this transition to occur (see figure 1.2). Figure 1.3 shows the connected solution landscape of the problem. At magnetic field strength $\mu = 1.3$ we have multiple solutions: two stable ones that lie on curves A (pattern 1 of

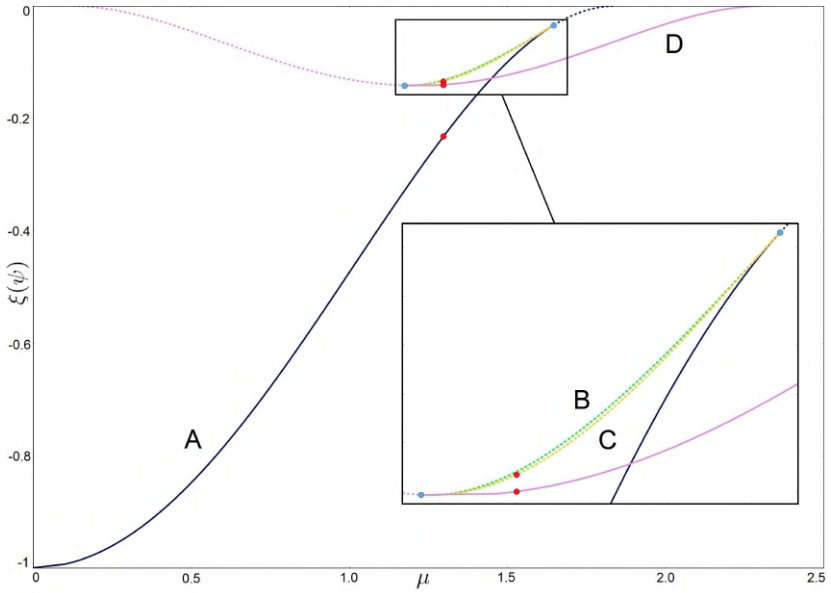


Figure 1.3: Recreated from Schlömer [92]. Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a square-shaped material with side length 3 (measured in units of its coherence length), subject to a homogeneous magnetic field. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate intersection points, red dots the points associated with the patterns given in figure 1.1. See section 9.5.1 for more details.

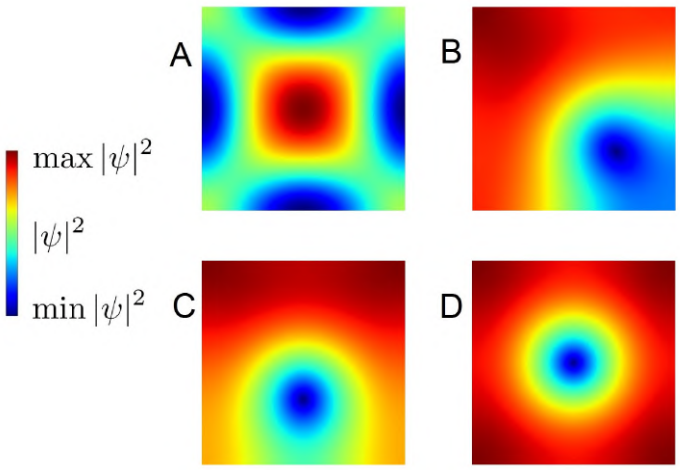


Figure 1.4: Recreated from Schlömer [92]. Representative solutions for the different curves of figure 1.3.

figure 1.1) and curve D (pattern 2), and two unstable ones on curves B and C (pattern 3). In order to perform a transition from the solution on A to the one on D, we need to add enough energy to bypass the barrier induced by the unstable solution on curve C - the unstable solution with the lowest energy that lies on a curve connecting A and D. ◇

An essential tool for the construction of a connected solution landscape is numerical continuation, used to approximate solution curves by generating finite sets of equilibrium points. Numerical continuation is based on numerical algorithms like the Newton method. Though solving nonlinear systems of the form (1.2) for a fixed set of physical parameters is already a challenge itself, using solvers for this equation in a numerical continuation setting further complicates the problem.

Furthermore, a lot of applications yield solution landscapes with hundreds of different curves. Though a relatively straight-forward algorithm (called pseudo-arclength continuation) allows for the approximation of a single one, there is also a need for techniques that automatically generate *all* of the interconnected curves when performing numerical continuation. Without these techniques, generated landscapes are possibly incomplete, giving rise to incorrect conclusions about the behaviour in the considered dynamical system.

An essential step for automatic exploration techniques is the determination of bifurcation points: equilibria of the dynamical system that mark transitions in behaviour and properties of its states. Intersections of solution curves are an important class of bifurcations, and lie at the base of automatic exploration. These intersections are called branch points. Mathematically bifurcations are defined by the (partial) Jacobian F_Ψ of the function F in equation (1.2). In bifurcation points this Jacobian contains one or multiple eigenvalues with zero real part.

Except for the determination of bifurcation points, a second essential step in automatic exploration is the construction of tangent directions to new curves, emerging from branch points. Given such a point and direction, new curves can be initialized and approximated by the pseudo-arclength continuation algorithm.

1.4 Aim of the thesis

The main goal of the thesis is to *analyse and develop robust numerical methods that generate the complete solution landscape of interconnected curves for bifurcation problems described by a partial differential equation.* We limit ourselves to dynamical systems for which the nonlinear function F, that appears in the equilibrium equation (1.2), contains a Hermitian (or self-adjoint) partial Jacobian F_Ψ. Systems with this property appear naturally in many branches of physics, e.g. in multiple models that describe phenomena of quantum mechanics. We see three main challenges, which will be studied in detail in the thesis.

1.4.1 Challenge 1: Numerical methods must be based on sparse linear algebra

To apply numerical continuation to problems derived from partial differential equations, typically large, sparse systems need to be solved or analysed. For example, some underlying problems require the (partial) Jacobian F_Ψ of (1.2): Linear systems with this operator need to be solved when the Newton method is applied, and its eigenvalues are required for e.g. stability analysis. For partial differential equations, the equilibrium equation's Jacobian is usually a large, but sparse, operator.

Techniques based on dense linear algebra cannot be used for these kinds of problems, due to unacceptable computational requirements. Instead, the used numerical methods must be based on sparse linear algebra, like Krylov methods for solving linear systems or approximation by Ritz pairs for eigenvalue problems. When developing new numerical methods, we should be aware that matrix/tensor forms of operators cannot be used. Their sparsity structure can however be exploited. In this thesis we will develop the required tools that can solve the large systems of equations appearing in numerical continuation, solely based on sparse linear algebra.

1.4.2 Challenge 2: Need for a reliable solver near bifurcation points

A second challenge concerns the determination of bifurcation points: equilibria in which the Jacobian contains one or multiple eigenvalues with zero real part. By restricting our analysis to Hermitian systems, all eigenvalues of the partial Jacobian F_Ψ will have a zero imaginary part. This excludes multiple classes of bifurcation points, like the Hopf bifurcation [72, 95], from the analysis. With the restriction, bifurcations are exactly the points for which this partial Jacobian becomes singular.

Though only a selected group of bifurcations is considered, the Jacobian's singularity induces issues when approximating these points by methods based on the Newton algorithm. Several adjustments have already been proposed to counter these issues, but these are either based on dense linear algebra [28, 56, 111], or are only able to attain a reduced accuracy [4, 42].

Instead of applying one of the adjusted methods presented in [28, 56, 111] or [4, 42], a new class of Newton methods will be derived in the thesis. The proposed algorithm is able to effectively approximate the zeros of a given function near and in the points where the Jacobian is singular. The algorithm works for large-scale systems, and only requires the Jacobian to be given as a linear operator. Its matrix form is not required. The adjusted Newton method is an essential part of the algorithms that we will use for the approximation of bifurcation points.

1.4.3 Challenge 3: Construction of tangent directions for systems with underlying symmetries

Once the bifurcation points are identified, automatic exploration requires an approximation to the tangent directions to curves emerging from these points. For small-scaled problems derived from e.g. ordinary differential equations, determining these directions is easily done by constructing and solving the algebraic branching equation (ABE) [61, 3, 10, 62, 73]: a simple quadratic equation that is solved for coefficients used to construct the tangent as a linear combination of the Jacobian's null vectors.

For partial differential equations the same is true for dynamical systems that do not contain any underlying symmetries, in this case the algebraic branching equation is sufficient. However, when symmetries are present in the system, bifurcations arise for which an extended analysis is required. These symmetries are often known in advance.

In the thesis an algorithm will be derived that constructs the tangent directions for problems that contain a two-dimensional (e.g. dihedral) symmetry. When available, the algorithm allows the use of prior knowledge on the symmetry to reduce the computational work. Except for the construction of tangent directions, prior knowledge on symmetries will also be incorporated in other algorithms discussed throughout the thesis.

1.4.4 Implementation of the numerical methods

The numerical methods discussed throughout the thesis have been implemented in Python. The package PyNCT [36], originally developed for numerical continuation in auxine transport models [34, 35], was used as a basis for the implementation. The goal of PyNCT is to automatically generate a connected solution landscape, given the equilibrium equation (and possibly its derivatives) and an initial guess of a solution. The algorithms in PyNCT are entirely based on sparse linear algebra and allow for prior knowledge on symmetries to be provided as well, which is used to reduce computational work.

There are other software packages with the same goal as PyNCT, but these have their shortcomings. Existing tools such as AUTO [31] and MatCont [30] for example, are able to generate connected solution landscapes for small systems of coupled ordinary differential equations. However, these tools are based on dense linear algebra and they cannot scale to the large sparse systems that appear in the models we will consider, like the Ginzburg-Landau equations. Furthermore, it is not possible to incorporate prior knowledge on symmetries that appear in the dynamical system, complicating the construction of a solution landscape.

Besides these tools there is LOCA [90] that is developed around sparse linear algebra, but is less easy to use and requires knowledge in advanced C++ programming. Furthermore it does not include an automatic exploration functionality. In particular branch switching at bifurcation points is not provided. Contrary to PyNCT, the discussed tools ([30, 31, 90]) do however contain bifurcation tracking algorithms, used to determine how the location of a bifurcation is perturbed when a second physical parameter changes.

1.5 Outline of the thesis

This thesis focusses on the development of the required tools for automatically exploring a connected solution landscape for large-scale dynamical systems. The derived algorithms are solely based on sparse linear algebra. When relevant, pseudo-code for the discussed numerical methods will be provided in an appendix at the end of the appropriate chapter. A summary of the forthcoming chapters is given below.

Chapter 2. Applications We start by describing several dynamical systems and other examples. These systems will be used throughout the thesis to illustrate some of the numerical techniques. Connected solution landscapes of these problems will be constructed further in the thesis (chapter 9). The information of the systems that is required for this construction, will be derived in chapter 2 as well. The Ginzburg-Landau model, our main application, is discussed in detail.

Chapter 3. Review of numerical methods A lot of different numerical methods are required for the construction of an approximate connected solution landscape. This chapter gives an overview of some of the algorithms that lie at the base of numerical continuation, even though their immediate application might not be apparent. We start by discussing the approximation of eigenpairs by Ritz pairs, and continue with a discussion of Krylov solvers. The importance of deflation techniques in these methods is underlined for ill-conditioned linear operators. The chapter also discusses the nonlinear conjugate gradients and Crank-Nicolson methods.

Chapter 4. The Newton-Krylov method near bifurcation points One of the main contributions of the thesis will be discussed in this chapter. Though the standard Newton-Krylov method is an efficient solver for general nonlinear problems, its convergence is obstructed when the Jacobian of the equation is ill-conditioned. This happens, for example, in the equilibrium equation of dynamical systems near bifurcation points. The chapter derives the necessary adjustments to the Newton-Krylov method, eventually yielding an alternative that also works for nonlinear problems with an ill-conditioned Jacobian.

Chapter 5. The Newton-Krylov method for extended nonlinear systems To use the adjusted Newton-Krylov methods derived in chapter 4 in a numerical continuation setting, they need to be combined with a block elimination technique. This technique allows to solve a nonlinear system with a linear extension by application of the numerical tools used for the original system. The chapter starts with a discussion of the standard method, and again derives the necessary adjustments for application to problems with an ill-conditioned Jacobian.

Chapter 6. Calculation of solution curves This chapter discusses pseudo-arclength continuation, the numerical technique used for the approximation of a single curve of a solution landscape. We start with a summary of the under-

lying theory, and discuss complications that arise when the dynamical system contains (continuous) symmetries. Except for the pseudo-arclength continuation method, we also describe several algorithms for the approximation of bifurcation points. The chapter ends with a discussion on physical stability.

Chapter 7. Automatic exploration The necessary requisites for automatically exploring a connected solution landscape are derived in this chapter, focussing on the construction of tangent directions to curves that emanate from a given bifurcation point. An algorithm is derived for a specific case that appears in our applications, and techniques are discussed that allow a reduction of the computational work by exploiting prior knowledge on the dynamical system's symmetries.

Chapter 8. Implementation in Python This chapter describes the last details of numerical continuation, and contains the eventual algorithm used to generate a connected solution landscape for a given nonlinear function (possibly derived from a dynamical system). An implementation of the required techniques was made in Python. The structure of this implementation is discussed, and some other notes on the code are made as well.

Chapter 9. Numerical results Connected solution landscapes, for the dynamical systems and other examples of chapter 2, are provided and discussed in chapter 9. Landscapes with many interconnected curves were generated, indicating the effectiveness of the derived methods.

Chapter 10. Conclusions & outlook The final chapter of the thesis contains a summary of the main numerical techniques, results and contributions. Possible topics for future research are also proposed.

CHAPTER 2

Applications

"What is mathematics? It is only a systematic effort of solving puzzles posed by nature."

– ***Shakuntala Devi*** –

Chapter highlights:

- We derive the equilibrium equation for multiple dynamical systems and other examples.
- We describe the properties of the Ginzburg-Landau model in detail, including a proposition on its symmetries and a possible preconditioner to use for derived linear problems.
- The results in this chapter are mainly based on the following references: [37, 92, 93, 107].

2.1 Introduction

This chapter contains descriptions of dynamical systems and other examples that will be used throughout the thesis. Some of these will mainly be used to test the different techniques and their implementations, others yield more interesting results, which are analysed in detail in chapter 9. To apply the numerical continuation techniques described in chapters 6 and 7, a (nonlinear) function of the form

$$\mathcal{F} : V \subseteq \mathbb{C}^n \times \mathbb{R} \to W \subseteq \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu) \tag{2.1}$$

needs to be provided. Numerical continuation analyses the zeros of this function. We will call the vector $\psi \in V \subseteq \mathbb{C}^n$ and the value $\mu \in \mathbb{R}$ respectively the state and (physical) parameter of the system described by $\mathcal{F}$. For dynamical systems, the state is typically a discretized function of space. Examples of physical parameters include the temperature and the strength of a magnetic field.

Note that we only consider problems with a Hermitian (or self-adjoint) partial Jacobian $\mathcal{F}_\psi$ in the thesis. For some of the examples discussed in this chapter a preconditioner is derived, which is used to reduce computational work in the underlying numerical algorithms (see e.g. chapter 3). The preconditioners discussed in the current chapter are Hermitian and positive definite (except in possibly some uninteresting cases, in which the preconditioner is still positive semi-definite, see e.g. section 2.5.6).

In the remainder of this chapter, we will derive the required equations of form (2.1) for several examples.

2.2 Equation of a circle

One of the simplest examples numerical continuation can be applied to, is the equation of a circle. The equation we consider is given by

$$\psi^2 + \mu^2 - 1 = 0$$

with $\psi, \mu \in \mathbb{R}$. The associated function to use for numerical continuation is provided by

$$\mathcal{F} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} : (\psi, \mu) \to \psi^2 + \mu^2 - 1. \tag{2.2}$$

Though it lacks physical relevance, the example is often used to test the implementation of a numerical continuation algorithm. Its bifurcation diagram is predictable and consists of only a single solution curve, removing the requirement of automatic exploration techniques. If the constructed diagram does not represent a circle with solutions that satisfy the relation

$$\psi = \pm\sqrt{1 - \mu^2},$$

the implementation or used algorithm contains errors. Derivatives of $\mathcal{F}$ are given by (for $\psi \in \mathbb{R}, \mu \in \mathbb{R}$)

$$\begin{aligned} &\mathcal{F}_\psi(\psi, \mu) = 2\psi, && \mathcal{F}_\mu(\psi, \mu) = 2\mu, && \\ &\mathcal{F}_{\psi\psi}(\psi, \mu) = 2, && \mathcal{F}_{\psi\mu}(\psi, \mu) = 0, && \mathcal{F}_{\mu\mu}(\psi, \mu) = 2. \end{aligned}$$

If not specified otherwise, we will use the values $\psi^{(0)} = 1, \mu^{(0)} = 0$ as a starting point when executing numerical continuation on the example.

2.3 Intersection of two cylinders

An example with two solution curves is described by the equation

$$\begin{cases} \psi_1^2 + \mu^2 - 1 = 0, \\ \psi_2^2 + \mu^2 - 1 = 0, \end{cases} \tag{2.3}$$

with $\psi_1, \psi_2, \mu \in \mathbb{R}$. This equation represents the intersection of two cylinders (see figure 2.1), which yields two ellipses that intersect at the points

$$\psi = \begin{pmatrix} 0 & 0 \end{pmatrix}^T, \mu = 1 \quad \text{and} \quad \psi = \begin{pmatrix} 0 & 0 \end{pmatrix}^T, \mu = -1.$$

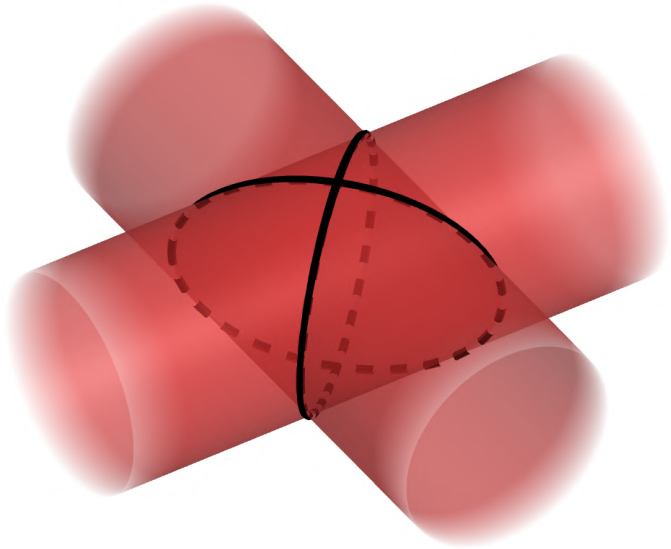


Figure 2.1: Intersection of two cylinders. The intersection consists of two ellipses and is described by equation (2.3).

The associated equation for numerical continuation is given by

$$\mathcal{F} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R} : (\psi, \mu) \to \begin{pmatrix} \psi_1^2 + \mu^2 - 1 \\ \psi_2^2 + \mu^2 - 1 \end{pmatrix}. \tag{2.4}$$

This example again lacks physical relevance, but is often used to test the automatic exploration part of a numerical continuation algorithm: when the implementation is applied, a bifurcation diagram that contains both solution curves should be constructed. Derivatives of $\mathcal{F}$ are given by (for $\psi \in \mathbb{R}^2, \mu \in \mathbb{R}$)

$$\begin{aligned}
&\mathcal{F}_\psi(\psi, \mu) : \mathbb{R}^2 \to \mathbb{R}^2 : \phi \to \begin{pmatrix} 2\psi_1\phi_1 \\ 2\psi_2\phi_2 \end{pmatrix}, \\
&\mathcal{F}_\mu(\psi, \mu) = \begin{pmatrix} 2\mu \\ 2\mu \end{pmatrix}, \\
&\mathcal{F}_{\psi\psi}(\psi, \mu) : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2 : (\phi, \xi) \to \begin{pmatrix} 2\phi_1\xi_1 \\ 2\phi_2\xi_2 \end{pmatrix}, \\
&\mathcal{F}_{\psi\mu}(\psi, \mu) : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2 : (\phi, \tau) \to \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \\
&\mathcal{F}_{\mu\mu}(\psi, \mu) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}.
\end{aligned}$$

Unless specified otherwise, the point

$$\psi = \begin{pmatrix} 1 & 1 \end{pmatrix}^T, \quad \mu = 0$$

will be used as the starting point when numerical continuation is applied.

2.4 The Liouville-Bratu-Gelfand equation

For the next example, we consider the following partial differential equation, defined on a domain Ω [107]:

$$\frac{\partial \Psi(x,t)}{\partial t} = \Delta\Psi(x,t) - \lambda\left(\Psi(x,t) - \mu e^{\Psi(x,t)}\right),$$
$$\text{with } \Psi : \Omega \times \mathbb{R} \to \mathbb{R},$$

where Δ represents the Laplacian operator, $\lambda \in \mathbb{R}$ and $\mu \in \mathbb{R}$ are parameters. This equation is used in multiple aspects of physics to describe certain dynamical systems. Examples include the density distribution of matter under its own gravitation [19], the thermal reaction process in a combustible, non-deformable material placed in a spherical vessel [60], and the space charge of electricity around a glowing wire [83]. We will only consider domains of the form $\Omega = [-a, a]$, a line segment of length $a \in \mathbb{R}$, and $\Omega = [-a, a]^2$, a square with side length a.

To study steady states, we set $\frac{\partial \Psi(x,t)}{\partial t} = 0$ in this equation and drop the dependency of Ψ on the time t. This yields the equation:

$$-\Delta\Psi(x) + \lambda\left(\Psi(x) - \mu e^{\Psi(x)}\right) = 0. \tag{2.5}$$

To apply numerical continuation, we require a discretized version of this equation. In the line segment case, we discretize the domain Ω by considering n grid points given by

$$\forall i = 1, \ldots, n : x_i = -a + 2a\frac{i-1}{n-1}.$$

We define $\psi_i = \Psi(x_i)$ for this case. If the mesh represents a square, the n grid points (x_i, y_j) (for $i, j = 1, \ldots, k$, with $k = \sqrt{n}$) are defined by:

$$\forall i = 1, \ldots, k : x_i = -a + 2a\frac{i-1}{k-1},$$
$$\forall j = 1, \ldots, k : y_j = -a + 2a\frac{j-1}{k-1}.$$

In this case we define ψ as

$$\psi = \begin{pmatrix} \Psi(x_1, y_1) \\ \Psi(x_1, y_2) \\ \vdots \\ \Psi(x_1, y_k) \\ \Psi(x_2, y_1) \\ \vdots \\ \Psi(x_2, y_k) \\ \vdots \\ \Psi(x_k, y_1) \\ \vdots \\ \Psi(x_k, y_k) \end{pmatrix}.$$

The discretized version of (2.5) becomes

$$-A\psi + \lambda\left(\psi - \mu e^{\psi}\right) = 0, \tag{2.6}$$

with A the discretization of the Laplacian operator, where Neumann boundary conditions are considered. When used as an example for numerical continuation, we will set $\lambda = 10$ and $a = 0.5$ in the equation. The required function to use in the algorithms becomes [107]

$$\mathcal{F} : \mathbb{R}^n \times \mathbb{R} : (\psi, \mu) \to -A\psi + 10\left(\psi - \mu e^{\psi}\right). \tag{2.7}$$

As a starting point we use the trivial solution $\psi = 0$, $\mu = 0$ of $\mathcal{F}(\psi, \mu) = 0$, unless specified otherwise. The inner products

$$\langle \cdot, \cdot \rangle_h = \frac{1}{n-1}\langle \cdot, \cdot \rangle_2, \tag{2.8}$$

$$\langle \cdot, \cdot \rangle_{h^2} = \left(\frac{1}{\sqrt{n}-1}\right)^2 \langle \cdot, \cdot \rangle_2 \tag{2.9}$$

will be used in the algorithms and solvers for respectively the line segment and square meshes.

For the line segment ($\Omega = [-a, a]$), the resulting bifurcation diagram will contain multiple solution curves, which will be generated by the same techniques as used for the example described in section 2.3. The case where Ω represents a square is an example of how symmetry complicates the construction of a bifurcation diagram. Due to the possibility of more than two solution curves intersecting in a single point, the techniques described in sections 7.4 or 7.5 will be required. The derivatives of $\mathcal{F}$ (for $\psi \in \mathbb{R}^n$, $\mu \in \mathbb{R}$) are given by:

$$\begin{aligned}
&\mathcal{F}_{\psi}(\psi, \mu) : \mathbb{R}^n \to \mathbb{R}^n : \phi \to -A\phi + 10\left(\phi - \mu e^{\psi}\phi\right), \\
&\mathcal{F}_{\mu}(\psi, \mu) = -10e^{\psi}, \\
&\mathcal{F}_{\psi\psi}(\psi, \mu) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n : (\phi, \xi) \to -10\mu e^{\psi}\phi\xi, \\
&\mathcal{F}_{\psi\mu}(\psi, \mu) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n : (\phi, \tau) \to -10\tau e^{\psi}\phi \\
&\mathcal{F}_{\mu\mu}(\psi, \mu) = 0.
\end{aligned}$$

Note that since A is a symmetric matrix, the linear operator $\mathcal{F}_{\psi}(\psi, \mu)$ is symmetric as well. This induces some interesting properties, for example, all of the eigenvalues of $\mathcal{F}_{\psi}(\psi, \mu)$ will be real for any $\psi \in \mathbb{R}^n$, $\mu \in \mathbb{R}$.

To perform numerical continuation efficiently, a fast method for solving linear systems of the form

$$\mathcal{F}_{\psi}(\psi, \mu)\phi = b$$

for certain $b \in \mathbb{R}^n$, needs to be derived. We will approximate the solutions of these systems with a Krylov method (see section 3.3). To reduce the computational work, a preconditioner is used for the Liouville-Bratu-Gelfand problem. This preconditioner consists of the inverse of the linear operator

$$\mathcal{R}(\psi, \mu) : \mathbb{R}^n \to \mathbb{R}^n : \phi \to -A\phi + 10\phi. \tag{2.10}$$

The linear operator $\mathcal{R}(\psi, \mu)$ consists of the part of $\mathcal{F}_\psi(\psi, \mu)$ that is independent of the state ψ and physical parameter μ. Though independent of these variables, we choose to use the notation $\mathcal{R}(\psi, \mu)$ for generality. The used preconditioner is symmetric and positive definite.

2.5 The Ginzburg-Landau equation

The main application that is considered in the thesis is the extreme type-II Ginzburg-Landau equation, a nonlinear model that appears when describing vortices in a wide range of superconductors.

These vortices organise themselves in regular patterns, for small nano devices these patterns are very sensitive to system parameters and the geometry of the considered sample. Scientists are trying to improve the critical field and temperatures of superconductors by exploiting and engineering the properties of geometrical and material parameters [23] or by understanding the effect of the geometry on the vortex dynamics [38]. Numerical continuation allows to predict the different patterns that might appear theoretically, as well as the values for which transitions between these patterns happen.

We start this section with a brief introduction to the different types and states of superconductors.

2.5.1 An introduction to superconductors

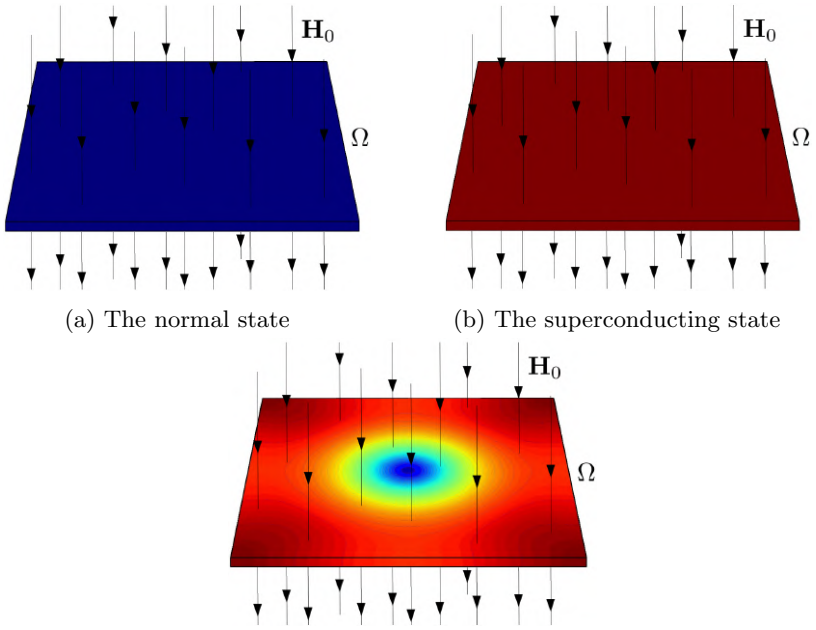
Superconductors are materials that, when below a certain characteristic temperature (T_{c_1}), expel magnetic fields and exhibit a complete loss of electrical resistivity [44, 105]. We will consider mesoscopic samples of superconducting material that occupy an open, bounded region Ω of the two-dimensional Euclidean space, subject to an external, homogeneous magnetic field $\mathbf{H}_0$ (see figure 2.2). The total magnetic field is denoted as $\mathbf{B} : \mathbb{R}^3 \to \mathbb{R}^3$.

A superconducting material can adopt different states, depending on the temperature T and strength μ of the applied magnetic field $\mathbf{H}_0$. We distinguish the normal (or homogeneously non-superconducting), the mixed and the (homogeneously) superconducting state.

Material parameters of the sample, specifically the coherence length ξ and the penetration depth λ, determine the type of the superconductor [44, 105]. If the ratio $\kappa = \frac{\lambda}{\xi}$ is smaller than $1/\sqrt{2}$ it is said to be of type I, otherwise the superconductor is of type II. For extreme type-II superconductors the sample's penetration depth dominates its coherence length ($\kappa \gg 1$) [20, 74, 92].

Each type behaves like a normal conductor in the normal state (figure 2.2a). The total magnetic field $\mathbf{B}$ entirely penetrates the sample in this case. The normal state occurs for high temperatures ($T > T_{c_2}$) or for magnetic field strengths above a certain critical value μ_{c_2}.

In the homogeneously superconducting state (figure 2.2b) the total magnetic field $\mathbf{B}$ is expelled from the interior of the sample and the material exhibits zero electrical resistance. The magnetic field is not entirely expelled, instead it penetrates the superconductor to a very small distance characterized by the penetration depth λ [64]. Note that for extreme type-II superconductors



(a) The normal state

(b) The superconducting state

(c) The mixed state

Figure 2.2: Possible states of an extreme type-II superconducting material, subject to an external homogeneous magnetic field $\mathbf{H}_0$. Red and blue parts of the domain correspond to respectively a high and low Cooper pair density.

this distance is more profound, for $\kappa \to \infty$ the field $\mathbf{B}$ even coincides with the external one ($\mathbf{H}_0$) [92]. The superconducting state occurs for low temperatures ($T < T_{c_1}$) provided that the external magnetic field's strength is sufficiently low ($\mu < \mu_{c_1}$).

A third state, called the mixed state (figure 2.2c), occurs for temperatures and magnetic field strengths between two critical values ($T_{c_1} < T < T_{c_2}$ and $\mu_{c_1} < \mu < \mu_{c_2}$) [49]. In this case the material will only be locally penetrated by the total magnetic field $\mathbf{B}$. In type-II superconductors circular vortices of non-superconducting currents will appear in the material [44]. In large samples, these vortices organise themselves in regular patterns known as the Abrikosov lattice [1]. Superconductors of type I typically cannot adopt the mixed state, in this case the critical values for the transitions are equal: $T_{c_1} = T_{c_2}$ and $\mu_{c_1} = \mu_{c_2}$. When the critical temperature or field strength is crossed, the material switches abruptly from the superconducting to the normal state [44]. For certain mesoscopic samples however, mixed state patterns can occur even for type-I superconductors [9].

In the thesis we will consider mesoscopic samples of extreme type-II superconductors. The amount of superconductivity at each location of the sample Ω is measured by the density of superconducting charge carriers, also called Cooper pairs. We denote this density by $\rho_C : \Omega \cup \partial\Omega \to \mathbb{R}$.

2.5.2 Derivation of the Ginzburg-Landau equation

Both the total magnetic field $\mathbf{B}$ and the density of Cooper pairs ρ_C are determined by the Ginzburg-Landau system [37], which will be described in this section. The derivation is based on the analysis done in Schlömer [92]. We only consider extreme type-II superconductors in the thesis, for which the system decouples [20, 92].

For an open, bounded domain $\Omega \subset \mathbb{R}^2$, with a piecewise smooth boundary $\partial\Omega$, the Ginzburg-Landau problem is derived by minimizing the Gibbs free energy functional [37]

$$G(\Psi, \mathbf{A}(\mu)) - G_n = \xi \frac{|\alpha|^2}{\beta} \int_\Omega \Big[-|\Psi|^2 + \frac{1}{2}|\Psi|^4 + \|-\mathrm{i}\nabla\Psi - \mathbf{A}(\mu)\Psi\|^2 + \kappa^2(\nabla \times \mathbf{A}(\mu))^2 - 2\kappa^2(\nabla \times \mathbf{A}(\mu)) \cdot \mathbf{H}_0 \Big] d\Omega. \tag{2.11}$$

The state $(\Psi, \mathbf{A}(\mu))$ is in the natural energy space such that the integral is well-defined. $\Psi \in H^2_{\mathbb{C}}(\Omega)$ represents a scalar-valued function and is commonly referred to as the order parameter, and the magnetic vector potential corresponding to the total magnetic field $\mathbf{B}$ is given by $\mathbf{A}(\mu) \in H^2_{\mathbb{R}^2}(\Omega)$. This vector potential depends on the parameter $\mu \in \mathbb{R}$, which represents the strength of the applied magnetic field $\mathbf{H}_0$. The magnetic field $\mathbf{B}$ and Cooper pair density ρ_C are determined by the state $(\Psi, \mathbf{A}(\mu))$ through

$$\mathbf{B} = \nabla \times \mathbf{A}(\mu), \tag{2.12}$$

$$\rho_C = |\Psi|^2. \tag{2.13}$$

The constant G_n that appears in (2.11) represents the energy associated with the normal (non-superconducting) state. The total Gibbs free energy depends upon the impinging magnetic field $\mathbf{H}_0$ and the material parameters $\alpha, \beta, \lambda, \xi \in \mathbb{R}$. It is presented in its dimensionless form, where the domain Ω is scaled in units of the coherence length ξ. As stated in section 2.5.1, the type of the superconductor is completely determined by the ratio $\kappa = \lambda/\xi$, with λ the penetration depth.

Using standard calculus of variations, minimization of the Gibbs free energy functional gives rise to the Ginzburg-Landau equation [37]: a boundary-value problem in the unknowns Ψ and $\mathbf{A}(\mu)$. For extreme type-II superconductors the limit $\kappa \to \infty$ is considered, in this case the Ginzburg-Landau problem decouples for Ψ and $\mathbf{A}(\mu)$ [92]. The magnetic vector potential $\mathbf{A}(\mu)$ is determined by the applied magnetic field $\mathbf{H}_0$ through the system

$$\begin{cases} \nabla \times (\nabla \times \mathbf{A}(\mu)) = 0 & \text{in } \Omega, \\ \mathbf{n} \times (\nabla \times \mathbf{A}(\mu)) = \mathbf{n} \times \mathbf{H}_0 & \text{on } \partial\Omega. \end{cases} \tag{2.14}$$

With the magnetic vector potential determined, the order parameter Ψ is de-

rived by solving the equation

$$\begin{aligned}&\mathcal{G} : X \times \mathbb{R} \to Y,\\ &0 = \mathcal{G}(\Psi, \mu) = \begin{cases}(-\mathrm{i}\nabla - \mathbf{A}(\mu))^2\Psi - \Psi(1 - |\Psi|^2) & \text{in } \Omega,\\ \mathbf{n} \cdot (-\mathrm{i}\nabla - \mathbf{A}(\mu))\Psi & \text{on } \partial\Omega.\end{cases}\end{aligned} \tag{2.15}$$

The space X corresponds to the natural energy space over Ω associated with the Gibbs energy (2.11) and Y to its dual space. The part

$$\mathcal{K}(\mu) = (-\mathrm{i}\nabla - \mathbf{A}(\mu))^2$$

is often referred to as the kinetic energy operator [93].

Numerical continuation will be applied to the discretization of (2.15), described in section 2.5.5. A magnetic vector potential $\mathbf{A}_0$ is a priori defined, such that $\nabla \times \mathbf{A}_0$ equals the external magnetic field strength $\mathbf{H}_0$ of strength $\mu = 1$. The potential

$$\mathbf{A}(\mu) = \mu\mathbf{A}_0 \tag{2.16}$$

is then used in the discretization of (2.15). Remember that by taking the limit $\kappa \to \infty$, the magnetic fields $\mathbf{H}_0$ and $\mathbf{B}$ coincide (see section 2.5.1), so (2.12) is satisfied.

Note that a discretized version of the inner product

$$\langle \cdot, \cdot \rangle_{\mathbb{R}} = \mathfrak{Re}\langle \cdot, \cdot \rangle_{L^2_{\mathbb{C}}(\Omega)} \tag{2.17}$$

is used in the algorithms and solvers of numerical continuation when applied to the Ginzburg-Landau example, this inner product coincides with the natural one in $(L^2_{\mathbb{R}}(\Omega))^2$ [92].

2.5.3 Underlying symmetries

One of the factors that influences the formed patterns is the geometry - more specifically the symmetries - of the sample. In Schlömer [92] a square sample subject to a perpendicular, homogeneous magnetic field is considered. The D_4 symmetry of this mesh induces an invariance (see definition 6.8) of (2.15) under the symmetry group $S^1 \times D_4$. A more general result is given in proposition 2.1.

Proposition 2.1. Let the domain $\Omega \subset \mathbb{R}^2$ and the magnetic vector potential $A(\mu)$ both be invariant under the actions of a dihedral group $D_m = \langle \tau_\omega, \sigma \rangle$ (with $\omega = 2\pi/m$, $m \in \mathbb{N}$) defined by

$$\begin{aligned}&\tau_\omega : \mathbb{R}^2 \to \mathbb{R}^2 : \begin{pmatrix}x\\y\end{pmatrix} \to \begin{pmatrix}\cos(\omega) & -\sin(\omega)\\ \sin(\omega) & \cos(\omega)\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix},\\ &\sigma : \mathbb{R}^2 \to \mathbb{R}^2 : \begin{pmatrix}x\\y\end{pmatrix} \to \begin{pmatrix}-x\\y\end{pmatrix},\end{aligned}$$

then (2.15) is invariant under the actions of $S^1 \times D_m = \langle \theta_\eta, \tau_\omega, \sigma \rangle$, given by

$$\begin{aligned}&\theta_\eta : X \to X : \Psi \to e^{\mathrm{i}\eta}\Psi \quad (\forall \eta \in [-\pi, \pi]),\\ &\tau_\omega : X \to X : \Psi(x, y) \to \Psi(\tau_\omega(x, y)),\\ &\sigma : X \to X : \Psi(x, y) \to \overline{\Psi(-x, y)}.\end{aligned}$$

A similar result holds for invariance under the actions of the cyclic group $C_m = \langle \tau_\omega \rangle$.

The continuous S^1 symmetry is independent of the domain Ω and will further complicate the construction of bifurcation diagrams. This symmetry is also referred to as phase symmetry [92]. Due to its continuity, every solution Ψ of (2.15) is part of a complete family of solutions given by [93]

$$\{\theta_\eta \Psi | \theta_\eta \in S^1\}. \tag{2.18}$$

We will only consider a single representative for each family of the form (2.18). All of the solutions in one such family have the same density of Cooper pairs (we have $|\Psi| = |\theta_\eta \Psi|$ for each $\theta_\eta \in S^1$).

Remark 2.2. The phase symmetry of (2.15) is a special case of the Gauge invariance: if a given state $(\Psi, \mathbf{A}(\mu))$ minimizes the Gibbs free energy functional (2.11), so does the state $(\Upsilon, \mathbf{Q}(\mu))$ defined by

$$\Upsilon = \Psi e^{\mathrm{i}\kappa\Phi}, \qquad\qquad \mathbf{Q}(\mu) = \mathbf{A}(\mu) + \nabla\Phi,$$

for each possible choice $\Phi \in H^2_{\mathbb{C}}(\Omega)$ [37]. The phase symmetry is derived from this by choosing Φ a constant function. Note that the Gauge invariance is valid for general superconductors (not necessarily of extreme type-II).

2.5.4 Derivatives and their properties

The Jacobian operator $\mathcal{G}_\Psi(\Psi, \mu)$

To derive the partial Jacobian $\mathcal{G}_\Psi(\Psi, \mu)$ for $\Psi \in X$, $\mu \in \mathbb{R}$, take $\Psi, \delta\Psi \in X$, $\mu \in \mathbb{R}$. We have [92]:

$$\begin{aligned}
&\mathcal{G}(\Psi + \delta\Psi, \mu) - \mathcal{G}(\Psi, \mu) \\
&\quad = (-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 (\Psi + \delta\Psi) - (\Psi + \delta\Psi)\left(1 - \overline{(\Psi + \delta\Psi)}(\Psi + \delta\Psi)\right) \\
&\qquad - (-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 \Psi + \Psi\left(1 - \overline{\Psi}\Psi\right) \\
&\quad = (-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 \delta\Psi + 2\delta\Psi|\Psi|^2 + \overline{\delta\Psi}\Psi^2 + \Psi|\delta\Psi|^2 \\
&\qquad - \delta\Psi + \Psi|\delta\Psi|^2 + \overline{\Psi}\delta\Psi^2 + \delta\Psi|\delta\Psi|^2.
\end{aligned}$$

Neglecting higher-order terms in $\delta\Psi$, we obtain the Jacobian operator [92]:

$$\mathcal{G}_\Psi(\Psi, \mu) : X \to Y : \Phi \to \left((-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 - 1 + 2|\Psi|^2\right)\Phi + \Psi^2\overline{\Phi}. \tag{2.19}$$

The operator $\mathcal{G}_\Psi(\Psi, \mu)$ is linear when defined over X and Y as $\mathbb{R}$-vector spaces and is self-adjoint with respect to inner product (2.17) [92]. Its spectrum is a subset of $\mathbb{R}$. If (Ψ_s, μ_s) is a solution of (2.15), $\mathrm{i}\Psi_s$ is a null vector of $\mathcal{G}_\Psi(\Psi_s, \mu_s)$ [92, 93]. We indeed have [92]:

$$\begin{aligned}
\mathcal{G}_\Psi(\Psi_s, \mu_s)\,\mathrm{i}\Psi_s &= (-\mathrm{i}\nabla - \mathbf{A}(\mu))^2\,\mathrm{i}\Psi_s - \mathrm{i}\Psi_s + 2|\Psi_s|^2\,\mathrm{i}\Psi_s + \Psi_s^2\overline{\mathrm{i}\Psi_s} \\
&= \mathrm{i}\Psi_s\left(1 - |\Psi_s|^2\right) - \mathrm{i}\Psi_s + 2\,\mathrm{i}|\Psi_s|^2\Psi_s - \mathrm{i}|\Psi_s|^2\Psi_s \\
&= 0.
\end{aligned}$$

The existence of this null vector is a consequence of the continuous S^1 symmetry (see proposition 2.1) [92, 93]. The singularity of the Jacobian operator (2.19) in a solution hampers the convergence of classic solvers like the Newton-Krylov method, and is one of the reasons for the introduction of deflation techniques further in the thesis (see section 3.3.2).

Other derivatives

The partial Hessian $\mathcal{G}_{\Psi\Psi}(\Psi, \mu)$ is derived in a similar way as (2.19): let Ψ, Φ, $\delta\Psi \in X$ and $\mu \in \mathbb{R}$. We have

$$\begin{aligned}
\mathcal{G}_{\Psi}(\Psi + \delta\Psi, \mu)\Phi - \mathcal{G}_{\Psi}(\Psi, \mu)\Phi & \\
&= \left((-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 - 1 + 2|\Psi + \delta\Psi|^2\right)\Phi + (\Psi + \delta\Psi)^2\overline{\Phi} \\
&\quad - \left((-\mathrm{i}\nabla - \mathbf{A}(\mu))^2 - 1 + 2|\Psi|^2\right)\Phi - \Psi^2\overline{\Phi} \\
&= 2\left(\overline{\Psi}\delta\Psi\Phi + \Psi\overline{\delta\Psi}\Phi + \Psi\delta\Psi\overline{\Phi}\right) + 2|\delta\Psi|^2\Phi + (\delta\Psi)^2\,\overline{\Phi}.
\end{aligned}$$

The Hessian operator is obtained from this equation by neglecting the higher-order terms in $\delta\Psi$:

$$\mathcal{G}_{\Psi\Psi}(\Psi, \mu) : X \times X \to Y : (\Phi_1, \Phi_2) \to 2\left(\overline{\Psi}\Phi_1\Phi_2 + \Psi\overline{\Phi_1}\Phi_2 + \Psi\Phi_1\overline{\Phi_2}\right).$$

Note that this operator is independent of the strength μ of the applied magnetic field. It is bilinear when defined over X and Y as $\mathbb{R}$-vector spaces. The third partial derivative of (2.15) to Ψ is again derived in a similar way:

$$\begin{aligned}
\mathcal{G}_{\Psi\Psi\Psi}(\Psi, \mu) : & X \times X \times X \to Y : \\
& (\Phi_1, \Phi_2, \Phi_3) \to 2\left(\overline{\Phi_1}\Phi_2\Phi_3 + \Phi_1\overline{\Phi_2}\Phi_3 + \Phi_1\Phi_2\overline{\Phi_3}\right)
\end{aligned}$$

and is independent of the order parameter Ψ as well. These independencies imply the following conditions on higher-order derivatives:

$$\forall k \geq 4 : \frac{\partial^k \mathcal{G}}{\partial \Psi^k}(\Psi, \mu) = 0, \qquad \forall k \geq 1 : \frac{\partial^k}{\partial \mu^k}\frac{\partial^2 \mathcal{G}}{\partial \Psi^2}(\Psi, \mu) = 0.$$

The partial derivatives that have not been discussed in this section so far will be approximated by applying a second-order finite difference scheme [43]. This includes the partial Jacobian $\mathcal{G}_{\mu}$ and partial Hessians $\mathcal{G}_{\Psi\mu}$ and $\mathcal{G}_{\mu\mu}$.

2.5.5 Discretization

Derivation of the discretized system

In order to apply numerical continuation, we need a discretized version of (2.15). Let $(x_1, y_1), \ldots, (x_n, y_n)$ be a set of n discretization points of the domain Ω. If possible, this set should be chosen in such a way that any symmetries of Ω are preserved. States $\Psi \in X$ will be approximated by $\psi \in \mathbb{C}^n$, with

$$\psi = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_n \end{pmatrix} = \begin{pmatrix} \Psi(x_1, y_1) \\ \vdots \\ \Psi(x_n, y_n) \end{pmatrix}.$$

Let $\{T_i\}_{i=1}^m$ ($m \in \mathbb{N}$) be the Delaunay triangulation of $(x_1, y_1), \ldots, (x_n, y_n)$ and $\{V_j\}_{j=1}^n$ its corresponding Voronoi tessellation [78]. An edge $(x_j, y_j) - (x_k, y_k)$ of a triangle T_i is denoted as $e_{j,k}$ ($\forall i = 1, \ldots, m$, with $j, k \in \{1, \ldots, n\}$). We first discretize the operator $(-\mathrm{i}\nabla - \mathbf{A}(\mu))^2$, this discretization is given by $\mathcal{K}^{(h)}(\mu)$, defined by the property [93]

$$\begin{aligned}&\forall \psi, \phi \in \mathbb{C}^n : \\ &\sum_{i=1}^{n} |V_i| \overline{\phi_i} \left(K^{(h)}(\mu)\psi \right)_i = \sum_{i=1}^{m} \sum_{\text{edges } e_{j,k} \text{ of } T_i} \alpha_{j,k}^{(i)} \Big(\big(\psi_j - U_{j,k}(\mu)\psi_k\big)\overline{\phi_j} \\ &\qquad + \big(\psi_k - \overline{U_{j,k}(\mu)}\psi_j\big)\overline{\phi_k} \Big).\end{aligned}$$

For a triangle T_i ($i \in \{1, \ldots, m\}$) consisting of the edges $e_{j,k}$, $e_{k,l}$ and $e_{l,j}$ ($j, k, l \in \{1, \ldots, n\}$), the coefficient $\alpha_{j,k}^{(i)} \in \mathbb{R}$ is given by the formula

$$\alpha_{j,k}^{(i)} = \frac{1}{2} \frac{t_{j,k}}{\sqrt{1 - t_{j,k}^2}} \qquad \text{with } t_{j,k} = \left\langle \frac{e_{k,l}}{\|e_{k,l}\|_2}, \frac{e_{l,j}}{\|e_{l,j}\|_2} \right\rangle_2.$$

The coefficients $\alpha_{k,l}^{(i)}$ and $\alpha_{l,j}^{(i)}$ are defined by a similar formula. The values $U_{j,k}(\mu) \in \mathbb{C}$ are given by

$$\forall j, k = 1, \ldots, n : U_{j,k}(\mu) = \exp\left(-\mathrm{i} \int_{(x_k, y_k)}^{(x_j, y_j)} \langle e_{j,k}, \mathbf{A}(\mu, \omega) \rangle_2 d\omega \right)$$

with $\mathbf{A}(\mu, \omega)$ the magnetic vector potential evaluated at the location $\omega \in \Omega$. Using the discretization $K^{(h)}$ of $(-\mathrm{i}\nabla - \mathbf{A}(\mu))^2$, we discretize the Ginzburg-Landau equation (2.15) as [93]

$$\begin{aligned}&\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n, \\ &0 = \mathcal{F}(\psi, \mu) = \begin{cases} \left(K^{(h)}(\mu)\psi\right)_1 - \psi_1 \left(1 - |\psi_1|^2\right) = 0, \\ \vdots \\ \left(K^{(h)}(\mu)\psi\right)_n - \psi_n \left(1 - |\psi_n|^2\right) = 0. \end{cases}\end{aligned} \tag{2.20}$$

By discretizing the equation as (2.20), the Gauge invariance (see remark 2.2) is preserved [93].

The function (2.20) is used for the numerical continuation algorithms, where the magnetic vector potential $\mathbf{A}(\mu)$ is defined as in (2.16). Unless otherwise specified, we use the trivial solution $\psi = 1, \mu = 0$ as a starting point. In the algorithms and solvers the inner product

$$\langle \cdot, \cdot \rangle_{\mathbb{R}} = \mathfrak{Re} \langle \cdot, \cdot \rangle_2 \tag{2.21}$$

will be used. This is the discretized version of (2.17). The partial Jacobian $\mathcal{F}_\psi$ and Hessian $\mathcal{F}_{\psi\psi}$ are discretized in a similar way as (2.20). The partial Jacobian becomes [93]:

$$\mathcal{F}_\psi(\psi, \mu) : \mathbb{C}^n \to \mathbb{C}^n : \phi \to \mathcal{K}^{(h)}\phi - \phi + 2 \begin{pmatrix} |\psi_1|^2 \phi_1 \\ \vdots \\ |\psi_n|^2 \phi_n \end{pmatrix} + \begin{pmatrix} \psi_1^2 \overline{\phi_1} \\ \vdots \\ \psi_n^2 \overline{\phi_n} \end{pmatrix}. \tag{2.22}$$

The partial Hessian is given by

$$\mathcal{F}_{\psi\psi}(\psi,\mu):\mathbb{C}^n\times\mathbb{C}^n\to\mathbb{C}^n:(\phi,\varphi)\to 2\begin{pmatrix}\overline{\psi_1}\phi_1\varphi_1+\psi_1\overline{\phi_1}\varphi_1+\psi_1\phi_1\overline{\varphi_1}\\ \vdots\\ \overline{\psi_n}\phi_n\varphi_n+\psi_n\overline{\phi_n}\varphi_n+\psi_n\phi_n\overline{\varphi_n}\end{pmatrix}.$$

Properties of the discretized system

Most properties of the discretized system are inherited from the continuous one. Proposition 2.3 describes the symmetries of (2.20) and is similar to proposition 2.1.

Proposition 2.3. Let the domain $\Omega\subset\mathbb{R}^2$, its set of discretization points $(x_1,y_1),\ldots,(x_n,y_n)$ and the magnetic vector potential $A(\mu)$ be invariant under the actions of a dihedral group $D_m=\langle\tau_\omega,\sigma\rangle$ (with $\omega=2\pi/m$, $m\in\mathbb{N}$). Then (2.20) is invariant under the actions of $S^1\times D_m=\langle\theta_\eta,\tau_\omega,\sigma\rangle$, given by

$$\begin{aligned}&\theta_\eta:\mathbb{C}^n\to\mathbb{C}^n:\psi\to e^{\mathrm{i}\eta}\psi\quad(\forall\eta\in[-\pi,\pi]),\\&\tau_\omega:\mathbb{C}^n\to\mathbb{C}^n:\psi\to P_{\tau_\omega}\psi,\\&\sigma:\mathbb{C}^n\to\mathbb{C}^n:\psi\to\overline{P_\sigma\psi},\end{aligned}$$

with P_{τ_ω} and P_σ permutation matrices such that $\forall\Psi\in X$:

$$\begin{aligned}P_{\tau_\omega}\begin{pmatrix}\Psi(x_1,y_1)\\ \vdots\\ \Psi(x_n,y_n)\end{pmatrix}&=\begin{pmatrix}\Psi(\tau_\omega(x_1,y_1))\\ \vdots\\ \Psi(\tau_\omega(x_n,y_n))\end{pmatrix},\\P_\sigma\begin{pmatrix}\Psi(x_1,y_1)\\ \vdots\\ \Psi(x_n,y_n)\end{pmatrix}&=\begin{pmatrix}\Psi(-x_1,y_1)\\ \vdots\\ \Psi(-x_n,y_n)\end{pmatrix}.\end{aligned}$$

A similar result holds for invariance under the actions of the cyclic group $C_m=\langle\tau_\omega\rangle$.

Each solution ψ of (2.20) is again part of a family of solutions $\{\theta_\eta\psi|\theta_\eta\in S^1\}$, and we will only consider a single representative for each such family when performing numerical continuation. As in the continuous case, the S^1 symmetry induces a null vector for the partial Jacobian $\mathcal{F}_\psi(\psi_s,\mu_s)$ evaluated at a solution ψ_s,μ_s of (2.20) [93]:

$$\mathcal{F}_\psi(\psi_s,\mu_s)\,\mathrm{i}\psi_s=0. \tag{2.23}$$

The self-adjointness of the continuous Jacobian (2.19) with respect to the inner product (2.17) induces the same property in the discretized case [93]: the partial Jacobian $\mathcal{F}_\psi(\psi,\mu)$ is self-adjoint with respect to inner product (2.21) for each $\psi\in\mathbb{C}^n$, $\mu\in\mathbb{R}$. The spectrum of this operator is again a subset of $\mathbb{R}$.

2.5.6 A preconditioner for the Jacobian system

Multiple parts of the numerical continuation algortihm require solving linear systems of the form

$$\mathcal{F}_\psi(\psi, \mu)\phi = b \tag{2.24}$$

for a certain $b \in \mathbb{C}^n$. The solutions of these systems will be approximated by applying a Krylov method (see section 3.3). To accelerate these methods when applied to the Ginzburg-Landau equation, approximate inverses of the operator

$$\mathcal{R}(\psi, \mu) : \mathbb{C}^n \to \mathbb{C}^n : \phi \to \mathcal{K}^{(h)}\phi + 2 \begin{pmatrix} |\psi_1|^2\phi_1 \\ \vdots \\ |\psi_n|^2\phi_n \end{pmatrix} \tag{2.25}$$

are used as a preconditioner [93]. Approximate inversion is realized by an algebraic multigrid (AMG) strategy [93, 71]. The operator (2.25) is self-adjoint with respect to the inner product (2.21) and is positive semi-definite. It is strictly positive definite for $\mathbf{A} \neq 0$, which is the case for the applications considered in chapter 9. The approximate inverse of (2.25), derived with algebraic multigrid, is also self-adjoint with respect to (2.21) and positive definite for $\mathbf{A} \neq 0$ [93].

The use of the preconditioner together with a Krylov method yields an optimal solver for linear systems of the form (2.24): the amount of Krylov iterations required to converge is independent of the number of unknowns in the linear system [93]. Numerical experiments in [93] show that the approximate inversion of (2.25) with a single V-cycle yields the fastest solver.

CHAPTER 3

Review of numerical methods

"It's dangerous to go alone! Take this."

– Old Man –
The Legend of Zelda

Chapter highlights:

- We explain how a linear operator's eigenpairs are approximated by Ritz pairs, and give an algorithm based on explicit restarts.
- We discuss Krylov methods, used to approximate the solution to linear problems.
- We indicate the importance of deflation for problems with ill-conditioned linear operators.
- We discuss the nonlinear conjugate gradients method, used for small-scale nonlinear minimization problems.
- We review the Crank-Nicolson method, a time step scheme that will be used to validate stability results.
- The results in this chapter are mainly based on the following references: [7, 25, 45, 63, 97].

3.1 Introduction

The techniques used to create bifurcation diagrams rely on a lot of different numerical solvers. Especially the Newton-Krylov method with block elimination is essential when performing the pseudo-arclength continuation algorithm (see section 6.4). An alternative version of this method will play a key role in the approximation of bifurcation points (see section 6.6.2). Newton-Krylov methods rely on other solvers themselves, which will be discussed first. This chapter reviews numerical methods used in Newton-Krylov solvers, or in other algorithms throughout the thesis.

3.2 Eigenpair solvers

Solvers for the eigenvalues and -vectors of (Hermitian) Jacobian operators are discussed in this section. The eigenvalues have multiple uses, for example to determine the stability of a certain state (see section 6.8) and to indicate the proximity of bifurcations (see sections 6.5 and 6.6), which the convergence criterion used in algorithm 6.6 (page 198) is based on.

For certain bifurcation points, the approximate null vectors of the Jacobian operator are required in the construction of tangent directions (see section 7.2.2). To find these points themselves, linear systems with a Jacobian that is approximately singular need to be solved. A Krylov solver is used for this purpose, we use deflation techniques to ensure convergence. It is essential to provide the approximate null vectors for deflation to work (see section 3.3.2).

In this section we provide methods to calculate eigenpairs of a general, Hermitian, linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$ ($n \in \mathbb{N}$). The methods allow to provide already known (approximate) null vectors that should not be returned. This is for example useful when $\mathcal{A}$ is the Jacobian of a problem that is invariant under certain continuous symmetries (see example 3.1). The provided (approximate) null vectors are denoted by $\varphi_1, \ldots, \varphi_l$ (for a certain $l \in \mathbb{N}$).

Example 3.1. The discrete Ginzburg-Landau equation (2.20) described in section 2.5 is invariant under the actions of S^1 for any choice of domain Ω. For a solution (ψ_s, μ_s) of (2.20), this continuous symmetry induces a null vector $\mathrm{i}\psi_s$ for the Jacobian $\mathcal{F}_\psi(\psi_s, \mu_s)$ given by (2.22). ◇

The eigenvalues of $\mathcal{A}$ and their corresponding eigenvectors will be denoted by respectively λ_i and ϕ_i (for $i = 1, \ldots, n$). The eigenvalues are ordered according to absolute value:

$$|\lambda_1| \leq |\lambda_2| \leq |\lambda_3| \leq \cdots \leq |\lambda_n|.$$

In our applications we are mainly interested in eigenvalues close to zero.

When a (Hermitian, positive definite) preconditioner $\mathcal{P}$ is provided, eigenpairs of the preconditioned system are calculated:

$$\mathcal{P}\mathcal{A}\phi_i = \lambda_i \phi_i \qquad \text{for } i \in \{1, \ldots, n\}.$$

In this case vectors τ_i ($i \in \{1, \ldots, n\}$), defined by the formula

$$\forall i = 1, \ldots, n : \mathcal{P}\tau_i = \phi_i,$$

will be approximated and returned by the algorithms as well. Note that, if $\mathcal{P}$ is invertible, $\mathcal{A}\phi_i = \lambda_i \tau_i$ holds ($\forall i = 1, \ldots, n$). The unpreconditioned problem can be derived from the preconditioned one by setting $\mathcal{P} = \mathcal{I}$.

Note that the eigenpairs of the preconditioned linear operator $\mathcal{P}\mathcal{A}$ possibly differ from the ones of the original operator $\mathcal{A}$. Since only Hermitian, positive definite preconditioners are considered in the thesis (see chapter 2), the qualitative properties of the eigenvalues are the same for both operators (see proposition 3.2).

Proposition 3.2. Let $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$ and $\mathcal{P} : \mathbb{C}^n \to \mathbb{C}^n$ be Hermitian linear operators. Assume $\mathcal{P}$ is positive definite. Then each eigenvalue λ of $\mathcal{A}$ corresponds to an eigenvalue λ' of $\mathcal{PA}$ with the same sign and multiplicity.

Corollary 3.3. The kernels of the linear operators $\mathcal{A}$ and $\mathcal{PA}$ defined in proposition 3.2 have the same dimension.

Only the signs of eigenvalues, and whether they approximate zero, are used in the algorithms discussed in the thesis. By proposition 3.2 the eigenpairs of the preconditioned operator $\mathcal{PA}$ can be used in these algorithms without nullifying results on e.g. stability or bifurcation detection.

3.2.1 Approximation by Ritz pairs

We will approximate the eigenpairs of $\mathcal{A}$ by Ritz pairs [87]. To calculate Ritz pairs, first a base for an appropriate Krylov subspace needs to be constructed.

Definition 3.4 (Krylov subspace)**.** Given a linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$ and a vector $b \in \mathbb{C}^n$. The m-dimensional Krylov subspace ($m \leq n$) spanned by $\mathcal{A}$ and b is given by [63]

$$\mathcal{K}_m(\mathcal{A}, b) = \left\{ \sum_{i=0}^{m-1} a_i \mathcal{A}^i b \middle| a_0, \ldots, a_{m-1} \in \mathbb{R} \right\}.$$

To construct a base for the Krylov subspace spanned by $\mathcal{A}$ and a given vector b, the Arnoldi algorithm is applied (see algorithm 3.1 on page 42 in appendix 3.6, taken from Saad [86]). In absence of preconditioners, matrices V and H are constructed by this algorithm. The columns of V form an orthonormal base for $\mathcal{K}_m(\mathcal{A}, b)$, the matrix H is of upper Hessenberg form and represents the orthogonal projection of the operator $\mathcal{A}$ onto $\mathcal{K}_m(\mathcal{A}, b)$. It is possible to provide the inner product $\langle \cdot, \cdot \rangle$ used for orthonormality of the base vectors in algorithm 3.1. If the inner product is not specified, the Euclidean one is used. The algorithm allows for the deflation of vectors given as the columns of a deflation matrix K. When provided, the base vectors will be orthogonalized to the columns of K as well. After the construction of matrices H and V, the eigenpairs of $\mathcal{A}$ are approximated by Ritz pairs.

Definition 3.5 (Ritz pairs [87])**.** Let $\mathcal{A}$ be a $\mathbb{C}^n \times \mathbb{C}^n$ linear operator and $b \in \mathbb{C}^n$. The columns of $V \in \mathbb{C}^{n \times m}$ form a base for the m-dimensional Krylov subspace spanned by $\mathcal{A}$ and b, the upper Hessenberg matrix $H \in \mathbb{C}^{m \times m}$ represents the orthogonal projection of $\mathcal{A}$ onto this subspace.

Let $(\lambda_1, \beta_1), (\lambda_2, \beta_2), \ldots, (\lambda_m, \beta_m)$ be the eigenpairs of H. We call the values $\lambda_1, \lambda_2, \ldots, \lambda_m$ Ritz values of $\mathcal{A}$. The vectors $V\beta_1, V\beta_2, \ldots, V\beta_m$ are the corresponding Ritz vectors.

If the Krylov subspace is constructed with a general (random) vector b, the Ritz pairs approximate eigenpairs of $\mathcal{A}$. The precision of the approximation is determined by the dimension of the subspace, and can be improved by performing more Arnoldi iterations. Typically Ritz values converge to the extreme eigenvalues of $\mathcal{A}$ [87]. This is a valuable property, since we are mainly interested in eigenvalues close to zero.

When a self-adjoint, positive definite preconditioner $\mathcal{P}$ is provided, the Arnoldi algorithm (algorithm 3.1) constructs matrices V, P and H. The columns of V, respectively P, form an orthonormal base for $\mathcal{K}_m(\mathcal{PA}, \mathcal{P}b)$ and $\mathcal{K}_m(\mathcal{AP}, b)$ and satisfy the relation $\mathcal{P}P = V$. The upper Hessenberg matrix H again represents the orthogonal projection of the two linear operators onto their respective Krylov subspaces. This projected matrix is identical for both operators. The columns of V and P are orthonormal for adjusted inner products $\langle \cdot, \cdot \rangle_{\mathcal{P}^{-1}}$, respectively $\langle \cdot, \cdot \rangle_{\mathcal{P}}$, as defined in proposition 3.6. Note that the linear operator $\mathcal{P}^{-1}$ itself does not appear in algorithm 3.1.

Proposition 3.6. Let $\langle \cdot, \cdot \rangle$ be an inner product and $\mathcal{P}$ a linear operator that is self-adjoint and positive definite with respect to $\langle \cdot, \cdot \rangle$. Then the maps defined by

$$\langle \cdot, \cdot \rangle_{\mathcal{P}} : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R} : (v, w) \to \langle v, \mathcal{P}w \rangle,$$
$$\langle \cdot, \cdot \rangle_{\mathcal{P}^{-1}} : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R} : (v, w) \to \langle v, \mathcal{P}^{-1}w \rangle,$$

are inner products as well.

Proof. This follows from the definitions of positive definiteness and an inner product. □

The columns $V_1, \ldots, V_m$ and $P_1, \ldots, P_m$ of matrices V and P satisfy

$$\langle V_i, V_j \rangle_{\mathcal{P}^{-1}} = \langle P_i, V_j \rangle = \langle P_i, P_j \rangle_{\mathcal{P}} = \delta_{ij}.$$

Ritz pairs of $\mathcal{PA}$ are calculated in a similar way as for the unpreconditioned case. Remember that, when the preconditioner $\mathcal{P}$ is self-adjoint and positive definite, the eigenvalues of $\mathcal{A}$ and $\mathcal{PA}$ have the same qualitative properties (see proposition 3.2). The same holds for approximations by Ritz pairs.

A first algorithm for the approximation of eigenpairs of a linear operator $\mathcal{PA}$ (with $\mathcal{P}$ the preconditioner) is given by algorithm 3.2 on page 42 in appendix 3.6 (taken from Saad [86]). First the Arnoldi algorithm is applied with a random vector b, then k Ritz pairs are calculated. The unpreconditioned case is derived from algorithm 3.2 by setting $\mathcal{P} = \mathcal{I}$. The algorithm returns a diagonal matrix

$$L = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_k \end{pmatrix}$$

with the Ritz values on the diagonal, a matrix

$$W = \begin{pmatrix} \phi_1 & \phi_2 & \ldots & \phi_k \end{pmatrix}$$

with the corresponding Ritz vectors as columns, and a matrix

$$U = \begin{pmatrix} \tau_1 & \tau_2 & \ldots & \tau_k \end{pmatrix}$$

such that $\mathcal{P}U = W$. If necessary, already known (approximate) null vectors $\varphi_1, \varphi_2, \ldots, \varphi_l$ can be provided by setting the deflation matrix K as

$$K = \begin{pmatrix} \varphi_1 & \varphi_2 & \ldots & \varphi_l \end{pmatrix}.$$

3.2.2 Explicit restarts

Algorithm 3.2 (on page 42) did not contain any criteria to check the convergence of the calculated Ritz pairs. An indication for the precision of a Ritz pair can be derived from the Arnoldi factorization: if (λ, ϕ_H) is an eigenpair of the upper Hessenberg matrix H, created for an m-dimensional Krylov subspace, then the value $\gamma = |H_{m+1,m}\langle e_m, \phi_H\rangle|$ (with e_m the mth unit vector) indicates the proximity of the corresponding Ritz pair to the actual eigenpair of $\mathcal{A}$. The approximation becomes better for values of γ close to zero [86].

Typically Ritz values near the extreme eigenvalues of $\mathcal{A}$ converge fast, convergence to other pairs can be slow. If multiple eigenpairs need to be approximated, a lot of Arnoldi iterations are possibly required before the desired tolerance is reached. This can lead to memory issues. To counter this problem, we will create a new algorithm that allows for restarts. In this algorithm we allow multiple Arnoldi factorizations to be calculated, but each factorization should only require a low amount (m) of iterations.

Restarts are realized in two different ways [86]. If an Arnoldi factorization (H, V) for the Krylov subspace $\mathcal{K}_m(\mathcal{A}, b)$, constructed from a random vector b, does not yield (low magnitude) Ritz pairs converged up to the desired tolerance, a restart is made. A new Arnoldi factorization is started, this time for the subspace $\mathcal{K}_m(\mathcal{A}, \phi)$, with ϕ the current Ritz vector corresponding to the lowest magnitude Ritz value. Note that when a preconditioner is provided, the vector $\tau = \mathcal{P}^{-1}\phi$ is used for the restart. Due to this explicit choice of the vector used to span the Krylov subspace, the Ritz vector close to ϕ should converge faster [86]. The process can be repeated until the desired tolerance is reached.

A second kind of restart is performed after one or more Ritz pairs, calculated from a factorization (H, V), have been accepted. To prevent the algorithm from converging to these pairs again, they are included in the deflation matrix K. A new factorization is constructed with this updated deflation matrix, which will yield Ritz values that converge to the new extreme eigenvalues of $\mathcal{A}$, excluding the ones already approximated.

The updated algorithm, allowing for explicit restarts, is given by algorithm 3.3 on page 43 in appendix 3.6, taken from Saad [86]. Arnoldi factorizations for m-dimensional Krylov subspaces are performed until one or more low magnitude (magnitude below ϵ_2) Ritz pairs converge up to the desired tolerance (ϵ_1). A maximum of $n_{maxiter}$ restarts (of the first kind) are allowed for this part. Accepted Ritz pairs are stored, and another restart (of the second kind) is made, with an updated deflation matrix. This process is repeated until the wanted amount k of Ritz pairs is calculated, or until no converged pairs are found after $n_{maxiter}$ restarts (of the first kind). The algorithm again allows for a self-adjoint, positive definite preconditioner $\mathcal{P}$ to be provided, setting $\mathcal{P} = \mathcal{I}$ yields the unpreconditioned method.

3.2.3 Possible improvements and alternative methods

For the applications considered in the thesis, eigenpairs are efficiently approximated by algorithm 3.3. Though acceptable, the computational work required

for solving eigenvalue problems is often quite high due to a possible need of many restarts. Several improvements can however be made to the method.

A first improvement concerns the use of implicit restarts instead of explicit ones. In section 3.2.2 each restart constructs an entirely new Arnoldi factorization, using a new starting vector and possibly an updated deflation matrix, produced by the previous iteration(s). A more efficient approach, called implicit restarting, omits the requirement of multiple factorizations by combining the Arnoldi algorithm with shifted QR iterations [68]. After each implicit restart, part of the matrices V and H are removed. The shifts are chosen such that only eigenvalues of interest remain in the factorization. Computational requirements are strongly reduced compared to the method with explicit restarts.

Many eigenvalue problems need to be solved when numerical continuation is applied. Often the linear operator $\mathcal{A}$, for which eigenpairs are approximated, resembles an operator $\mathcal{A}'$ that appeared in a previous problem. This yields a second possible improvement to algorithm 3.3: due to the resemblance, the spectral information calculated for $\mathcal{A}'$ can be included in the problem for $\mathcal{A}$ to reduce computational work. For example, the initial vector used in the Arnoldi factorization can be chosen as a linear combination of eigenvectors of $\mathcal{A}'$, typically reducing the amount of Arnoldi steps required for convergence [89].

The proposed adjustments reduce the required computational work of algorithm 3.3. However, if the linear operator $\mathcal{A}$ contains an eigenvalue λ of high multiplicity (e.g. multiplicity 24, which might occur if $\mathcal{A}$ is the Jacobian of a nonlinear system with certain three-dimensional symmetries), the computational work possibly remains high. A normal Arnoldi factorization (without restarts) is only able to find a single eigenvector for each unique eigenvalue [85]. To find all eigenvectors with eigenvalue λ, each time an eigenvector is accepted, it needs to be deflated by an appropriate restart. The minimal amount of restarts that is required, to find all of the wanted eigenvectors, is equal to the multiplicity of the eigenvalue. Since restarting the Arnoldi factorization is computationally expensive, alternative methods for solving eigenvalue problems should be considered for such cases.

One such alternative method is the Jacobi-Davidson algorithm [99, 101, 100]. Similar to the Arnoldi factorization, the eigenvalue problem is projected onto a subspace, which is extended in each iteration. Instead of approximating the Ritz pair after a fixed amount of iterations, each Jacobi-Davidson step k constructs a guess $\tilde{\phi}^{(k)}$ for the wanted eigenvector. Contrary to the Arnoldi factorization, where the subspace is extended by application of $\mathcal{A}$ to one of its vectors, the subspace is extended by solving a projected linear system for an update vector $d\phi$. This update vector approximates the orthogonal correction for $\tilde{\phi}^{(k-1)}$. The subspace is extended by $d\phi$ (after orthogonalization), and a new guess $\tilde{\phi}^{(k)}$ is calculated similar to definition 3.5.

To reduce computational work and storage costs, restarting and deflation strategies can also be incorporated in the Jacobi-Davidson method. When the linear operator $\mathcal{A}$ contains an eigenvalue λ of high multiplicity, the method should be slightly adapted such that all eigenvectors with this eigenvalue are approximated simultaneously [8]. The resulting method is typically more efficient than using an Arnoldi factorization for these kinds of problems.

Though computational work can be reduced and alternative methods are more efficient in certain cases, algorithm 3.3 will be used for approximating eigenpairs in the remainder of the thesis. Wanted eigenvalues never had a multiplicity of more than 2, and were computed in an acceptable time for the applications considered in chapter 2.

3.3 Krylov methods

In multiple applications a linear system of the form

$$\mathcal{F}_\psi(\psi, \mu)x = b \tag{3.1}$$

needs to be solved, for a certain right-hand side $b \in \mathbb{C}^n$ ($n \in \mathbb{N}$) and $\mathcal{F}_\psi(\psi, \mu) : \mathbb{C}^n \to \mathbb{C}^n$ the partial Jacobian of a certain function evaluated at (ψ, μ). These Jacobian systems appear naturally when executing the Newton algorithm (see chapters 4 and 5), but they also appear in certain algorithms for the construction of tangent directions (see chapter 7).

We will not solve systems of the form (3.1) exactly. Often the size of the Jacobian is too big for this to be practical, for some problems (e.g. the Ginzburg-Landau equation, see section 2.5) it is even impossible due to the Jacobian only being given in the form of a linear operator. Instead, a Krylov method will be used to approximate the solution x of the linear system. These methods make full use of the sparsity structure of operators, since only their application to given vectors needs to be calculated [63]. The problems we consider (see chapter 2) either have a small-scale or a sparse (Hermitian) Jacobian.

As in section 3.2, we will consider a general, Hermitian, $\mathbb{C}^n \times \mathbb{C}^n$ linear operator $\mathcal{A}$, not necessarily derived from a Jacobian. We will describe algorithms that solve a linear system

$$\mathcal{A}x = b \tag{3.2}$$

for $x \in \mathbb{C}^n$, with $b \in \mathbb{C}^n$. It is possible to provide a preconditioner $\mathcal{P}$ to speed up the algorithm. In this case the solution of

$$\mathcal{P}\mathcal{A}x = \mathcal{P}b$$

is searched. We will only consider Hermitian, positive definite preconditioners. We assume that any (approximate) null vectors of $\mathcal{A}$ are known (possibly after application of algorithm 3.3) and denote these by $\varphi_1, \ldots, \varphi_l$ ($l \in \mathbb{N}$). If a preconditioner is provided, these vectors are of the form

$$\mathcal{P}\mathcal{A}\mathcal{P}\varphi_i = \lambda_i \mathcal{P}\varphi_i \quad \text{for } i = 1, \ldots, l$$

with $\lambda_i \approx 0$ ($\forall i = 1, \ldots, l$).

3.3.1 The GMRES algorithm

In Krylov methods an approximation to the solution is searched within a subspace $x^{(0)} + \mathcal{K}_m(\mathcal{A}, r^{(0)})$ ($m \leq n$), for a certain initial guess $x^{(0)}$, with initial

residual $r^{(0)} = \mathcal{A}x^{(0)} - b$. In the GMRES method, the approximation is chosen such that the residual norm is minimal [63, 88]. We will approximate x by $\tilde{x}$, given by

$$\tilde{x} \quad = \underset{x \in x^{(0)} + \mathcal{K}_m(\mathcal{A}, r^{(0)})}{\arg\min} \|\mathcal{A}x - b\|.$$

In order to calculate $\tilde{x}$ with GMRES the Arnoldi procedure (see section 3.2.1) is used to construct an orthonormal base V for the Krylov subspace $\mathcal{K}_m(\mathcal{A}, r^{(0)})$, and an upper Hessenberg matrix H that represents the orthogonal projection of the operator $\mathcal{A}$ onto this subspace. Next, the required linear combination $y \in \mathbb{R}^m$ such that $Vy = \tilde{x}$ is calculated. This is done by minimizing the Euclidean norm

$$\|\|r^{(0)}\|e_1 - Hy\|_2 \tag{3.3}$$

over $y \in \mathbb{R}^m$, using Givens rotations [63]. The upper Hessenberg form of H allows this minimization problem to be solved efficiently. The approximation $\tilde{x}$ is then calculated as $\tilde{x} = x^{(0)} + Vy$.

GMRES remains unchanged when a preconditioner $\mathcal{P}$ is provided, except for the construction of the matrices V and H. With preconditioning, bases V and P for the Krylov subspaces $\mathcal{K}_m(\mathcal{P}\mathcal{A}, \mathcal{P}r^{(0)})$ and $\mathcal{K}_m(\mathcal{A}\mathcal{P}, r^{(0)})$ are constructed. These bases are orthonormal with respect to the inner products $\langle \cdot, \cdot \rangle_{\mathcal{P}^{-1}}$, respectively $\langle \cdot, \cdot \rangle_{\mathcal{P}}$. The remainder of the method remains unchanged.

We will not describe the GMRES method by an explicit algorithm. Instead, we note that the standard GMRES method is a special case of algorithm 3.4 (see page 44 in appendix 3.6), when the deflation matrix K is set as an empty $n \times 0$ matrix.

If the operator $\mathcal{A}$ is well-conditioned, the standard GMRES method generally works fine and typically exhibits superlinear convergence [108]. This is not necessarily the case when $\mathcal{A}$ is ill-conditioned, the (approximate) null vectors of $\mathcal{A}$ possibly hamper the convergence [63]. To prevent their influence, these vectors should be deflated. This is the topic of the next section.

3.3.2 Deflated GMRES

If the linear operator $\mathcal{A}$ is ill-conditioned, GMRES might yield an approximation far from the actual solution. To illustrate this, consider a linear operator $\mathcal{A}$ with an eigenpair (λ, φ) such that $|\lambda| \ll 1$ and $\|\varphi\| = 1$, the eigenvector φ is an approximate null vector:

$$\mathcal{A}\varphi = \lambda\varphi \approx 0. \tag{3.4}$$

The operator $\mathcal{A}$ is ill-conditioned. Consider a right-hand side $b \in \mathbb{C}^n$ of the form

$$b = \hat{b} + \epsilon\varphi \tag{3.5}$$

with $\langle \hat{b}, \varphi \rangle = 0$ and $\epsilon \in \mathbb{R}_0$ small, such that $|\lambda| \lesssim |\epsilon|$. The $\epsilon\varphi$ part of b is possibly caused by (rounding) errors, and should ideally only have a small influence on the approximation $\tilde{x}$.

Assume we perform a small amount (m) of GMRES iterations. After m steps the Arnoldi procedure yields matrices H, V, with V an m-dimensional base for the Krylov subspace $\mathcal{K}_m(\mathcal{A}, b)$. Since Ritz pairs tend to converge to the extreme eigenpairs of $\mathcal{A}$ first, a linear combination $z \in \mathbb{R}^m$ of V that approximates φ will appear after only a few iterations. Indeed, if $(\tilde{\lambda}, z)$ is the eigenpair of H such that $\tilde{\lambda} \approx \lambda$, then the vector Vz will approximate φ.

Let $\hat{x}$ be the solution of

$$\mathcal{A}\hat{x} = \hat{b}.$$

Since $\langle \hat{b}, \varphi \rangle = 0$, we have $\langle \hat{x}, \varphi \rangle = 0$ as well. The full solution x of the system

$$\mathcal{A}x = b = \hat{b} + \epsilon\varphi$$

becomes:

$$x = \hat{x} + \frac{\epsilon}{\lambda}\varphi.$$

Though the right-hand side b only contained a small part in the φ direction ($|\langle b, \varphi \rangle| = |\epsilon| \approx 0$), for the solution of the linear system we get

$$|\langle x, \varphi \rangle| = \frac{|\epsilon|}{|\lambda|} \gtrsim 1.$$

The approximate null vector φ has a big influence on the solution of the system, even though its part in the vector b is negligible. Since φ is approximately contained in the base V, the φ part of the solution x will be approximated well by $\tilde{x}$, the approximation found by the GMRES algorithm. This property leads to problems for e.g. the Newton method (see example 4.3 in section 4.2), and should be eliminated: we do not want a small perturbation of the right-hand side b in the φ direction to have a big influence on the approximation $\tilde{x}$.

Note that in above illustration, we specifically chose the right-hand side b to contain a part in the direction of the approximate null vector φ. However, even for b orthogonal to this vector the GMRES algorithm might still blow up the solution in the φ direction. This can be caused by rounding errors within the algorithm, or due to the error made when approximating φ itself. This is illustrated by example 3.7.

Example 3.7. Consider the discrete Ginzburg-Landau equation (see section 2.5), applied to a material shaped as a pentagon of side length 2.35, with magnetic field strength $\mu = 1.06$. $n = 10401$ discretization points are used. The linear operator $\mathcal{A}$ is chosen as the Jacobian (2.22) evaluated in a point $\tilde{\psi}$ near the solution of the system $\mathcal{F}(\psi, \mu) = 0$ (with $\mathcal{F}$ given by (2.20)). For this choice the linear operator $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, 1.06)$ contains three approximate null vectors. Approximations for these vectors are available by application of algorithm 3.3, these are denoted by $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$.

The right-hand side b is constructed by applying the linear operator $\mathcal{A}$ to a random vector $\hat{x} \in \mathbb{C}^n$, with $\hat{x}$ orthogonal to the approximate null vectors $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$. By construction, b is orthogonal to these vectors as well. The exact solution of the linear system $\mathcal{A}x = b$ is given by the vector $\hat{x}$.

We apply preconditioned GMRES (algorithm 3.4, page 44) to the linear system $\mathcal{A}x = b$. The preconditioner is given by (2.25), and the inner product

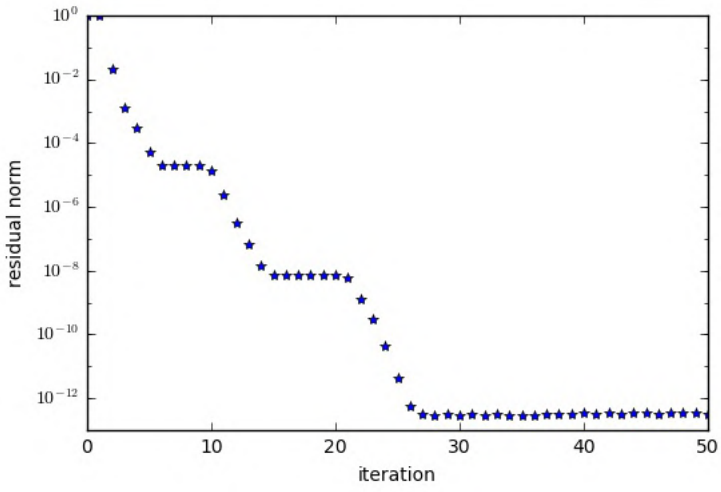


Figure 3.1: Residual plot of example 3.7. The GMRES method (without deflation) is applied to an ill-conditioned linear problem. The residual norm converges to approximately 10^{-12} after 27 iterations, which is the attainable accuracy.

(2.21) is used. $m = 50$ iterations are allowed and the tolerance ϵ is set to 0 to better investigate the behaviour of the algorithm. As an initial guess $x^{(0)}$ a random perturbation of the solution $\hat{x}$ is used, with $x^{(0)}$ still orthogonal to $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$. Deflation of these vectors is not used.

The residual plot is given by figure 3.1, a plot with the error norms (calculated as $\|\hat{x} - x^{(k)}\|$ for each iteration k) by figure 3.2. Though the residual norm decreases fast (convergence up to a tolerance of 10^{-12} is reached in less than 30 iterations), the error norms do not. After 40 iterations, the guess $x^{(k)}$ diverges from the solution $\hat{x}$. This is due to parts in $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$ being contained in $x^{(k)}$, though the actual solution $\hat{x}$ is orthogonal to these vectors. The residual norm does however not increase because $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$ are approximate null vectors.

$\diamond$

Solving linear systems with an ill-conditioned operator is required for multiple applications in the thesis, for example when the Newton method is executed near a bifurcation point (see section 6.6.2). Even far away from bifurcations possible continuous symmetries induce null vectors in the Jacobian (see e.g. the Ginzburg-Landau equation in section 2.5), leading to ill-conditioned systems in the Newton method.

To reduce the influence of (approximate) null vectors $\varphi_1, \ldots, \varphi_l$ in the GMRES algorithm, we introduce deflation. Given the deflation matrix

$$K = \begin{pmatrix} \varphi_1 & \varphi_2 & \ldots & \varphi_l \end{pmatrix},$$

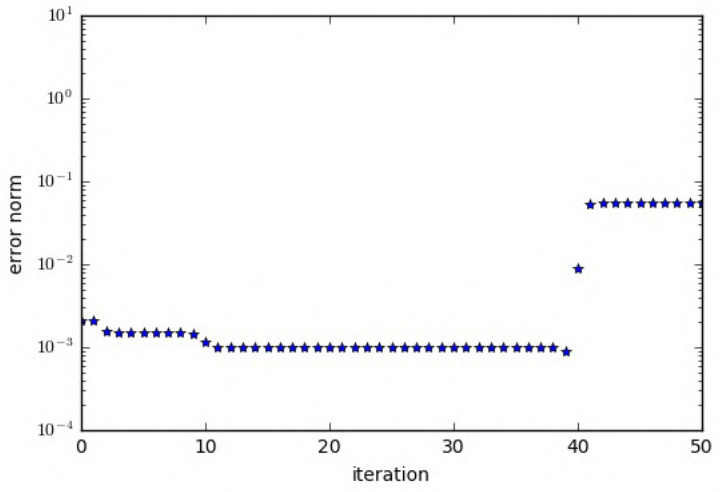


Figure 3.2: Error norms of example 3.7. The GMRES method (without deflation) is applied to an ill-conditioned linear problem. After an initial decrease of the error norm to approximately 10^{-3}, at iteration 40 it suddenly increases to 10^{-2} and then 10^{-1}. Note that the residual norm converged for this example (see figure 3.1).

the projection operator

$$\mathcal{Q} : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, K\rangle^{-1}\langle K, y\rangle$$

is defined. This operator orthogonalizes a vector $y \in \mathbb{C}^n$ to the approximate null vectors $\varphi_1, \ldots, \varphi_l$. Instead of solving the linear system (3.2), we search an approximate solution of the system [16, 45, 46]

$$\mathcal{Q}\mathcal{A}x = \mathcal{Q}b. \tag{3.6}$$

The base vectors V_i for the Krylov subspace are orthogonalized to the approximate null vectors $\varphi_1, \ldots, \varphi_l$ in the Arnoldi procedure. The approximate solution $\tilde{x}$ - which is constructed as a linear combination of the base vectors - will be orthogonal to $\varphi_1, \ldots, \varphi_l$ as well. Any influence of null vectors is eliminated from this approximation.

If a preconditioner $\mathcal{P}$ is provided, the projection operator

$$\mathcal{Q} : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, L\rangle^{-1}\langle L, y\rangle$$

is used, with $L = \mathcal{P}K$. The considered linear system becomes

$$\mathcal{P}\mathcal{Q}\mathcal{A}x = \mathcal{P}\mathcal{Q}b. \tag{3.7}$$

The deflated GMRES method is given by algorithm 3.4 on page 44 in appendix 3.6 (based on [45]). Each iteration performs one Arnoldi cycle, the constructed base vector is orthogonalized to the vectors provided by the deflation

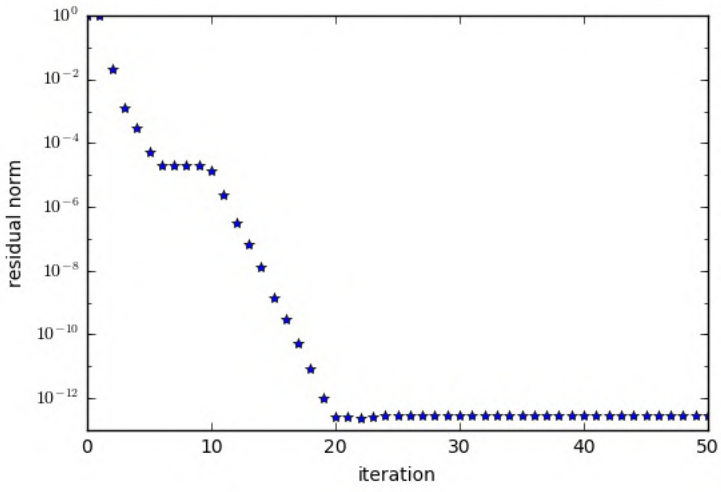


Figure 3.3: Residual plot of example 3.8. The deflated GMRES method is applied to an ill-conditioned linear problem. The residual norm converges to approximately 10^{-12} after 20 iterations, which is the attainable accuracy.

matrix K by applying the projection operator $\mathcal{Q}$. After the Arnoldi cycle the vector y that minimizes (3.3) is calculated by Givens rotations, and the corresponding approximation $\tilde{x}$ is created. The algorithm stops if the residual norm lies below a provided threshold ϵ. If this does not happen after a maximum of m iterations, $\tilde{x}$ is accepted regardless of this norm.

Note that the standard GMRES algorithm (without deflation) is derived from algorithm 3.4 by setting K as an empty $n \times 0$ matrix. In this case the projection operator $\mathcal{Q}$ becomes the identity operator $\mathcal{I}$.

When the deflated GMRES algorithm is applied to the illustration at the start of this section (see (3.4) and (3.5)), the approximation $\tilde{x}$ will approximate the solution $\hat{x}$ of

$$\mathcal{A}\hat{x} = \hat{b}.$$

If the $\epsilon\varphi$ part of b was caused by (rounding) errors, a good approximation for x will be found. This is illustrated by example 3.8.

Example 3.8. Consider the same setting as example 3.7. We again apply algorithm 3.4 to the linear system $\mathcal{A}x = b$, this time deflation is used. The deflation matrix is given by $K = \begin{pmatrix} \tilde{\varphi}_1 & \tilde{\varphi}_2 & \tilde{\varphi}_3 \end{pmatrix}$.

Figure 3.3 shows the residual plot of the example, figure 3.4 a plot of the error norms. The residual norm decreases faster than before (20 iterations are required to converge up to a tolerance of 10^{-12}, instead of 30). The error norms show that the guess $x^{(k)}$ does not diverge from the actual solution $\hat{x}$ when deflation is applied. A guess orthogonal to $\tilde{\varphi}_1$, $\tilde{\varphi}_2$ and $\tilde{\varphi}_3$ is found. $\diamond$

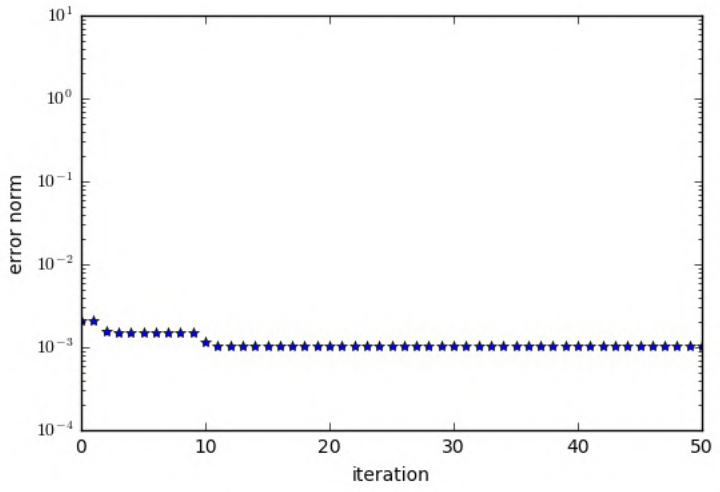


Figure 3.4: Error norms of example 3.8. The deflated GMRES method is applied to an ill-conditioned linear problem. Contrary to the method without deflation (see figure 3.2), the error norm does not show sudden increases. It converges to approximately 10^{-3} after 10 iterations. The residual norm converged for this example as well (see figure 3.3).

Note that if $\epsilon\varphi$ was not caused by (rounding) errors, the approximation found with deflated GMRES might contain a big error in the null vector part. The method only approximates the $\hat{x}$ part of the solution. To approximate the null vector parts of x different methods than GMRES should be used, depending on the underlying problem.

3.4 Nonlinear conjugate gradients

In some of the Newton-Krylov methods (see chapters 4 and 5) a nonlinear function $g : \mathbb{R}^l \to \mathbb{R}$, with $l \in \mathbb{N}$ small (for example $l = 3$), needs to be minimized. We use the nonlinear conjugate gradients method (NCG) for this problem [81]. NCG is an iterative algorithm that uses the gradient g' and Hessian g'' of g, given by

$$g' : \mathbb{R}^l \to \mathbb{R}^l : a \to \begin{pmatrix} \frac{\partial g(a)}{\partial a_1} \\ \frac{\partial g(a)}{\partial a_2} \\ \vdots \\ \frac{\partial g(a)}{\partial a_l} \end{pmatrix},$$

$$g'' : \mathbb{R}^l \to \mathbb{R}^{l\times l} : a \to \begin{pmatrix} \frac{\partial^2 g(a)}{\partial a_1^2} & \frac{\partial^2 g(a)}{\partial a_1 \partial a_2} & \cdots & \frac{\partial^2 g(a)}{\partial a_1 \partial a_l} \\ \frac{\partial^2 g(a)}{\partial a_2 \partial a_1} & \frac{\partial^2 g(a)}{\partial a_2^2} & \cdots & \frac{\partial^2 g(a)}{\partial a_2 \partial a_l} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g(a)}{\partial a_l \partial a_1} & \frac{\partial^2 g(a)}{\partial a_l \partial a_2} & \cdots & \frac{\partial^2 g(a)}{\partial a_l^2} \end{pmatrix},$$

to construct descent directions $d^{(i)}$. In each iteration i the vector $\hat{d}$ is calculated by

$$\hat{d} = -g''(a^{(i)})^{-1} g'(a^{(i)}),$$

with $a^{(i)}$ the current guess. This vector is then used either to update $d^{(i+1)}$ (using the Fletcher-Reeves formula [81]), or as $d^{(i+1)}$ itself. A new guess is then constructed by updating the old one in such a descent direction, where line search is used to find the approximate minimum.

An implementation of the nonlinear conjugate gradients method is given by algorithm 3.5 on page 45 in appendix 3.6 (taken from Shewchuck [97]). An initial guess $a^{(0)}$ is updated over a maximum of m_{out} iterations. The algorithm stops if the relative difference in residual is sufficiently low (lower than a tolerance ϵ_{out}). In each iteration a line search is performed to find the minimum of g along a single direction. Line search steps are performed until the relative update lies below a certain tolerance ϵ_{in}, or until a maximum of m_{in} steps.

If the gradient function g' is not provided, the algorithm approximates it by the first-order central finite difference scheme [43]:

$$\begin{aligned} &g' : \mathbb{R}^l \to \mathbb{R}^l : a \to g'(a) \text{ such that } \forall i = 1, \ldots, l : \\ &g'(a)_i = \frac{1}{2\varepsilon} \left(g(a + \varepsilon e_i) - g(a - \varepsilon e_i) \right) \end{aligned} \tag{3.8}$$

with $\varepsilon \in \mathbb{R}_0^+$ a small real number (typically chosen as $\varepsilon = \sqrt[3]{\varepsilon_{mach}}$, with ε_{mach} the machine precision [63]), and e_i the ith unit vector. The same is done for the Hessian function g'' if it is not provided, in this case we approximate it with the second-order central scheme [43] (typically choosing $\varepsilon = \sqrt[4]{\varepsilon_{mach}}$ [63]):

$$\begin{aligned} &g'' : \mathbb{R}^l \to \mathbb{R}^{l\times l} : a \to g''(a) \text{ such that } \forall i, j = 1, \ldots, l : \\ &g''(a)_{ij} = \begin{cases} \frac{1}{\varepsilon^2} \left(g(a + \varepsilon e_i) - 2g(a) + g(a - \varepsilon e_i) \right) & \text{if } i = j \\ \frac{1}{4\varepsilon^2} \left(g(a + \varepsilon e_i + \varepsilon e_j) - g(a + \varepsilon e_i - \varepsilon e_j) \right. & \\ \qquad \left. - g(a - \varepsilon e_i + \varepsilon e_j) + g(a - \varepsilon e_i - \varepsilon e_j) \right) & \text{if } i \neq j \end{cases} \end{aligned} \tag{3.9}$$

Though the nonlinear conjugate gradients method works well for general functions g, it is only efficient if the number of variables l is small. For a higher amount of variables the cost of calculating the elements in the gradient $g'(a^{(i)})$ and Hessian $g''(a^{(i)})$ for each iteration would become too high. When used in the Newton-Krylov methods discussed in chapters 4 and 5, l will approximate the amount of null vectors in the Jacobian of the function a zero is searched for. This is generally a small number.

3.5 The Crank-Nicolson method

We end the chapter by a review of the Crank Nicolson scheme, a time step method [25]. Consider a partial differential equation of the form

$$\frac{\partial \Psi(t)}{\partial t} = F(\Psi(t)), \tag{3.10}$$

with a Hermitian partial Jacobian $F_\Psi(\Psi(t))$. The state $\Psi : \mathbb{R} \to \mathbb{C}^n$ is a function of the time t. Given $l \in \mathbb{N}$, a time step Δt and initial time t_0, the discretization

$$\forall j = 0, \ldots, l : t_j = t_0 + j\Delta t$$

is considered. Denoting $\Psi(t_j)$ as $\Psi^{(j)}$ $(\forall j = 0, \ldots, l)$, the Crank-Nicolson method calculates $\Psi^{(1)}, \ldots, \Psi^{(l)}$ from a given initial state $\Psi^{(0)}$ by iteratively solving the equation

$$\frac{1}{\Delta t}\left(\Psi^{(j+1)} - \Psi^{(j)}\right) = \frac{1}{2}\left(F(\Psi^{(j+1)}) + F(\Psi^{(j)})\right) \tag{3.11}$$

for $j = 0, \ldots, l-1$. Pseudo-code for the method is given by algorithm 3.6 on page 47 in appendix 3.6.

In practice solving (3.11) is performed by a Newton algorithm (see algorithm 4.1 on page 100, described in section 4.2). The function to solve with Newton is given by $(\forall j = 0, \ldots, l)$

$$G(\Psi^{(j+1)}) = -\Psi^{(j+1)} + \Psi^{(j)} + \frac{\Delta t}{2}\left(F(\Psi^{(j+1)}) + F(\Psi^{(j)})\right) = 0. \tag{3.12}$$

Note that the Newton method uses GMRES to solve its underlying Jacobian system, it is possible to provide an inner product $(\langle \cdot, \cdot \rangle)$ and (Hermitian, positive definite) preconditioner (P) for this purpose.

The Crank-Nicolson method yields evaluations of the unknown function Ψ for the given discrete points in time. It is used to generate an evolution in time of a given initial state $\Psi^{(0)}$, by using the corresponding partial differential equation (3.10).

Note that it is possible to find steady states by applying the Crank-Nicolson method: $\Psi^{(l)} = \Psi(t_0 + l\Delta t)$ should converge to a steady state when the amount of time steps l is increased. Only states that are physically stable (see section 6.8) can be found by this method however. Since we are interested in both physically stable and unstable steady states for our applications, we will not use the Crank-Nicolson method for this purpose. It will only be used to verify physical stability results (e.g. in section 9.5.1).

3.6 Appendix

The Arnoldi algorithm

Algorithm 3.1 Arnoldi

Input $m_{eig} \in \mathbb{N}$, Linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$, vector $b \in \mathbb{C}^n$, Preconditioner $\mathcal{P} : \mathbb{C}^n \to \mathbb{C}^n$, inner product $\langle \cdot, \cdot \rangle$, deflation matrix $K \in \mathbb{C}^{n \times l}$

Output Matrices $H \in \mathbb{C}^{m_{eig} \times m_{eig}}$, $V \in \mathbb{C}^{n \times m_{eig}}$ and $P \in \mathbb{C}^{n \times m_{eig}}$

Set $\mathcal{P} = \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
Set K an empty $n \times 0$ matrix if not specified
$L = \mathcal{P}K$
Define $\mathcal{Q} : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, L \rangle^{-1} \langle L, y \rangle$
Initialize $H \in \mathbb{C}^{m_{eig}+1 \times m_{eig}}$, $V \in \mathbb{C}^{n \times m_{eig}+1}$ and $P \in \mathbb{C}^{n \times m_{eig}+1}$
$w = \mathcal{Q}b$
$v = \mathcal{P}w$
$\alpha = \sqrt{\langle w, v \rangle}$
$P_1 = \alpha^{-1} w$
$V_1 = \alpha^{-1} v$
for $j = 2, \ldots, m_{eig} + 1$ **do**
    $w = \mathcal{Q}\mathcal{A}V_{j-1}$
    for $l = 1, \ldots, j - 1$ **do**
        $H_{l,j-1} = \langle V_l, w \rangle$
        $w \leftarrow w - H_{l,j-1} P_l$
    end for
    $v = \mathcal{P}w$
    $H_{j,j-1} = \sqrt{\langle v, w \rangle}$
    $P_j = H_{j,j-1}^{-1} w$
    $V_j = H_{j,j-1}^{-1} v$
end for
$\alpha = H_{m_{eig}+1, m_{eig}}$
$P \leftarrow P_{1:m_{eig}}$
$V \leftarrow V_{1:m_{eig}}$
$H \leftarrow H_{1:m_{eig}, 1:m_{eig}}$
Return H, V, P, α

Calculation of Ritz pairs

Algorithm 3.2 Ritz

Input $k_{eig} \in \mathbb{N}$, Linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$, Preconditioner $\mathcal{P} : \mathbb{C}^n \to \mathbb{C}^n$, inner product $\langle \cdot, \cdot \rangle$, deflation matrix $K \in \mathbb{C}^{n \times l}$

Output Ritz pairs represented by matrices L, W, U such that $\mathcal{P}U = W$ and $\mathcal{P}\mathcal{A}W = WL$

Set $\mathcal{P} = \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified

3: Set K an empty $n \times 0$ matrix if not specified
4: Initialize a random vector $b \in \mathbb{C}^n$
5: Calculate (H, V, P, α) by executing Arnoldi (algorithm 3.1) with given $\mathcal{A}$, b, $\mathcal{P}$, $\langle \cdot, \cdot \rangle$ and K
6: Calculate k_{eig} eigenpairs L, W_H of H exactly, ordered by increasing absolute eigenvalue
7: $W = VW_H$
8: $U = PW_H$
9: Return L, W, U

Calculation of Ritz pairs with explicit restarts

Algorithm 3.3 RitzRestart

Input $k_{eig}, n_{eig} \in \mathbb{N}$, tolerances $\epsilon_{eig1}, \epsilon_{eig2} \in \mathbb{R}_0^+$, Linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$, Preconditioner $\mathcal{P} : \mathbb{C}^n \to \mathbb{C}^n$, inner product $\langle \cdot, \cdot \rangle$, deflation matrix $K \in \mathbb{C}^{n \times l}$

Output Ritz pairs represented by matrices L, W, U such that $\mathcal{P}U = W$ and $\mathcal{P}\mathcal{A}W = WL$

1: Set $\mathcal{P} = \mathcal{I}$ if not specified
2: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
3: Set K an empty $n \times 0$ matrix if not specified
4: Initialize L an empty 0×0 matrix
5: Initialize U an empty $n \times 0$ matrix
6: Initialize W an empty $n \times 0$ matrix
7: $a = 0$
8: $stop = False$
9: **while** $a < k_{eig}$ and not $stop$ **do**
10: Initialize a random vector $b \in \mathbb{C}^n$
11: $i = 0$
12: $conv = False$
13: **while** $i < n_{eig}$ and not $conv$ **do**
14: $i \leftarrow i + 1$
15: Calculate (H, V, P, α) by executing Arnoldi (algorithm 3.1) with given $\mathcal{A}$, b, $\mathcal{P}$, $\langle \cdot, \cdot \rangle$ and K
16: Calculate $k_{eig} - a$ eigenpairs L_H, W_H of H exactly, ordered by increasing absolute eigenvalue
17: $j = 0$
18: **while** $j < k_{eig} - a$ and $a < k_{eig}$ **do**
19: $j \leftarrow j + 1$
20: $\gamma = |\alpha \langle e_m, W_{Hj} \rangle|$
21: **if** $\gamma \leq \epsilon_{eig1}$ and $|L_{Hjj}| \leq \epsilon_{eig2}$ **then**
22: $conv \leftarrow True$
23: $a \leftarrow a + 1$
24: $\tau = PW_{Hj}$

25: $L \leftarrow \begin{pmatrix} L & 0 \\ 0 & L_{Hjj} \end{pmatrix}$
26: $W \leftarrow \begin{pmatrix} W & VW_{Hj} \end{pmatrix}$
27: $U \leftarrow \begin{pmatrix} U & \tau \end{pmatrix}$
28: $K \leftarrow \begin{pmatrix} K & \tau \end{pmatrix}$
29: **end if**
30: **end while**
31: **if** not *conv* **then**
32: $\tau = PW_{H1}$
33: $b \leftarrow \tau$
34: **end if**
35: **end while**
36: **if** not *conv* **then**
37: $stop \leftarrow True$
38: **end if**
39: **end while**
40: Return L, W, U

GMRES

Algorithm 3.4 GMRES

Input $m_{lin} \in \mathbb{N}$, tolerance $\epsilon_{lin} \in \mathbb{R}_0^+$, Linear operator $\mathcal{A} : \mathbb{C}^n \to \mathbb{C}^n$, vector $b \in \mathbb{C}^n$, Preconditioner $\mathcal{P} : \mathbb{C}^n \to \mathbb{C}^n$, inner product $\langle \cdot, \cdot \rangle$, deflation matrix $K \in \mathbb{C}^{n \times l}$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for the system $\mathcal{P}\mathcal{A}x = \mathcal{P}b$

1: Set $\mathcal{P} = \mathcal{I}$ if not specified
2: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
3: Set K an empty $n \times 0$ matrix if not specified
4: Set $x^{(0)} = 0$ if not specified
5: $L = \mathcal{P}K$
6: Define $\mathcal{Q} : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, L \rangle^{-1} \langle L, y \rangle$
7: Initialize $H \in \mathbb{C}^{m_{lin}+1 \times m_{lin}}$, $V \in \mathbb{C}^{n \times m_{lin}+1}$ and $P \in \mathbb{C}^{n \times m_{lin}+1}$
8: Initialize $R \in \mathbb{R}^{m_{lin}+1 \times m_{lin}}$, $S \in \mathbb{R}^{m_{lin}}$, $C \in \mathbb{R}^{m_{lin}}$ and $\tilde{b} \in \mathbb{R}^{m_{lin}+1}$
9: $w = Q(b - \mathcal{A}x^{(0)})$
10: $v = \mathcal{P}w$
11: $\alpha = \sqrt{\langle w, v \rangle}$
12: $P_1 = \alpha^{-1} w$
13: $V_1 = \alpha^{-1} v$
14: $\tilde{b}_1 = \alpha$
15: $j = 1$
16: $stop = False$
17: **while** $j < m_{lin} + 1$ and not *stop* **do**
18: $j \leftarrow j + 1$
19: $w = \mathcal{Q}\mathcal{A}V_{j-1}$

20: **for** $i = 1, \dots, j-1$ **do**
21: $H_{i,j-1} = \langle V_i, w \rangle$
22: $w \leftarrow w - H_{i,j-1} P_i$
23: **end for**
24: $v = \mathcal{P} w$
25: $H_{j,j-1} = \sqrt{\langle v, w \rangle}$
26: $P_j = H_{j,j-1}^{-1} w$
27: $V_j = H_{j,j-1}^{-1} v$
28: $R_{1,j-1} = H_{1,j-1}$
29: **for** $i = 1, \dots, j-2$ **do**
30: $\gamma = C_i R_{i,j-1} + S_i H_{i+1,j-1}$
31: $R_{i+1,,j-1} = -S_i R_{i,j-1} + C_i H_{i+1,j-1}$
32: $R_{i,j-1} \leftarrow \gamma$
33: **end for**
34: $\delta = \sqrt{R_{j-1,j-1}^2 + H_{j,j-1}^2}$
35: $C_{j-1} = \delta^{-1} R_{j-1,j-1}$
36: $S_{j-1} = \delta^{-1} H_{j,j-1}$
37: $R_{j-1,j-1} \leftarrow C_{j-1} R_{j-1,j-1} + S_{j-1} H_{j,j-1}$
38: $\tilde{b}_j = -S_{j-1} \tilde{b}_{j-1}$
39: $\tilde{b}_{j-1} \leftarrow C_{j-1} \tilde{b}_{j-1}$
40: Initialize $y \in \mathbb{R}^{j-1}$
41: **for** $i = 1, \dots, j-1$ **do**
42: $y_{j-i} = R_{j-i,j-i}^{-1} \left(\tilde{b}_{j-i} - \sum_{p=1}^{i-1} R_{j-i,j-i+p} y_{j-i+p} \right)$
43: **end for**
44: $\tilde{x} = x^{(0)} + \sum_{i=1}^{j-1} V_i y_i$
45: $q = \mathcal{Q} b - \mathcal{Q} \mathcal{A} \tilde{x}$
46: $r = \mathcal{P} q$
47: $\rho = \sqrt{\langle q, r \rangle}$
48: **if** $\alpha^{-1} \rho < \epsilon_{lin}$ **then**
49: $stop \leftarrow True$
50: **end if**
51: **end while**
52: Return $\tilde{x}$

Nonlinear conjugate gradients (NCG)

Algorithm 3.5 NCG

Input $m_{NCG1}, m_{NCG2} \in \mathbb{N}$, tolerances $\epsilon_{NCG1}, \epsilon_{NCG2} \in \mathbb{R}_0^+$, functions $g : \mathbb{R}^l \to \mathbb{R}$, $g' : \mathbb{R}^l \to \mathbb{R}^l$ and $g'' : \mathbb{R}^l \to \mathbb{R}^{l \times l}$, initial guess $a^{(0)} \in \mathbb{R}^l$

Output Approximation $\tilde{a}$ for the argument that minimizes g

1: Define $g' : \mathbb{R}^l \to \mathbb{R}^l$ by (3.8) if not specified
2: Define $g'' : \mathbb{R}^l \to \mathbb{R}^{l \times l}$ by (3.9) if not specified
3: $\tilde{a} = a^{(0)}$

$k = 0$
$r = g(\tilde{a})$
$q = r$
$s = g'(\tilde{a})$
$h = g''(\tilde{a})$
$d = h^{-1}s$
$\delta = s^*d$
$i = 0$
$stop = False$
while $i < m_{NCG2}$ and not $stop$ **do**
  $i \leftarrow i + 1$
  $\delta_d = d^*d$
  $\xi = \infty$
  $j = 0$
  while $j < m_{NCG1}$ and $\xi^2\delta_d < \epsilon_{NCG1}$ **do**
    $j \leftarrow j + 1$
    $\xi \leftarrow -\frac{s^*d}{d^*hd}$
    $\tilde{a} \leftarrow \tilde{a} + \xi d$
    $s \leftarrow g'(\tilde{a})$
    $h \leftarrow g''(\tilde{a})$
  end while
  $\hat{d} = -h^{-1}s$
  $\delta_0 = \delta$
  $\delta = -s^*\hat{d}$
  $\beta = \frac{\delta}{\delta_0}$
  if $k = l$ or $-s^*d < 0$ or $\beta \leq 0$ **then**
    $k \leftarrow 0$
    $d \leftarrow \hat{d}$
  else
    $k \leftarrow k + 1$
    $d \leftarrow \hat{d} + \beta d$
  end if
  $r \leftarrow g(\tilde{a})$
  if $\left|\frac{r-q}{q}\right| < \epsilon_{NCG2}$ **then**
    $stop \leftarrow True$
  end if
  $q \leftarrow r$
end while
Return $\tilde{a}$

Crank-Nicolson

Algorithm 3.6 CrankNicolson

Input $m_{CN} \in \mathbb{N}$, time step $\Delta t \in \mathbb{R}_0^+$, function $F : \mathbb{C}^n \to \mathbb{C}^n$, initial state $\Psi^{(0)} \in \mathbb{C}^n$

Output Approximations $\tilde{\Psi}^{(0)}, \ldots, \tilde{\Psi}^{(m_{CN})}$ of the time evolution of $\Psi^{(0)}$, corresponding to the PDE (3.10)

1: $\tilde{\Psi}^{(0)} = \Psi^{(0)}$
2: **for** $j = 0, \ldots, m_{CN} - 1$ **do**
3: Define G by (3.12) (with $\Psi^{(j)} = \tilde{\Psi}^{(j)}$)
4: Calculate $\tilde{\Psi}^{(j+1)}$ by executing NewtonClassic (algorithm 4.1) with $F = G$ and $x^{(0)} = \tilde{\psi}^{(j)}$
5: **end for**
6: Return $\tilde{\Psi}^{(0)}, \tilde{\Psi}^{(1)}, \ldots, \tilde{\Psi}^{(m_{CN})}$

CHAPTER 4

The Newton-Krylov method near bifurcation points

"Divide and conquer."

– Philip II of Macedon –

Chapter highlights:

- We review several existing Newton methods, and highlight their problems when applied to ill-conditioned nonlinear equations.
- We analyse how the approximate singularity of the nonlinear equation's Jacobian induces these problems.
- We derive a new class of Newton methods for use on ill-conditioned nonlinear equations, based on splitting the update vector in a range and kernel part.
- We show how the introduction of extra terms to the update vector gives way to an improvement in convergence.
- We explain how these extra terms are replaced by incorporating information contained in previous Newton iterations, yielding a computationally more efficient alternative.
- The results in the first part of this chapter are mainly based on the following references: [29, 46, 63].
- A journal article about the contents of this chapter and the next is in preparation.

4.1 Introduction

The Newton method is an iterative algorithm used to approximate the zeros of a nonlinear function $F : \mathbb{C}^n \to \mathbb{C}^n$ [29, 63]. In each iteration a linear system

with the functions Jacobian needs to be solved, in the Newton-Krylov method the solutions of these linear systems are approximated with a Krylov method (see section 3.3).

The algorithm used to perform numerical continuation, described in section 6.4, uses the Newton-Krylov method in combination with block elimination techniques (see chapter 5). When applied to general points of a system without continuous symmetries, the standard method generally works well and convergence is fast. Only in some special cases an adjusted method performs better. If the system contains continuous symmetries, we will see that a simple adjustment concerning the use of deflation yields a decent solver.

Bigger problems arise when the Newton method is executed near a bifurcation point, which is the case for the algorithm used to find such points (see section 6.6.2). The Jacobian of the function becomes more and more ill-conditioned the closer the algorithm converges to the bifurcation point [10, 62]. In order for the Newton method to work in this algorithm, adjustments need to be made. Before the algorithms combined with block elimination are described, we first derive the required adjustments for the normal Newton-Krylov method, ignoring block elimination.

The goal is to create an alternative Newton-Krylov method that also works for large-scale functions with an ill-conditioned Jacobian. Typically the Jacobian is only ill-conditioned near the solution we search, and well-conditioned for bad approximations. The alternative method will be based on splitting the unknown vector in a range and kernel part. Note that we will restrict ourselves to problems with a Hermitian Jacobian.

4.1.1 Literature review

Several adjustments to the Newton method have already been proposed to speed up convergence or to counter the issues caused by an ill-conditioned Jacobian. None of these are efficient for the large-scale systems considered in the thesis however. The current section reviews a selection of these alternative Newton methods.

Some of these methods use a perturbed linear system for the calculation of update vectors. In regularized Newton (RN) methods [69], for example, a specific diagonal matrix is added to the Jacobian whenever an update is calculated. By adding the perturbation, the used linear operator becomes nonsingular and quadratic convergence is expected.

The RN method is however developed for application to problems with a positive semi-definite Jacobian [69]. The problems discussed in the thesis (see chapter 2) do not necessarily have this property. Though an extension for general problems is made in [96], this requires application of the modified Cholesky factorization algorithm [22] to the Jacobian. For large-scale systems, this requires an unacceptable amount of computational work.

A similar approach as in the RN method is applied in [39, 112]. These papers discuss the Levenberg-Marquardt (LM) method, which also uses a perturbed linear system in each Newton step: the adjoint Jacobian is first applied to the standard system, and a perturbation by a specific diagonal matrix is

added. As discussed in [69], quadratic convergence of the LM method, applied to ill-conditioned problems, is not guaranteed.

Instead of basing each Newton step on the current guess for the solution, previous iterations are used as well in the accelerated inexact Newton (AIN) method [42]. Similar to Krylov methods, a search space is created for the solution of the nonlinear problem. Each step calculates the update vector of the standard Newton method by solving a linear system with the Jacobian, and adds this vector to the search space. The actual vector used to update the guess is then calculated as a linear combination in this space, e.g. by minimizing the residual norm.

Though the AIN method speeds up convergence for well-conditioned problems, it generally fails when the Jacobian is approximately singular. This causes errors in the search space, typically leading to stagnation of the residual. No diverging updates are used when the update vector is calculated by minimizing the residual norm however.

Similar to the AIN method we will not use the standard Newton update vector, but create one as a linear combination of other vectors. The coefficients for this linear combination will be calculated by minimizing the residual norm, just as in the AIN method.

In tensor methods (described by [94, 5]) the linear model of the Newton method is extended with a tensor term, resulting in a quadratic model. This term is chosen such that the model interpolates one or more previous iterates. By including second-order information, the authors hope to restore convergence even when the Jacobian is ill-conditioned. The methods described by [94, 5] are only applicable to small- or medium-scale problems. An extension for large-scale ones is made in [4], where the tensor model is locally minimized by a Krylov-like method.

Though applicable to the problems described in chapter 2, execution of a tensor method typically leads to stagnation of the residual norms before convergence up to the desired tolerance is reached. In [6] an average improvement of around 40% (in terms of nonlinear iterations and function evaluations) over standard methods (Newton-Krylov with line search, see section 4.4) is observed for singular problems with rank deficiency 1. Though this is a significant improvement for many problems, it is not sufficient when a stagnation of residual norms occurs.

Two similar methods are described by [11, 12] and [40]. The method of [11, 12] solves the local tensor model using standard Krylov subspace methods for linear equations, which involves constructing an inexact tensor step from the approximate solutions of two linear Jacobian systems. Convergence up to a specified tolerance is not guaranteed, raising the possibility of less accurate steps [6]. The method described in [40] uses GMRES to first create the classic Newton update, the actual update is then created as a linear combination of the used Arnoldi base by minimizing a projected version of the quadratic model. Though its performance is often on par with tensor methods, the method does not always converge [6]. It's also possible that the minimization of the quadratic model is not optimal, due to the requirement that the Krylov subspace used to

calculate the Newton update is large enough to capture important directions in the tensor step.

The strength of the methods that will be derived in the current chapter lies in gathering sufficient directions (vector parts) that include second-order information to the model, accounting for the lack of first-order information in the directions of the Jacobian's null vectors. Tensor methods solely rely on previous iterates to provide this information, which is often not sufficient.

Though not specifically developed for ill-conditioned problems, Chebyshev-Halley (CH) methods [53] are also based on a quadratic model, the second-order Taylor expansion around the guess. In [52] this model is solved by two linear systems for each iteration. An inexact version of the methods is defined in [102].

For well-conditioned problems the methods typically converge fast, but no convergence is guaranteed for ill-conditioned ones. One of the linear systems solved in each iteration uses the ill-conditioned Jacobian, yielding an unreliable solution. Nonsingularity of the operators that appear in the linear systems is assumed in [52, 102].

Another adaptation of the Newton method, with the goal of solving ill-conditioned nonlinear problems, is described by [63, 14]. The steps for calculating an update vector consist of solving two linear Jacobian systems, the update is then constructed as a linear combination of the two solutions.

Several constants appear in the method, chosen to speed up the convergence in the null space while keeping the guesses in the convergence region. A robust method to determine the optimal values of these constants does, however, not exist: a bad choice might result in diverging updates.

A generalization, based on applying multistage explicit Runge-Kutta discretization to the continuous Newton problem, is described by [82]. The method does however not converge unless certain conditions on the solution are met.

The recursive projection (RP) method [98] is developed to stabilize general fixed-point iterative procedures that solve nonlinear problems. This is done by computing a projection onto the unstable subspace, on which a special Newton iteration is performed. The fixed-point iteration is only applied to the complement. Examples in [98] show how the method manages to restore convergence of time integration methods for unstable equilibria.

For solving ill-conditioned problems by Newton's method (a fixed-point iterative procedure) the unstable subspace consists of approximate null vectors of the Jacobian. The goal of the RP method is to prevent convergence failure due to the Jacobian's approximate singularity by splitting the space in a stable and unstable part. Though the update part in the stable subspace is successfully solved by the fixed-point iteration, solving the part of the unstable subspace requires certain derivatives of the Jacobian's inverse. These are usually not analytically available. Approximation by a finite difference scheme is possible, but requires solving linear systems with the Jacobian. This operator is approximately singular, inducing convergence issues.

The adaptations of the Newton method described in the current chapter are based on splitting the update vector into parts concerning stable and un-

stable subspaces as well (see section 4.5), the unstable part is solved by the nonlinear conjugate gradients algorithm. This does not require application of the Jacobian's inverse to vectors in the unstable subspace.

The methods discussed so far are applicable to large-scale problems. Other methods focus on small- to medium-scale ones, requiring the Jacobian to be known as a (sparse) matrix. These methods cannot be applied to the ones considered in the thesis, for which the Jacobian is often only available as a linear operator. Methods developed for small- to medium-scale problems include [28, 56, 111].

4.1.2 Preliminaries

In the remainder of the chapter a nonlinear function $F : \mathbb{C}^n \to \mathbb{C}^n$ is considered, we try to find an approximation $\tilde{x} \in \mathbb{C}^n$ of the equation

$$F(x) = 0. \tag{4.1}$$

We assume that the Jacobian $F_x(x) : \mathbb{C}^n \to \mathbb{C}^n$ is known as a linear operator for each $x \in \mathbb{C}^n$, its matrix form is not required. If $F_x(x)$ is not available, it is approximated by a first-order central finite difference scheme [43]:

$$F_x(x) : \mathbb{C}^n \to \mathbb{C}^n : v \to \frac{1}{2\varepsilon} \left(F(x + \varepsilon v) - F(x - \varepsilon v) \right) \tag{4.2}$$

with $\varepsilon \in \mathbb{R}_0^+$ a small real number. Typically the choice $\varepsilon = \sqrt[3]{\varepsilon_{mach}}$ is made [63]. Since only problems with a Hermitian Jacobian are considered in the thesis, we further assume $F_x(x)$ to be self-adjoint, which also simplifies the notation. The analysis done in this chapter can be straightforwardly extended to non-Hermitian problems.

We assume the Hessian operator $F_{xx}(x) : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n$ to be known as a bilinear operator for each $x \in \mathbb{C}^n$ as well, its tensor form is not required. If not given, a second-order central scheme is used as approximation [43]:

$$\begin{aligned} F_{xx}(x) :& \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n : \\ & (v, w) \to \frac{1}{4\varepsilon^2} \left(F(x + \varepsilon v + \varepsilon w) - F(x + \varepsilon v - \varepsilon w) \right. \\ & \qquad \left. - F(x - \varepsilon v + \varepsilon w) + F(x - \varepsilon v - \varepsilon w) \right), \end{aligned} \tag{4.3}$$

with $\varepsilon \in \mathbb{R}_0^+$ typically chosen as $\varepsilon = \sqrt[4]{\varepsilon_{mach}}$ [63].

Continuous symmetries of F induce (approximate) null vectors in its Jacobian. Often these vectors are known, as in example 3.1. To save computational work they can be provided in the algorithms derived in this chapter. If these vectors are not known in advance, they are treated as normal null vectors and will be calculated with an eigenpair solver when required.

The algorithms described in this chapter allow for a Hermitian, positive definite preconditioner to be provided when eigenpairs need to be found or linear systems need to be solved. We assume a single function $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$

is given to construct the preconditioner $P(x)$ for each possible guess $x \in \mathbb{C}^n$. The derivation of the algorithms will be made for the unpreconditioned problem first, notes and required changes to the algorithms in the preconditioned case are provided afterwards. The algorithms themselves are always given such that they allow for preconditioning. The unpreconditioned case is derived from these by defining P as

$$P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$$

with $\mathcal{I}$ the identity linear operator.

4.2 The standard method

4.2.1 Description of the method

We start by describing the standard Newton-Krylov method [63]. To find the zeros of F, an initial guess $x^{(0)}$ needs to be provided. In each iteration i an update vector Δx is calculated by solving the linear system

$$F_x\left(x^{(i-1)}\right)\Delta x = -F\left(x^{(i-1)}\right) \tag{4.4}$$

with GMRES. The right-hand side of this system is often called the residual. The update vector is then used to update the guess:

$$x^{(i)} = x^{(i-1)} + \Delta x.$$

This new guess should be a better approximation than the first. Though it is possible for F to contain multiple zeros, only a single one is found when the Newton method is applied. Depending on the chosen initial guess the method will converge to a different solution.

The linear system (4.4) is derived from the Taylor expansion of $F(x)$. Given a guess $\tilde{x}$ and defining an update Δx such that $\tilde{x} + \Delta x = x$, we have

$$0 = F(x) = F(\tilde{x} + \Delta x) = F(\tilde{x}) + F_x(\tilde{x})\Delta x + \mathcal{O}(\|\Delta x\|^2). \tag{4.5}$$

By dropping terms higher than order 2 and rearranging the remaining ones, (4.4) is derived. A linear approximation is used in each iteration i:

$$F\left(x^{(i-1)}\right) + F_x\left(x^{(i-1)}\right)\Delta x \approx 0. \tag{4.6}$$

Algorithm 4.1 on page 100 in appendix 4.10 (taken from Kelley [63]) contains an implementation of the *standard Newton method*. In each iteration a linear system is solved with GMRES (see algorithm 3.4 on page 44) and the guess is updated. This process repeats itself until the residual norm $\|F(\tilde{x})\|$ is sufficiently low (below a tolerance ϵ_{New}), or after a maximum of m_{New} iterations.

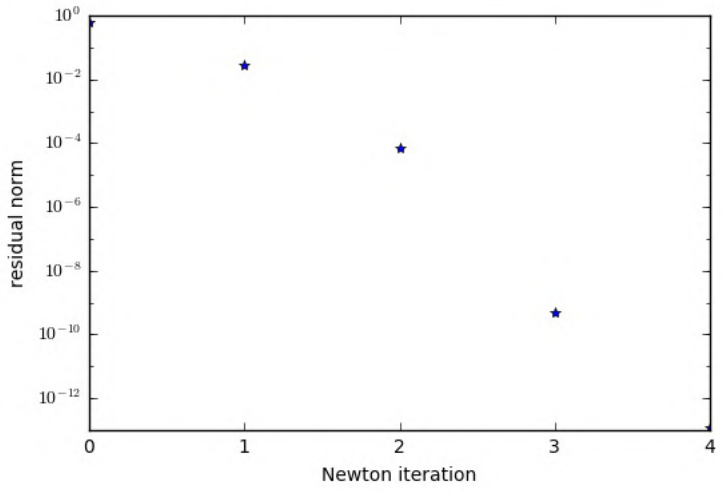


Figure 4.1: Residual plot of example 4.1. The standard Newton method is applied to a well-conditioned nonlinear problem. The residual norm converges to approximately 10^{-12} after 4 iterations, which is the attainable accuracy. Quadratic convergence is observed in the iterations.

4.2.2 Analysis of convergence for ill-conditioned problems

In example 4.1 the method is applied to the Liouville-Bratu-Gelfand equation. The algorithm is not executed near a bifurcation point, convergence is reached fast. The residual plot (figure 4.1) shows quadratic convergence, which is typical for the Newton method when applied to well-conditioned problems [63].

Example 4.1. Consider the Liouville-Bratu-Gelfand equation, described in section 2.4. A square domain with 900 discretization points is used. The standard Newton algorithm is executed to find a zero of (2.7) for the parameter $\mu = 0.3$. As an initial guess the vector

$$\psi^{(0)} = \begin{pmatrix} 0.372 \\ 0.372 \\ \vdots \\ 0.372 \end{pmatrix}$$

is used, this is a solution of the equation for $\mu = 0.2565$. The inner product (2.9) is used, a preconditioner is not applied. The residual plot of the problem is given by figure 4.1. After 4 Newton iterations, the algorithm converges to a tolerance of 10^{-12}. ◇

When applied to ill-conditioned problems, the standard Newton method generally fails to find a good approximation.

Lemma 4.2. An iteration of the standard Newton method possibly leads to a diverging update if executed in a guess $\tilde{x}$ for which the Jacobian $F_x(\tilde{x})$ is approximately singular. This is caused by a blow-up of the update vector in the directions of the null vectors.

Proof. We split the update vector Δx, calculated by (4.4), in a part $\Delta x_\parallel$ that lies in the approximate kernel of the current Jacobian (spanned by the vectors $\phi_1, \ldots, \phi_l$) and a part $\Delta x_\perp$ perpendicular to these vectors:

$$\Delta x = \Delta x_\perp + \Delta x_\parallel \tag{4.7}$$

$$= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j \tag{4.8}$$

with $a_1, \ldots, a_l \in \mathbb{R}$ given by

$$\forall j = 1, \ldots, l : a_j = -\lambda_j^{-1} \langle F(\tilde{x}), \phi_j \rangle, \tag{4.9}$$

λ_j the eigenvalue corresponding to ϕ_j ($\forall j = 1, \ldots, l$), and $\langle \Delta x_\perp, \phi_j \rangle = 0$ ($\forall j = 1, \ldots, l$). Similar to the analysis done in section 3.3.2, one can show that the kernel part $\Delta x_\parallel = \sum_{j=1}^{l} a_j \phi_j$ of the approximate update vector blows up when the Jacobian becomes too ill-conditioned. □

Lemma 4.2 shows how the approximate singularity of the Jacobian $F_x(\tilde{x})$ causes the possible failure of the standard Newton method: the local linear model (4.6) the method bases itself upon to calculate an update vector is not valid in the null vector directions. A diverging update might be calculated in this case, this results in a worse guess for the zero of F.

The standard Newton method is applied to two kinds of ill-conditioned problems in examples 4.3 and 4.4. Both examples are applied to a point in the Ginzburg-Landau equation.

In example 4.3 it is applied near a bifurcation point of the equation. Initially good updates are performed, for these iterations the approximation is far enough of the actual solution and the Jacobian does not contain approximate null vectors. When the approximate solution becomes better, a diverging update is performed though, due to the Jacobian becoming ill-conditioned. For this example, the Jacobian in the actual solution contains a three-dimensional approximate kernel ($l = 3$).

Example 4.4 shows another application of the method to a point in the Ginzburg-Landau equation. Though for this example no bifurcations are near, the convergence behaviour is still not optimal due to the Jacobian's singularity caused by the continuous symmetry of the equation. In the solution, the Jacobian contains a one-dimensional approximate kernel ($l = 1$).

Example 4.3. The Ginzburg-Landau equation (2.20), described in section 2.5, is considered. The equation is applied to a material shaped as a pentagon, $n = 10401$ discretization points are used. The magnetic field strength is fixed at $\mu = 1.05882456$. As an initial guess for the solution of $\mathcal{F}(\psi, \mu) = 0$ (with

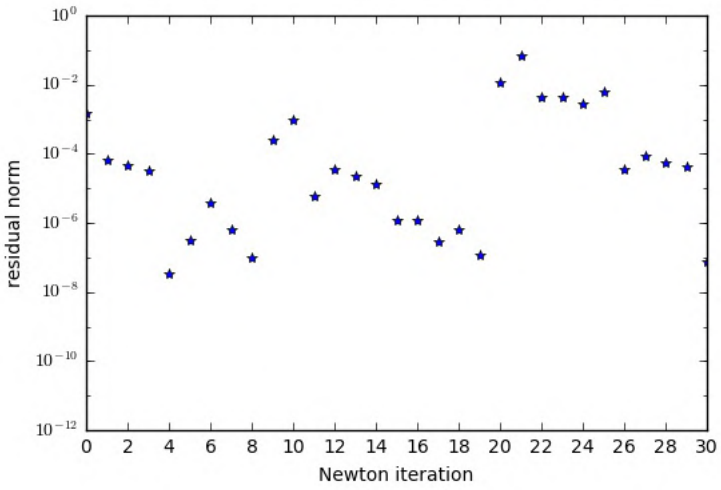


Figure 4.2: Residual plot of example 4.3. The standard Newton method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm does not converge, we observe a mixture of converging steps and sudden jumps that increase the residual norm.

$\mathcal{F}$ given by (2.20)), one of the solutions at $\mu = 1.059$ is used. The considered solution is the one that lies on branch B in figure 9.23 (see section 9.5.3). For the current set-up, the Jacobian (2.22) will be nearly singular in the solution of the problem due to a nearby bifurcation point. We use (2.21) as an inner product, (2.25) is used to precondition the linear systems.

Application of the standard Newton algorithm yields the residual plot in figure 4.2. For this example convergence is not reached up to the desired tolerance ($\epsilon = 10^{-12}$) within 30 Newton iterations. Some iterations show a blow-up of the residual norm, this is caused by the approximate singularity of the Jacobian. ◇

Example 4.4. The same equation, domain, inner product and preconditioner as for example 4.3 are considered. This time the magnetic field strength is fixed at $\mu = 1.4$, and one of the solutions (on branch B in figure 9.23) at $\mu = 1.383$ is used as an initial guess.

The residual plot of the standard Newton algorithm is given by figure 4.3. Though convergence (up to a tolerance $\epsilon = 10^{-12}$) is reached in 9 iterations, some of the Newton iterations yield a blow-up of the residual norm. This is again caused by the approximate singularity of the Jacobian near the solution of the problem. ◇

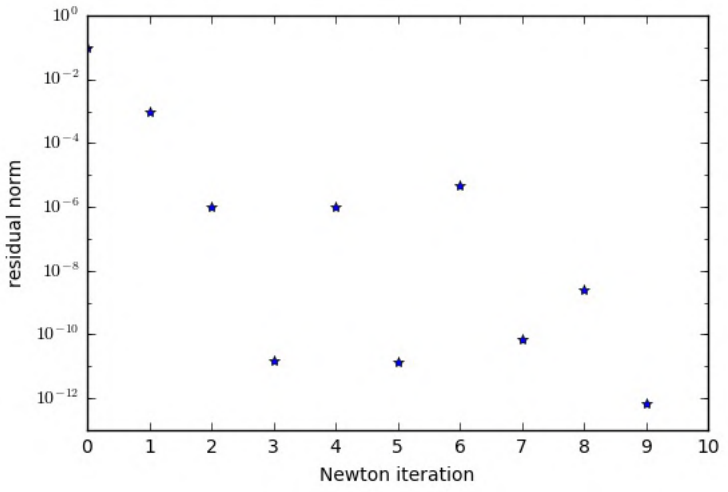


Figure 4.3: Residual plot of example 4.4. The standard Newton method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a one-dimensional kernel in the searched solution. This kernel is induced by a continuous symmetry. The residual norm converges to approximately 10^{-12} after 9 iterations, which is the attainable accuracy. Some of the iterations show sudden jumps that increase the residual norm.

4.2.3 The preconditioned case

If a preconditioner is provided, the linear system

$$P\left(x^{(i-1)}\right) F_x\left(x^{(i-1)}\right) \Delta x = -P\left(x^{(i-1)}\right) F\left(x^{(i-1)}\right) \tag{4.10}$$

is solved with GMRES, instead of (4.4). The convergence criterium is adjusted as well: Newton iterations are executed until the preconditioned residual norm $\|F(\tilde{x})\|_{P(\tilde{x})}$ is sufficiently low (or until a maximum of m_{New} iterations). The remainder of the method is unchanged.

Given a guess $\tilde{x} \in \mathbb{C}^n$, (4.10) is derived by substituting the Taylor expansion (4.5) of $F(x)$ in the equation

$$P(\tilde{x})F(x) = 0$$

and dropping terms of order 2 and higher.

The preconditioned method has the same problem as the unpreconditioned one. The update vector Δx can be split in a part $\Delta x_{\|}$ consisting of approximate null vectors $\phi_1, \ldots, \phi_l$ of the preconditioned Jacobian $P(\tilde{x})F_x(\tilde{x})$ and a part

$\Delta x_\perp$ perpendicular to these vectors:

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j \end{aligned} \tag{4.11}$$

with $a_j = -\lambda_j^{-1} \langle P(\tilde{x})F(\tilde{x}), \phi_j \rangle_{P(\tilde{x})^{-1}}$ $(\forall j = 1, \ldots, l)$ and $\langle \Delta x_\perp, \phi_j \rangle_{P(\tilde{x})^{-1}} = 0$ $(\forall j = 1, \ldots, l)$. When the Jacobian becomes too ill-conditioned, the part $\Delta x_\parallel = \sum_{j=1}^{l} a_j \phi_j$ blows up.

4.3 The use of deflation to prevent divergence

4.3.1 Description of the method

For problems with an ill-conditioned Jacobian the standard method (discussed in section 4.2) did not work well, due to the local linear model (4.6) not being valid in null vector directions. Consider a guess $\tilde{x} \in \mathbb{C}^n$. If the update vector Δx, calculated by (4.4), is split into

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j, \end{aligned} \tag{4.12}$$

with $\phi_1, \ldots, \phi_l$ the approximate null vectors of $F_x(\tilde{x})$, $a_1, \ldots, a_l \in \mathbb{R}$ and $\langle \Delta x_\perp, \phi_j \rangle = 0$ $(\forall j = 1, \ldots, l)$, we observe that the kernel part, $\Delta x_\parallel = \sum_{j=1}^{l} a_j \phi_j$, blows up. This possibly leads to a diverging update, resulting in a worse guess for the searched zero of F.

We try to use deflation to prevent this. Given a matrix K with the approximate null vectors $\phi_1, \ldots, \phi_l$ of $F_x(\tilde{x})$ as its columns, the projection operator $\mathcal{Q}(\tilde{x})$ is defined as

$$\mathcal{Q}(\tilde{x}) : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, K \rangle^{-1} \langle K, y \rangle. \tag{4.13}$$

Instead of solving (4.4), the deflated linear system

$$\mathcal{Q}(\tilde{x}) F_x(\tilde{x}) \Delta x = -\mathcal{Q}(\tilde{x}) F(\tilde{x}) \tag{4.14}$$

is now used to calculate Δx [46] . Due to the use of deflation, the calculated update vector Δx is perpendicular to the approximate null vectors: $\langle \Delta x, \phi_j \rangle = 0$ $(\forall j = 1, \ldots, l)$. For this deflated system, the local linear model (4.6) is only applied to vectors different from the ones in the kernel. The kernel part of the update vector Δx is not blown up, and a decent Newton update is expected.

Note that the use of the operator $\mathcal{Q}(\tilde{x})$ is equivalent to setting $a_j = 0$ $(\forall j = 1, \ldots, l)$ in (4.12). We have $\Delta x = \Delta x_\perp$ in this equation.

We call the adjusted Newton method, where the deflated linear system (4.14) is used, the *deflated Newton method*. It is described by algorithm 4.2 on

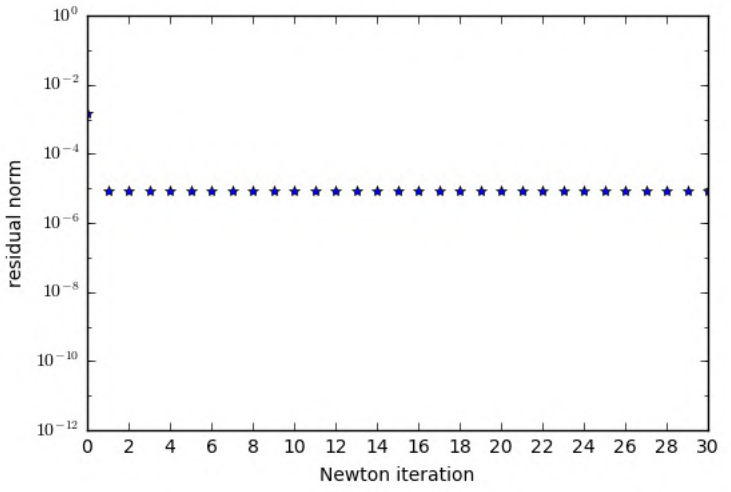


Figure 4.4: Residual plot of example 4.5. The deflated Newton method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-5} after 2 iterations, the attainable accuracy for well-conditioned problems is however 10^{-12}. After 2 iterations there is no further significant decrease.

page 100 in appendix 4.10. In each iteration approximate null vectors of the Jacobian are calculated with an eigenpair solver (algorithm 3.3, page 43). If the system contains continuous symmetries and it is known how the corresponding induced null vectors are calculated, these vectors can be provided separately to save computational work. After the approximate null vectors have been calculated, deflated GMRES (algorithm 3.4, page 44) is executed to solve the linear system. This yields an update vector used to calculate the next guess. Note that in absence of any null vectors, algorithm 4.2 is equivalent to algorithm 4.1.

4.3.2 Analysis of convergence for ill-conditioned problems

The same problem as described by example 4.3 is used in example 4.5: we apply the deflated Newton method near a bifurcation of the Ginzburg-Landau equation.

Example 4.5. Consider the same set-up as in example 4.3. Application of the deflated Newton algorithm yields the residual plot in figure 4.4. After an initial good update, the residual norms stagnate at approximately 10^{-5}. ◇

Though no diverging updates are performed for this example, the residual norms stagnate, possibly before the desired tolerance is reached. Note that quadratic convergence is typically observed during the Newton iterations before

stagnation, just like the standard method applied to well-conditioned problems. The convergence behaviour is improved for the new method, but it is often not yet sufficient for the application we are interested in (see section 6.6.2). The stagnation of the residual is caused by the use of deflation.

Lemma 4.6. The deflated Newton method typically leads to stagnation of residuals when applied to a problem for which the Jacobian is (approximately) singular in the searched solution. This is caused by the update part consisting of approximate null vectors being ignored.

Proof. When we split the calculated update vector Δx (see (4.12)),

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j, \end{aligned}$$

the values for $a_1, \ldots, a_l$ are set to 0 when deflated Newton is applied. Although the $\Delta x_\perp$ part of the update is calculated well, no update in the direction of the approximate null vectors is performed ($\Delta x_\parallel = 0$). Stagnation occurs when the residual $F(\tilde{x})$ resembles a linear combination of these null vectors, which is often the case after some initial iterations. The right-hand side $-\mathcal{Q}(\tilde{x})F(\tilde{x})$ of (4.14) equals the zero vector in this case, resulting in a zero update vector Δx. No update is performed, resulting in stagnation. □

To prevent stagnation of the residual norm, the approximate null vector part $\Delta x_\parallel$ of the update vector Δx should not be ignored.

Note that if the Jacobians singularity is solely caused by null vectors induced by continuous symmetries, the deflated Newton method often works fine. This is the case for example 4.7. Due to the continuous symmetry, each solution x of $F(x) = 0$ is part of a family of solutions. It is sufficient to calculate a single representative of such a family. Deflation prevents us from searching a solution in certain null vector directions, but these are not required to reach a single representative. Due to the continuous symmetries, we don't need the null vector directions they induce to find a solution.

Example 4.7. Consider the same set-up as in example 4.4. Figure 4.5 represents the residual plot associated with application of the deflated Newton method. Convergence (up to a tolerance of 10^{-12}) is reached in 4 iterations. Similar to example 4.1, quadratic convergence is observed for the current example. ◇

4.3.3 The preconditioned case

In the preconditioned case, the projection operator $\mathcal{Q}(\tilde{x})$ required for deflation is defined by

$$\mathcal{Q}(\tilde{x}) : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, L\rangle^{-1}\langle L, y\rangle, \tag{4.15}$$

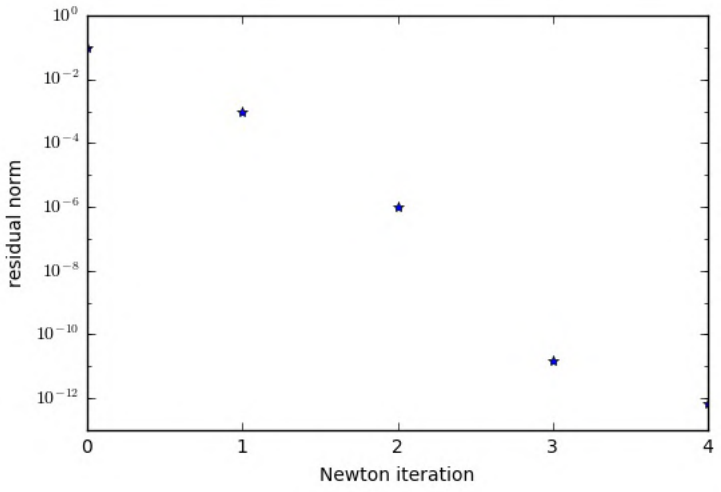


Figure 4.5: Residual plot of example 4.7. The deflated Newton method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a one-dimensional kernel in the searched solution. This kernel is induced by a continuous symmetry. The residual norm converges to approximately 10^{-12} after 4 iterations, which is the attainable accuracy. Quadratic convergence is observed in the iterations.

with

$$K = \begin{pmatrix} \tau_1 & \tau_2 & \dots & \tau_l \end{pmatrix},$$
$$L = \begin{pmatrix} \phi_1 & \phi_2 & \dots & \phi_l \end{pmatrix}$$

such that $P(\tilde{x})K = L$. The vectors $\phi_1, \dots, \phi_l$ are the approximate null vectors of $P(\tilde{x})F_x(\tilde{x})$. The linear system to solve in each Newton iteration becomes

$$P(\tilde{x})\mathcal{Q}(\tilde{x})F_x(\tilde{x})\Delta x = -P(\tilde{x})\mathcal{Q}(\tilde{x})F(\tilde{x}). \tag{4.16}$$

The calculated update vector Δx satisfies

$$\forall j = 1, \dots, l : \langle \Delta x, \phi_j \rangle_{P(\tilde{x})^{-1}} = 0.$$

A similar split as (4.11) shows that any updates in the direction of approximate null vectors $\phi_1, \dots, \phi_l$ are ignored (the calculated Δx approximates $\Delta x_\perp$ in (4.11)), possibly leading to stagnation of the residual norm.

4.4 The use of line search to prevent divergence

4.4.1 Description of the method

The use of deflation prevents the blow-up of update vectors, but leads to stagnation of the residual norm. A different method to counter this blow-up is to

apply a line search after the update vector Δx is calculated from the standard linear system (4.4). Consider a guess $\tilde{x} \in \mathbb{C}^n$. Instead of the update

$$\tilde{x} \leftarrow \tilde{x} + \Delta x,$$

we consider an update of the form [29]

$$\tilde{x} \leftarrow \tilde{x} + \xi \Delta x,$$

where $\xi \in \mathbb{R}$ is calculated by minimizing the function

$$g : \mathbb{R} \to \mathbb{R} : \xi \to \|F(\tilde{x} + \xi \Delta x)\|^2 . \tag{4.17}$$

Note that typically $\xi \in [0; 1]$, but this is not guaranteed. In practice the value for ξ is approximated with the nonlinear conjugate gradients (NCG) algorithm (see section 3.4). This requires the first and second derivative of g, given by

$$g' : \mathbb{R} \to \mathbb{R} : \xi \to 2\langle F(\tilde{x} + \xi \Delta x), F_x(\tilde{x} + \xi \Delta x)\Delta x\rangle, \tag{4.18}$$

$$\begin{aligned} g'' : \mathbb{R} \to \mathbb{R} : \xi \to & 2\langle F_x(\tilde{x} + \xi \Delta x)\Delta x, F_x(\tilde{x} + \xi \Delta x)\Delta x\rangle \\ & + 2\langle F(\tilde{x} + \xi \Delta x), F_{xx}(\tilde{x} + \xi \Delta x)\Delta x \Delta x\rangle. \end{aligned} \tag{4.19}$$

The linear system (4.4) remains untouched when line search is applied, the update vector Δx will contain parts in the approximate null vector directions, as well as in directions perpendicular to these. It is possible for Δx to be blown up due to the Jacobian's singularity, but by choosing ξ sufficiently small, this does not lead to a diverging Newton update.

The adaptation of the standard Newton method with a line search technique is given by algorithm 4.3 on page 101 in appendix 4.10. This method will be called the *Newton method with line search* (NLS) in the remainder of the chapter. In each iteration a linear system is solved as in the standard algorithm, but a line search is performed with nonlinear conjugate gradients before the guess is updated. As an initial guess for this minimization, the value $\xi^{(0)} = \min(1, 2\|F(\tilde{x})\|\|\Delta x\|^{-2})$ is chosen. This choice is explained in section 4.4.2 (see the bound given in lemma 4.9).

4.4.2 Analysis of convergence for ill-conditioned problems

The algorithm is applied in example 4.8 to the same problem as the one described in examples 4.3 and 4.5.

Example 4.8. Consider the same set-up as in example 4.3. Application of the NLS method yields the residual plot in figure 4.6. The initial 5 Newton iterations show a decent decrease in residual norm. After these iterations, convergence slows down significantly: the residual norm remains at approximately 10^{-8}. In fact the norm keeps decreasing, but this happens very slowly. ◇

Though no diverging updates are performed and initially convergence behaves fine, the decrease in residual norm slows down significantly after only a few iterations.

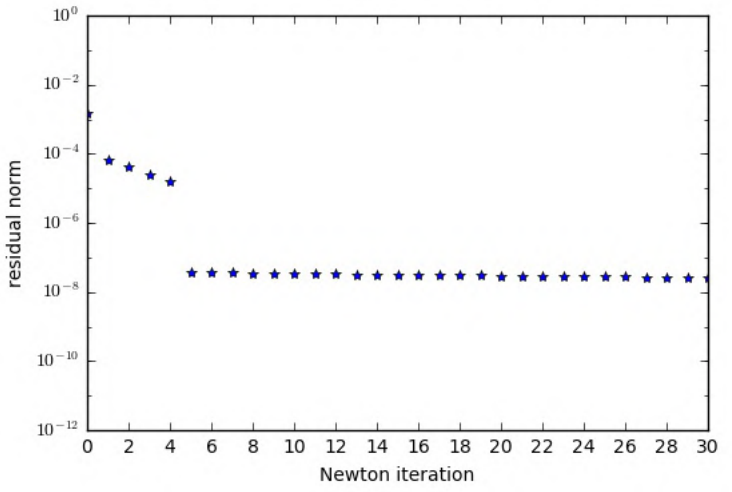


Figure 4.6: Residual plot of example 4.8. The NLS method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm decreases to approximately 10^{-8} after 5 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps.

Lemma 4.9. After some initial iterations, convergence of the NLS method is typically slow when applied to an ill-conditioned problem. The slowdown of convergence has multiple causes:

- The update part perpendicular to the Jacobian's null vectors is damped due to the multiplication with ξ. This part does not cause any problems and is actually approximated well, its damping might lead to an unnecessarily small update.
- The value ξ used in updating the guess $\tilde{x}$ typically satisfies the bound

$$|\xi| \lesssim 2\|F(\tilde{x})\|\|\Delta x\|^{-2}, \tag{4.20}$$

  The update vector Δx is calculated from (4.4), and typically blows up if the Jacobian in $\tilde{x}$ is approximately singular.

Proof. The first cause is explained by splitting the update vector Δx as in (4.12):

$$\begin{aligned} \Delta x &= \Delta x_{\perp} + \Delta x_{\parallel} \\ &= \Delta x_{\perp} + \sum_{j=1}^{l} a_j \phi_j. \end{aligned} \tag{4.21}$$

The values for $a_1, \ldots, a_l$ are fixed, they are given by (see (4.9))

$$\forall j = 1, \ldots, l : a_j = -\lambda_j^{-1} \langle F(\tilde{x}), \phi_j \rangle.$$

When line search is applied, the actual vector used to update the guess is given by

$$\begin{aligned} \xi \Delta x &= \xi \Delta x_\perp + \xi \Delta x_\parallel \\ &= \xi \Delta x_\perp + \xi \sum_{j=1}^{l} a_j \phi_j. \end{aligned} \tag{4.22}$$

During the initial iterations of the algorithm, the Jacobian is often still well-conditioned because the approximation $\tilde{x}$ lies far enough from the solution. The amount l of approximate null vectors equals zero for these steps, and decent convergence behaviour is expected. The values ξ that minimize the function (4.17) should approximate 1.

After these initial iterations, the guess $\tilde{x}$ lies sufficiently close to the solution for the Jacobian to become ill-conditioned. We have $|a_j| \gg 1$ $(\forall j = 1, \ldots, l)$ in (4.21) and the update vector Δx blows up. To prevent a diverging Newton update, the value for ξ has to be chosen small, such that the $\xi \sum_{j=1}^{l} a_j \phi_j$ term in (4.22) is damped.

The part $\Delta x_\perp$, perpendicular to the approximate null vectors $\phi_1, \ldots, \phi_l$, is however damped as well, though it did not blow up and actually represents a good update in the directions perpendicular to $\phi_1, \ldots, \phi_l$.

To analyse the second cause, consider the second-order Taylor expansion for the new residual $F(\tilde{x} + \xi \Delta x)$:

$$\begin{aligned} F(\tilde{x} + \xi \Delta x) &= F(\tilde{x}) + \xi F_x(\tilde{x}) \Delta x + \frac{1}{2} \xi^2 F_{xx}(\tilde{x}) \Delta x \Delta x + \mathcal{O}(\xi^3 \|\Delta x\|^3) \\ &= (1 - \xi) F(\tilde{x}) + \frac{1}{2} \xi^2 F_{xx}(\tilde{x}) \Delta x \Delta x + \mathcal{O}(\xi^3 \|\Delta x\|^3). \end{aligned}$$

The second equality follows from (4.4). The standard Newton method (see section 4.2) corresponds to choosing $\xi = 1$, for which the $F(\tilde{x})$ term disappears in above expansion. Line search does not make this choice if $\|\Delta x\| \gg 1$, which happens when the Jacobian is ill-conditioned: the $F(\tilde{x})$ term will not disappear in this Taylor expansion. The second derivative term possibly blows up for the choice $\xi = 1$, since its order is given by

$$\frac{1}{2} \xi^2 F_{xx}(x) \Delta x \Delta x = \mathcal{O}(\frac{1}{2} \xi^2 \|\Delta x\|^2).$$

To prevent the second derivative from dominating the residual $F(\tilde{x} + \xi \Delta x)$ of the next iteration, we should require

$$\frac{1}{2} \xi^2 \|\Delta x\|^2 \lesssim |\xi| \|F(\tilde{x})\|. \tag{4.23}$$

Note that we assumed $\|F(\tilde{x})_{xx} \Delta x \Delta x\| \approx \|\Delta x\|^2$ for simplicity. If this approximation does not hold, $\|F(\tilde{x})_{xx} \Delta x \Delta x\|$ should be used in (4.23) and results derived from this condition.

By imposing (4.23) the influence of the second derivative on the new residual is not greater than that of the first. This first derivative is used by the Newton method to reduce the current residual $F(\tilde{x})$. Rewriting inequality (4.23) yields the bound (4.20) on ξ. □

The first cause mentioned in lemma 4.9 concerns the part $\Delta x_\perp$ of the update vector. When line search is used, this part is possibly much smaller than required, leading to slow convergence.

The second cause concerns a bound on the entire update. If $\|\Delta x\| \gg 1$ (which happens for ill-conditioned problems), bound (4.20) is very strict. In this case we have $\xi \ll 1$. The residual term $F(\tilde{x})$ only reduces a little in each iteration, causing slow convergence, as is seen in figure 4.6.

In the special case where $F(\tilde{x})$ equals $b\phi$, with $b \in \mathbb{R}$ and $\phi \in \mathbb{C}^n$ an approximate null vector of the Jacobian such that $\|\phi\| = 1$, the update vector Δx is given by

$$\Delta x = -\frac{b}{\lambda}\phi.$$

Bound (4.20) becomes

$$|\xi| \lesssim 2\frac{\lambda^2}{|b|}$$

for this special case, which becomes more strict the closer λ gets to zero. If the Jacobian of F is exactly singular in the solution we search, this happens the further we converge.

4.4.3 The preconditioned case

Similar to the adjustments made for the standard method, the linear system

$$P(\tilde{x})F_x(\tilde{x})\Delta x = -P(\tilde{x})F(\tilde{x})$$

is solved with GMRES in the preconditioned case of the NLS method. The function to minimize needs to be adjusted for the preconditioned norm, and becomes

$$g : \mathbb{R} \to \mathbb{R} : \xi \to \|F(\tilde{x} + \xi\Delta x)\|^2_{P(\tilde{x})} \tag{4.24}$$

with derivatives

$$g' : \mathbb{R} \to \mathbb{R} : \xi \to 2\langle F(\tilde{x} + \xi\Delta x), F_x(\tilde{x} + \xi\Delta x)\Delta x\rangle_{P(\tilde{x})}, \tag{4.25}$$

$$\begin{aligned} g'' : \mathbb{R} \to \mathbb{R} : \xi \to & 2\langle F_x(\tilde{x} + \xi\Delta x)\Delta x, F_x(\tilde{x} + \xi\Delta x)\Delta x\rangle_{P(\tilde{x})} \\ & + 2\langle F(\tilde{x} + \xi\Delta x), F_{xx}(\tilde{x} + \xi\Delta x)\Delta x\Delta x\rangle_{P(\tilde{x})}. \end{aligned} \tag{4.26}$$

The value

$$\xi^{(0)} = \min\left(1, 2\|F(\tilde{x})\|_{P(\tilde{x})}\|\Delta x\|^{-2}_{P(\tilde{x})^{-1}}\right) \tag{4.27}$$

is used as an initial guess for the minimization of (4.24).

The preconditioned method again typically shows a slowdown in convergence when applied to an ill-conditioned problem, similar to the unpreconditioned case. Splitting the update vector Δx as in (4.11), the part $\Delta x_\perp$ is

possibly damped under the line search, even though the damping of this part is unjustified.

The second cause of slow convergence we analysed is similar for the preconditioned case as well. Using a second-order Taylor expansion we have

$$\begin{aligned} P(\tilde{x})F(\tilde{x}+\xi\Delta x) &= P(\tilde{x})F(\tilde{x}) + \xi P(\tilde{x})F_x(\tilde{x})\Delta x + \frac{1}{2}\xi^2 P(\tilde{x})F_{xx}(\tilde{x})\Delta x\Delta x \\ &\quad + \mathcal{O}(\xi^3\|\Delta x\|^3) \\ &= (1-\xi)P(\tilde{x})F(\tilde{x}) + \frac{1}{2}\xi^2 P(\tilde{x})F_{xx}(\tilde{x})\Delta x\Delta x \\ &\quad + \mathcal{O}(\xi^3\|\Delta x\|^3). \end{aligned}$$

To prevent dominance of the second derivative, we require

$$\frac{1}{2}\xi^2\|\Delta x\|^2_{P(\tilde{x})^{-1}} \lesssim |\xi|\|P(\tilde{x})F_x(\tilde{x})\|_{P(\tilde{x})^{-1}},$$

yielding the bound

$$|\xi| \lesssim 2\|F(\tilde{x})\|_{P(\tilde{x})}\|\Delta x\|^{-2}_{P(\tilde{x})^{-1}}. \tag{4.28}$$

This is the preconditioned version of bound (4.20). For ill-conditioned problems we have $\|\Delta x\|_{P(\tilde{x})^{-1}} \gg 1$, implying $|\xi| \ll 1$. The update barely reduces the current residual, resulting in slow convergence.

4.5 Splitting the update vector

4.5.1 Description of the method

Updating the Newton method with deflation or a line search solves the problem of diverging updates occurring, but introduces additional problems on the convergence behaviour: Often stagnation or very slow convergence is observed. To counter these problems we again split the update vector Δx into a part $\Delta x_{\|}$ consisting of approximate null vectors $\phi_1, \ldots, \phi_l$ of $F_x(\tilde{x})$ (with $\|\phi_1\| = \cdots = \|\phi_l\| = 1$), and a part $\Delta x_\perp$ perpendicular to these vectors:

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_{\|} \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j\phi_j, \end{aligned} \tag{4.29}$$

The use of deflation (see section 4.3) yields a good approximation for $\Delta x_\perp$. With

$$K = \begin{pmatrix} \phi_1 & \phi_2 & \ldots & \phi_l \end{pmatrix},$$

we again define the projection operator $\mathcal{Q}(\tilde{x})$ as

$$\mathcal{Q}(\tilde{x}) : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K\langle K, K\rangle^{-1}\langle K, y\rangle.$$

The $\Delta x_\perp$ part of the update vector is now calculated by solving

$$\mathcal{Q}(\tilde{x})F_x(\tilde{x})\Delta x_\perp = -\mathcal{Q}(\tilde{x})F(\tilde{x}) \tag{4.30}$$

with (deflated) GMRES. To prevent further stagnation, we have yet to find the update part $\Delta x_{\parallel}$ in the directions of $\phi_1, \ldots, \phi_l$. This corresponds to finding the values for $a_1, \ldots, a_l \in \mathbb{R}$ in (4.29). Similar to applying a line search, we will calculate these values by minimizing the function

$$g : \mathbb{R}^l \to \mathbb{R} : a \to \left\| F\left(\tilde{x} + \Delta x_{\perp} + \sum_{j=1}^{l} a_j \phi_j\right)\right\|^2 \tag{4.31}$$

with the nonlinear conjugate gradients method. Note that the update part $\Delta x_{\perp}$ is kept constant in this minimization. The first and second partial derivatives of g are given by $(\forall i, k = 1, \ldots, l)$

$$\frac{\partial g}{\partial a_i} : \mathbb{R}^l \to \mathbb{R} : a \to 2\langle F(x(a)), F_x(x(a))\phi_i\rangle, \tag{4.32}$$

$$\begin{aligned}\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \to \mathbb{R} : a \to & 2\langle F_x(x(a))\phi_i, F_x(x(a))\phi_k\rangle \\ & + 2\langle F(x(a)), F_{xx}(x(a))\phi_i\phi_k\rangle,\end{aligned} \tag{4.33}$$

$$\text{with } x(a) = \tilde{x} + \Delta x_{\perp} + \sum_{j=1}^{l} a_j \phi_j.$$

The minimization is only applied to the coefficients of the null vector directions, the part $\Delta x_{\perp}$ is not updated. The damping of $\Delta x_{\perp}$ was one of the causes of slow convergence discussed in section 4.4 (see lemma 4.9), but should not occur for the technique described in the current section. We call this new method, where the two update vector parts are calculated separately, the *split Newton method* (SN).

The SN method is described by algorithm 4.4 on page 101 in appendix 4.10. Each Newton iteration first approximates null vectors by applying an eigensolver (algorithm 3.3, page 43). Next, a deflated linear system is solved with GMRES (algorithm 3.4, page 44) for the first part of the update vector. The eventual update vector is then calculated by application of the nonlinear conjugate gradients method (algorithm 3.5, page 45) to minimize (4.48), the preconditioned version of (4.31). The initial guess $a^{(0)}$ used for this minimization is derived in section 4.5.3. After updating the guess, the next iteration is started.

4.5.2 Analysis of convergence for ill-conditioned problems

The algorithm is applied in example 4.10, the same setting as before is used.

Example 4.10. Consider the same set-up as in example 4.3. Application of the SN method yields the residual plot in figure 4.7. After 30 iterations, the residual norm reaches a value of approximately 10^{-7}. ◇

We again observe slow convergence after the initial iteration. Though the $\Delta x_{\perp}$ part should be calculated well by the SN method, we did not make adjustments to improve the bounds implied by second order derivatives.

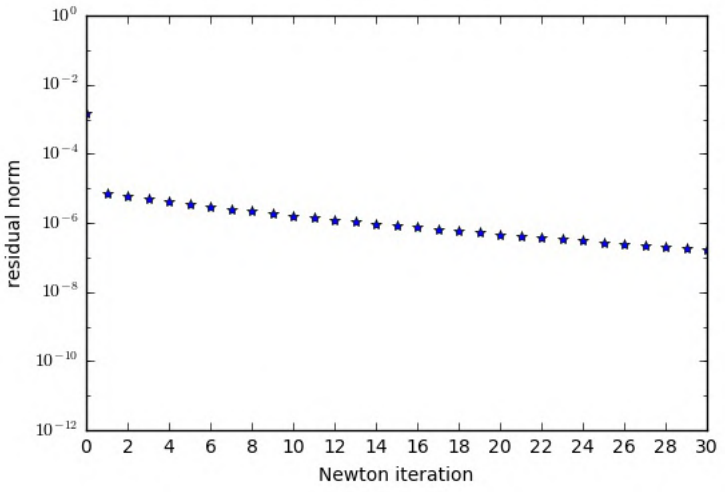


Figure 4.7: Residual plot of example 4.10. The SN method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm decreases to approximately 10^{-5} after 1 iteration. Further iterations show a slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps. The slowdown in convergence is more profound in later iterations, where the guess lies closer to the solution. After 30 iterations the residual norm reaches a value of approximately 10^{-7}.

Lemma 4.11. When applied to a problem with a (near) singular Jacobian in the solution, convergence of the SN method typically slows down after some initial iterations. This is caused by the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim |\lambda_s| \tag{4.34}$$

typically being satisfied by the $a_1, \ldots, a_l$ values in the update vector (4.29). The index s in (4.34) is chosen such that $|a_s \lambda_s|$ is maximal. $\lambda_1, \ldots, \lambda_l$ are the eigenvalues of approximate null vectors, so we have $\lambda_s \approx 0$, inducing small sized updates.

Proof. The slowdown in convergence is modeled and analysed in a similar way as in section 4.4. Consider the second-order Taylor expansion for the new residual:

$$F\left(\tilde{x} + \Delta x_\perp + \Delta x_\parallel\right) \tag{4.35}$$

$$\begin{aligned} = \; & F(\tilde{x}) + F_x(\tilde{x})\Delta x_\perp + F_x(\tilde{x})\Delta x_\parallel + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\perp \Delta x_\perp \\ & + F_{xx}(\tilde{x})\Delta x_\perp \Delta x_\parallel + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\parallel \Delta x_\parallel + \mathcal{O}(\|\Delta x\|^3). \end{aligned} \tag{4.36}$$

Substitution of $\Delta x_{\|}$ yields

$$F\left(\tilde{x} + \Delta x_{\perp} + \Delta x_{\|}\right) \tag{4.37}$$

$$\begin{aligned} &= F(\tilde{x}) + F_x(\tilde{x})\Delta x_{\perp} + \sum_{j=1}^{l} a_j F_x(\tilde{x})\phi_j + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_{\perp}\Delta x_{\perp} \\ &\quad + \sum_{j=1}^{l} a_j F_{xx}(\tilde{x})\Delta x_{\perp}\phi_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x})\phi_i\phi_j + \mathcal{O}(\|\Delta x\|^3). \end{aligned} \tag{4.38}$$

We split the current residual $F(\tilde{x})$ in

$$F(\tilde{x}) = F_{\perp} + F_{\|} \tag{4.39}$$

$$= F_{\perp} + \sum_{j=1}^{l} b_j \phi_j, \tag{4.40}$$

with $b_j = \langle F(\tilde{x}), \phi_j \rangle$ $(\forall j = 1, \ldots, l)$ and $\langle F_{\perp}, \phi_j \rangle = 0$ $(\forall j = 1, \ldots, l)$. Since $F_{\perp} = \mathcal{Q}(\tilde{x})F(\tilde{x})$, (4.30) implies that the calculated part $\Delta x_{\perp}$ satisfies the approximation $F_x(\tilde{x})\Delta x_{\perp} \approx -F_{\perp}$. With $\lambda, \ldots, \lambda_l$ the respective eigenvalues corresponding to the approximate null vectors $\phi_1, \ldots, \phi_l$, the Taylor expansion yields

$$\begin{aligned} &F\left(\tilde{x} + \Delta x_{\perp} + \Delta x_{\|}\right) \\ &\quad \approx \sum_{j=1}^{l} \left(b_j + a_j\lambda_j\right)\phi_j + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_{\perp}\Delta x_{\perp} + \sum_{j=1}^{l} a_j F_{xx}(\tilde{x})\Delta x_{\perp}\phi_j \\ &\qquad + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x})\phi_i\phi_j + \mathcal{O}(\|\Delta x\|^3). \end{aligned}$$

We need to choose the values of $a_1, \ldots, a_l$ such that the approximate null vector part $F_{\|} = \sum_{j=1}^{l} b_j\phi_j$ of the current residual reduces. To completely remove this term, the choice

$$\forall j = 1, \ldots, l : a_j = -\frac{b_j}{\lambda_j} \tag{4.41}$$

needs to be made. Due to the Jacobians ill-conditionedness, this choice would imply $|a_j| \gg 1$ $(\forall j = 1, \ldots, l)$. This results in a blow-up of the new residual due to the second order derivative term $\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x})\phi_i\phi_j$. There is a trade-off between reducing the current residual and preventing such a blow-up. To prevent dominance of the second order derivative in the new residual, we should have

$$\max_{i,j=1,\ldots,l} |a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j\lambda_j|. \tag{4.42}$$

Under this condition the influence of the second order derivatives does not outweigh that of the first, which we use to decrease the residual. Denoting $s \in \{1, \ldots, l\}$ the index for which $|a_s\lambda_s|$ is maximal, bound (4.34) represents a necessary condition to satisfy (4.42). □

Lemma 4.11 shows a very strict bound. Unless $\forall j = 1, \ldots, l : |b_j| \lesssim |\lambda_j \lambda_s|$ (with $b_1, \ldots, b_l$ defined by (4.40)), we have to choose $a_1, \ldots, a_l$ far from the choice (4.41) that completely removes the current residual. Only a tiny part of this residual will be reduced instead, resulting in slow convergence.

4.5.3 Minimization of the residual norm

We had not yet established how the initial guess $a^{(0)}$ for the minimization of (4.31) is chosen. This guess should be chosen in such a way that the terms of the Taylor expansion (4.38) of the new residual do not blow up. To this end, we will calculate $a^{(0)}$ by first minimizing the function

$$f : \mathbb{R}^l \to \mathbb{R} : a' \to \sum_{j=1}^{l} (b_j + a'_j \lambda_j)^2 + \sum_{j=1}^{l} \left(a'_j \|\Delta x_\perp\|\right)^2 + \sum_{i=1}^{l} \sum_{j=1}^{l} (a'_i a'_j)^2 \quad (4.43)$$

with the nonlinear conjugate gradients method. The values $b_1, \ldots, b_l$ are defined by (4.40). Note that we also prevent the possible blow-up of the term $\sum_{j=1}^{l} a_j F_{xx}(\tilde{x}) \Delta x_\perp \phi_j$. The function f is independent of the considered problem, so its minimization should be fast.

This extra minimization requires an initial guess $a'^{(0)}$ as well, which should satisfy the established condition (4.42). We do not know the index s for which $|a_s \lambda_s|$ is maximal in advance. Instead we choose s such that $|b_s|$ is maximal. The part $b_s \phi_s$ of the current residual needs to be reduced the most, so it makes sense to maximize the $a_s \phi_s$ term in Δx, which induces a change of $a_s \lambda_s \phi_s$ to the $b_s \phi_s$ part of the residual. We choose

$$\begin{aligned} a'^{(0)}_s &= -\operatorname{sgn}\left(\frac{b_s}{\lambda_s}\right) \min\left(\left|\frac{b_s}{\lambda_s}\right|, |\lambda_s|, \left|\frac{b_s}{\|\Delta x_\perp\|}\right|\right), \\ \forall j \neq s : a'^{(0)}_j &= -\operatorname{sgn}\left(\frac{b_j}{\lambda_j}\right) \min\left(\left|\frac{b_j}{\lambda_j}\right|, |a'^{(0)}_s|\right). \end{aligned} \quad (4.44)$$

This choice not only implies (4.42), but also prevents the blow-up of the $\sum_{j=1}^{l} \left(a'_j \|\Delta x_\perp\|\right)^2$ term in (4.43) by assuring that

$$\forall j = 1, \ldots, l : |a_j| \|\Delta x_\perp\| \lesssim |b_s|.$$

4.5.4 The preconditioned case

For the preconditioned case we again split the update vector Δx in

$$\Delta x = \Delta x_\perp + \Delta x_\parallel \quad (4.45)$$

$$= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j, \quad (4.46)$$

with $a_j = \langle \Delta x, \tau_j \rangle$ $(\forall j = 1, \ldots, l)$ and $\langle \Delta x_\perp, \tau_j \rangle = 0$ $(\forall j = 1, \ldots, l)$. The vectors $\phi_1, \ldots, \phi_l$ represent the approximate null vectors of $P(\tilde{x}) F_x(\tilde{x})$, with respective eigenvalues $\lambda_1, \ldots, \lambda_l$. The vectors $\tau_1, \ldots, \tau_l$ are defined such that

$P(\tilde{x})\tau_j = \phi_j$ $(\forall j = 1, \ldots, l)$. The part $\Delta x_\perp$ of the update vector is approximated by solving

$$P(\tilde{x})\mathcal{Q}(\tilde{x})F_x(\tilde{x})\Delta x_\perp = -P(\tilde{x})\mathcal{Q}(\tilde{x})F(\tilde{x}), \tag{4.47}$$

with deflated GMRES, where $\mathcal{Q}(\tilde{x})$ is defined by (4.15). To approximate the part $\Delta x_\parallel$ of the update vector, we minimize the function

$$g : \mathbb{R}^l \to \mathbb{R} : a \to \left\| F\left(\tilde{x} + \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j\right)\right\|^2_{P(\tilde{x})} \tag{4.48}$$

with the nonlinear conjugate gradients method. The first and second partial derivatives of g are given by $(\forall i, k = 1, \ldots, l)$

$$\frac{\partial g}{\partial a_i} : \mathbb{R}^l \to \mathbb{R} : a \to 2\langle F(x(a)), F_x(x(a))\phi_i\rangle_{P(\tilde{x})}, \tag{4.49}$$

$$\begin{aligned}\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \to \mathbb{R} : a \to & 2\langle F_x(x(a))\phi_i, F_x(x(a))\phi_k\rangle_{P(\tilde{x})} \\ & + 2\langle F(x(a)), F_{xx}(x(a))\phi_i\phi_k\rangle_{P(\tilde{x})},\end{aligned} \tag{4.50}$$

$$\text{with } x(a) = \tilde{x} + \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j.$$

To create an initial guess for the minimization of (4.48) we again first minimize a different function with nonlinear conjugate gradients. This function is given by

$$f : \mathbb{R}^l \to \mathbb{R} : a' \to \sum_{j=1}^{l}(b_j + a'_j\lambda_j)^2 + \sum_{j=1}^{l}\left(a'_j\|\Delta x_\perp\|_{P(\tilde{x})^{-1}}\right)^2 + \sum_{i=1}^{l}\sum_{j=1}^{l}(a'_i a'_j)^2 \tag{4.51}$$

As an initial guess for the minimization of f, we use the values

$$\begin{aligned} a'^{(0)}_s &= -\operatorname{sgn}\left(\frac{b_s}{\lambda_s}\right)\min\left(\left|\frac{b_s}{\lambda_s}\right|, |\lambda_s|, \left|\frac{b_s}{\|\Delta x_\perp\|_{P(\tilde{x}^{-1})}}\right|\right), \\ \forall j \neq s : a'^{(0)}_j &= -\operatorname{sgn}\left(\frac{b_j}{\lambda_j}\right)\min\left(\left|\frac{b_j}{\lambda_j}\right|, |a'^{(0)}_s|\right). \end{aligned} \tag{4.52}$$

The values $b_1, \ldots, b_l$ are defined by $b_j = \langle F(\tilde{x}), \phi_j\rangle$ $(\forall j = 1, \ldots, l)$, the index s is still chosen such that $|b_s|$ is maximal.

As in the unpreconditioned case, it is possible that the SN method converges slowly. This is again caused by the bound imposed by preventing dominance of the second order derivative in the new residual. Splitting the current residual $F(\tilde{x})$ in

$$\begin{aligned} F(\tilde{x}) &= F_\perp + F_\parallel \\ &= F_\perp + \sum_{j=1}^{l} b_j \tau_j, \end{aligned}$$

with $b_j = \langle F(\tilde{x}), \phi_j \rangle$ $(\forall j = 1, \ldots, l)$ and $\langle F_\perp, \phi_j \rangle = 0$ $(\forall j = 1, \ldots, l)$, we have $\mathcal{Q}(\tilde{x})F(\tilde{x}) = F_\perp$. A similar Taylor expansion as before yields

$$\begin{aligned} & P(\tilde{x})F\left(\tilde{x} + \Delta x_\perp + \Delta x_\parallel\right) \\ \approx \quad & \sum_{j=1}^{l} \left(b_j + a_j\lambda_j\right)\phi_j + \frac{1}{2}P(\tilde{x})F_{xx}(\tilde{x})\Delta x_\perp \Delta x_\perp + \sum_{j=1}^{l} a_j P(\tilde{x})F_{xx}(\tilde{x})\Delta x_\perp \phi_j \\ & + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j P(\tilde{x})F_{xx}(\tilde{x})\phi_i\phi_j + \mathcal{O}(\|\Delta x\|^3). \end{aligned}$$

There is still a trade-off between choosing the values for $a_1, \ldots, a_l$ such that the $F_\parallel = \sum_{j=1}^{l} b_j\phi_j$ part of the current (preconditioned) residual reduces, and preventing dominance of the term $\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j P(\tilde{x})F_{xx}(\tilde{x})\phi_i\phi_j$. Similar bounds as before are derived, resulting in the necessary condition (4.34). Unless $b_1, \ldots, b_l$ are sufficiently small, slow convergence is expected.

4.6 Addition of extra terms to the update vector

4.6.1 Description of the method

Lemma 4.11 illustrates slow convergence for the split Newton method (SN) applied to ill-conditioned problems. This needs to be eliminated. To this end we will introduce an additional part to the update vector. Consider a guess $\tilde{x} \in \mathbb{C}^n$. We first split the application of the second partial derivative to the approximate null vectors $\phi_1, \ldots, \phi_l$ of $F_x(\tilde{x})$ into a part perpendicular to, and a part consisting of these vectors:

$$\begin{aligned} \forall i, j = 1, \ldots, l : F_{xx}(\tilde{x})\phi_i\phi_j &= F_\perp^{(ij)} + F_\parallel^{(ij)} && (4.53) \\ &= F_\perp^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)}\phi_k, && (4.54) \end{aligned}$$

for certain values $c_k^{(ij)} \in \mathbb{R}$ $(i, j, k = 1, \ldots, l)$. Defining the projection operator $\mathcal{Q}(\tilde{x})$ as before (see (4.13)), we have

$$\forall i, j = 1, \ldots, l : \mathcal{Q}(\tilde{x})F_{xx}(\tilde{x})\phi_i\phi_j = F_\perp^{(ij)}.$$

Note that $\langle F_\perp^{(ij)}, \phi_k \rangle = 0$ $(\forall i, j, k = 1, \ldots, l)$.

We now introduce $z_{11}, z_{12}, \ldots, z_{ll} \in \mathbb{C}^n$ as the solution of the linear equations $(\forall i, j = 1, \ldots, l)$

$$\mathcal{Q}(\tilde{x})F_x(\tilde{x})z_{ij} = \mathcal{Q}(\tilde{x})F_{xx}(\tilde{x})\phi_i\phi_j \tag{4.55}$$

such that $\langle z_{ij}, \phi_k \rangle = 0$ $(\forall i, j, k = 1, \ldots, l)$. To further adjust the SN method, an additional part Δx_z is added to the update vector. We consider the vector

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel + \Delta x_z && (4.56) \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} z_{ij}, && (4.57) \end{aligned}$$

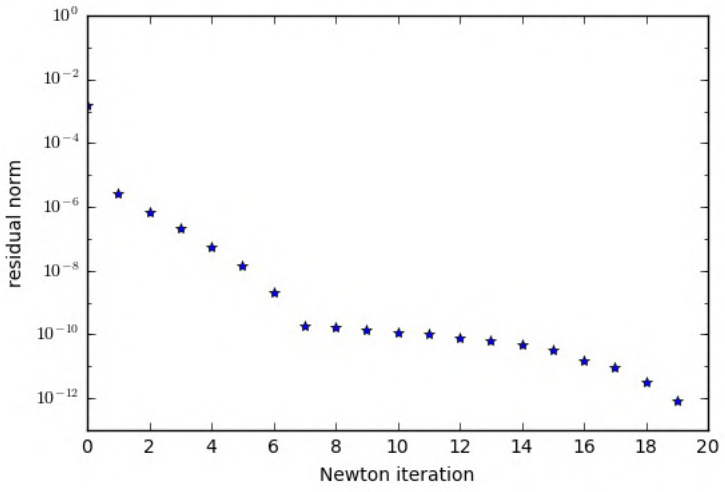


Figure 4.8: Residual plot of example 4.12. The SNE method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-12} after 19 iterations, which is the attainable accuracy.

with $\Delta x_\perp$ defined as before (see (4.30)). The values $a_1, \ldots, a_l$ and $\alpha_{11}, \ldots, \alpha_{ll}$ will be calculated by minimizing the residual norm (see section 4.6.3).

In practice the update part $\Delta x_\perp$ and $z_{11}, \ldots, z_{ll}$ vectors are calculated by solving (4.30), respectively (4.55), with deflated GMRES. Note that $z_{ij} = z_{ji}$ for each $i, j = 1, \ldots, l$.

This new method where the update vector is given by (4.57), will be called the *split Newton method with extra terms* (SNE), and is described by algorithm 4.5 on page 102 in appendix 4.10. In each Newton iteration approximate null vectors are calculated by an eigensolver (algorithm 3.3, page 43). Next, deflated linear systems of the forms (4.30) and (4.55) are solved with GMRES (algorithm 3.4, page 44). The function (4.70) (the preconditioned version of the residual norm) is minimized and the update vector is constructed by (4.57). The guess is updated and the next iteration is started.

4.6.2 Analysis of convergence for ill-conditioned problems

Algorithm 4.5 is applied in example 4.12.

Example 4.12. Consider the same set-up as in example 4.3. Application of the SNE method yields the residual plot in figure 4.8. Convergence (up to a tolerance of 10^{-12}) is reached after 19 Newton iterations. ◇

Contrary to the previous methods where we observed very slow convergence, stagnation, or even divergence, the new method achieves convergence in 19

iterations. The bound discussed in lemma 4.11 does not apply to the SNE method.

Lemma 4.13. By introducing the additional part Δx_z, defined in (4.57), the bound (4.34) discussed in lemma 4.11 is eliminated.

Proof. Application of a third order Taylor expansion to the next residual gives

$$\begin{aligned}
F(\tilde{x}+\Delta x) &= F(\tilde{x}) + F_x(\tilde{x})\Delta x + \frac{1}{2}F_{xx}(\tilde{x})\Delta x\Delta x + \frac{1}{6}F_{xxx}(\tilde{x})\Delta x\Delta x\Delta x \\
&\quad + \mathcal{O}(\|\Delta x\|^4) \\
&= F(\tilde{x}) + F_x(\tilde{x})\Delta x_\perp + F_x(\tilde{x})\Delta x_\parallel + F_x(\tilde{x})\Delta x_z \\
&\quad + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\perp\Delta x_\perp + F_{xx}(\tilde{x})\Delta x_\perp\Delta x_\parallel \\
&\quad + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\parallel\Delta x_\parallel + R(\tilde{x},\Delta x) \\
&= F(\tilde{x}) + F_x(\tilde{x})\Delta x_\perp + \sum_{j=1}^{l} a_j F_x(\tilde{x})\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}F_x(\tilde{x})z_{ij} \\
&\quad + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\perp\Delta x_\perp + \sum_{j=1}^{l} a_j F_{xx}(\tilde{x})\Delta x_\perp\phi_j \\
&\quad + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x})\phi_i\phi_j + R(\tilde{x},\Delta x).
\end{aligned}$$

The remainder $R(\tilde{x},\Delta x)$ is given by

$$\begin{aligned}
R(\tilde{x},\Delta x) &= F_{xx}(\tilde{x})\Delta x_\perp\Delta x_z + F_{xx}(\tilde{x})\Delta x_\parallel\Delta x_z + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_z\Delta x_z \\
&\quad + \frac{1}{6}F_{xxx}(\tilde{x})\Delta x\Delta x\Delta x + \mathcal{O}(\|\Delta x\|^4).
\end{aligned}$$

Splitting $F(\tilde{x})$ as before (see (4.40)) and denoting λ_j the eigenvalue corresponding to the approximate null vector ϕ_j ($\forall j = 1,\ldots,l$), (4.30) and (4.55) imply

$$\begin{aligned}
F(\tilde{x}+\Delta x) = &\sum_{k=1}^{l}\left(b_k + a_k\lambda_k + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}c_k^{(ij)}a_i a_j\right)\phi_k + \frac{1}{2}F_{xx}(\tilde{x})\Delta x_\perp\Delta x_\perp \\
&+ \sum_{j=1}^{l} a_j F_{xx}(\tilde{x})\Delta x_\perp\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{ij} + \frac{1}{2}a_i a_j)F_\perp^{(ij)} + R(\tilde{x},\Delta x).
\end{aligned}$$

Further splitting of the terms

$$\begin{aligned} F_{xx}(\tilde{x})\Delta x_\perp \Delta x_\perp &= F_\perp^{(00)} + F_\parallel^{(00)} \\ &= F_\perp^{(00)} + \sum_{k=1}^{l} c_k^{(00)} \phi_k \end{aligned} \tag{4.58}$$

$$\begin{aligned} \forall j = 1, \ldots, l : F_{xx}(\tilde{x})\Delta x_\perp \phi_j &= F_\perp^{(0j)} + F_\parallel^{(0j)} \\ &= F_\perp^{(0j)} + \sum_{k=1}^{l} c_k^{(0j)} \phi_k \end{aligned} \tag{4.59}$$

eventually yields

$$\begin{aligned} F(\tilde{x} + \Delta x) = &\sum_{k=1}^{l} \left(b_k + a_k \lambda_k + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a_j + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} c_k^{(ij)} a_i a_j \right) \phi_k \\ &+ \frac{1}{2} F_\perp^{(00)} + \sum_{j=1}^{l} a_j F_\perp^{(0j)} + \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_{ij} + \frac{1}{2} a_i a_j) F_\perp^{(ij)} + R(\tilde{x}, \Delta x). \end{aligned}$$

The SN method is equivalent to setting $\alpha_{11} = \cdots = \alpha_{ll} = 0$ in (4.57), in this case the parts $\frac{1}{2} a_i a_j F_\perp^{(ij)} \phi_i \phi_j$ (for $i, j = 1, \ldots, l$) of the second order derivative terms dominate the new residual if $a_1, \ldots, a_l$ don't satisfy (4.42). These are the parts perpendicular to null vector directions. By introducing the $\alpha_{11}, \ldots, \alpha_{ll}$ values, these terms disappear from the Taylor expansion if we choose

$$\forall i, j = 1, \ldots, l : \alpha_{ij} = -\frac{1}{2} a_i a_j. \tag{4.60}$$

The null vector parts $\frac{1}{2} \sum_{k=1}^{l} c_k^{(ij)} \phi_k$ (for $i, j = 1, \ldots, l$) of these derivatives are incorporated in the choice for $a_1, \ldots, a_l$, and will be reduced as well. By adding the part Δx_z (the terms $z_{11}, \ldots, z_{ll}$) to the update vector and choosing the values for $\alpha_{11}, \ldots, \alpha_{ll}$ as in (4.60), it is prevented that the second order derivative terms in the new residual dominate. This dominance lead to bound (4.42), which will be eliminated, or at least strongly reduced, for the new technique. □

The bound of lemma 4.11 is eliminated by a specific choice for the values $\alpha_{11}, \ldots, \alpha_{ll}$ in the additional part Δx_z of the update vector. There is however a similar bound to the values of $a_1, \ldots, a_l$ in (4.57).

Lemma 4.14. The values $a_1, \ldots, a_l$, that appear in the update vector (4.57) of the SNE method, typically satisfy the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt{|\lambda_s|}. \tag{4.61}$$

The index s is chosen such that $|a_s \lambda_s|$ is maximal.

Proof. Consider the same Taylor expansion as in the proof of lemma 4.13. The part

$$\frac{1}{6}F_{xxx}(\tilde{x})\Delta x_{\|}\Delta x_{\|}\Delta x_{\|} = \frac{1}{6}\sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l} a_k a_i a_j F_{xxx}(\tilde{x})\phi_k\phi_i\phi_l \tag{4.62}$$

of the third order derivative $\frac{1}{6}F_{xxx}(\tilde{x})\Delta x\Delta x\Delta x$ will induce bound (4.61). To prevent dominance of this derivative over the first order one, we need to impose

$$\max_{k,i,j=1,\ldots,l} |a_k a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j\lambda_j| \tag{4.63}$$

on $a_1,\ldots,a_l$. Denoting $s \in \{1,\ldots,l\}$ the index for which $|a_s\lambda_s|$ is maximal, (4.61) represents a necessary condition to satisfy this approximate inequality. □

The condition discussed in lemma 4.14 is however a lot less strict than the one in lemma 4.11. Convergence will not be reduced as much as bound (4.42) did for the SN method: by adding an additional part to the update vector, we can choose the values $a_1,\ldots,a_l$ much closer to the optimal ones that reduce the current residual.

Though the SNE method is more reliable, it also requires more computational work: instead of a single linear system that needs to be solved, the SNE method requires the solution of $1+\frac{1}{2}l(l+1)$ different linear systems (one for $\Delta x_\perp$, $\frac{1}{2}l(l+1)$ for the terms $z_{11},\ldots,z_{ll}$ using the relation $z_{ij}=z_{ji}$ $(\forall i,j=1,\ldots,l)$). The minimization of the residual norm with the nonlinear conjugate gradients method, which will be discussed in section 4.6.3, contains $l+\frac{1}{2}l(l+1)$ variables (a_j for $j=1,\ldots,l$ and α_{ij} for $i,j=1,\ldots,l$ with $i\leq j$, since $\alpha_{ij}=\alpha_{ji}$). It will also be slower than was the case in the previous methods. This work needs to be reduced before the solver can be applied in an efficient way. A technique for this purpose will be discussed in section 4.7.

4.6.3 Minimization of the residual norm

After calculation of the vectors $\Delta x_\perp$ and $z_{11},\ldots,z_{ll}$ (with deflated GMRES), the values $a_1,\ldots,a_l$ and $\alpha_{11},\ldots,\alpha_{ll}$ of the update vector (4.57) are calculated by minimizing

$$g:\mathbb{R}^l\times\mathbb{R}^{l\times l}\to\mathbb{R}:(a,\alpha)\to\left\|F\left(\tilde{x}+\Delta x_\perp+\sum_{j=1}^{l}a_j\phi_j+\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}z_{ij}\right)\right\|^2. \tag{4.64}$$

The vectors $\Delta x_\perp$, $\phi_1,\ldots,\phi_l$ and $z_{11},\ldots,z_{ll}$ are kept constant in this minimization. Note that we include $\alpha_{11},\ldots,\alpha_{ll}$ since the choice (4.60) might not be optimal. Due to symmetry α_{ij} and α_{ji} can be treated as the same variables for each $i,j=1,\ldots,l$. The minimization of g is done with the nonlinear conju-

gate gradients method, for which the first partial derivatives ($\forall i, j = 1, \ldots, l$)

$$\begin{aligned}
&\frac{\partial g}{\partial a_i} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a, \alpha) \to 2\langle F(x(a,\alpha)), F_x(x(a,\alpha))\phi_i\rangle, \\
&\frac{\partial g}{\partial \alpha_{ij}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a, \alpha) \to 2\langle F(x(a,\alpha)), F_x(x(a,\alpha))z_{ij}\rangle, \\
&\text{with } x(a,\alpha) = \tilde{x} + \Delta x_\perp + \sum_{j=1}^{l} a_j\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} z_{ij},
\end{aligned} \tag{4.65}$$

and second partial derivatives ($\forall i, j, k, q = 1, \ldots, l$)

$$\begin{aligned}
\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a, \alpha) \to &2\langle F_x(x(a,\alpha))\phi_i, F_x(x(a,\alpha))\phi_k\rangle \\
&+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\phi_i\phi_k\rangle, \\
\frac{\partial^2 g}{\partial a_i \partial \alpha_{jk}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a, \alpha) \to &2\langle F_x(x(a,\alpha))\phi_i, F_x(x(a,\alpha))z_{jk}\rangle \\
&+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\phi_i z_{jk}\rangle, \\
\frac{\partial^2 g}{\partial \alpha_{ij} \partial \alpha_{kq}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a, \alpha) \to &2\langle F_x(x(a,\alpha))z_{ij}, F_x(x(a,\alpha))z_{kq}\rangle \\
&+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))z_{ij}z_{kq}\rangle
\end{aligned} \tag{4.66}$$

are required.

The initial guess $a^{(0)}$ for a to use in the nonlinear conjugate gradients method should be chosen such that the terms in the Taylor expansion of the new residual do not blow up. The initial guess for α is then calculated by (4.60). To calculate $a^{(0)}$, we again first minimize a different function, given by

$$\begin{aligned}
f :&\mathbb{R}^l \to \mathbb{R} : \\
&a' \to \sum_{k=1}^{l} \left(b_k + a'_k\lambda_k + \frac{1}{2}c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a'_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} c_k^{(ij)} a'_i a'_j \right)^2 \\
&\quad + \sum_{j=1}^{l} \left(a'_j \|\Delta x_\perp\|\right)^2 + \sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l} (a'_k a'_i a'_j)^2.
\end{aligned} \tag{4.67}$$

The values $b_1, \ldots, b_l$ and $c_k^{(ij)}$ (for $i, j, k = 1, \ldots, l$) are defined as in (4.40), (4.54), (4.58) and (4.59). The function (4.67) not only prevents the possible blow-up of (4.62), but also of other (second and third) order derivatives since the following holds ($\forall i, j, k, q = 1, \ldots, l$):

$$\begin{aligned}
&a_j F_\perp^{(0j)} = \mathcal{O}(a_j\|\Delta x_\perp\|), && \alpha_{ij} F_{xx}(\tilde{x})\Delta x_\perp z_{ij} = \mathcal{O}(a_i a_j \|\Delta x_\perp\|), \\
&a_k \alpha_{ij} F_{xx}(\tilde{x})\phi_k z_{ij} = \mathcal{O}(a_k a_i a_j), && \alpha_{qk}\alpha_{ij} F_{xx}(\tilde{x}) z_{qk} z_{ij} = \mathcal{O}(a_q a_k a_i a_j),
\end{aligned}$$

where we used $\alpha_{ij} = \mathcal{O}(a_i a_j)$ ($\forall i, j = 1, \ldots, l$) by (4.60). For parts of the third order derivative, different from (4.62), a similar result is valid.

To minimize (4.67) with nonlinear conjugate gradients, we need an initial guess $a'^{(0)}$. We again choose the index s such that $|b_s|$ is maximal, and use

$$\begin{aligned} a_s'^{(0)} &= -\operatorname{sgn}\left(\frac{b_s}{\lambda_s}\right)\min\left(\left|\frac{b_s}{\lambda_s}\right|, \sqrt{|\lambda_s|}, \left|\frac{b_s}{\|\Delta x_\perp\|}\right|\right), \\ \forall j \neq s : a_j'^{(0)} &= -\operatorname{sgn}\left(\frac{b_j}{\lambda_j}\right)\min\left(\left|\frac{b_j}{\lambda_j}\right|, |a_s'^{(0)}|\right) \end{aligned} \tag{4.68}$$

as an initial guess to minimize (4.67). This choice satisfies bound (4.61), but also

$$\forall j = 1, \ldots, l : |a_j|\|\Delta x_\perp\| \lesssim |b_s|,$$

preventing the possible blow-up of other parts in (4.67).

4.6.4 The preconditioned case

The required adjustments for preconditioning are similar as before. The update vector is still split as in (4.57), where $\phi_1, \ldots, \phi_l$ this time represent the approximate null vectors of $P(\tilde{x})F_x(\tilde{x})$. Denoting $\tau_1, \ldots, \tau_l$ such that $P(\tilde{x})\tau_j = \phi_j$ $(\forall j = 1, \ldots, l)$, the second order derivatives are split as

$$\begin{aligned} F_{xx}(\tilde{x})\Delta x\Delta x &= F_\perp^{(00)} + F_\parallel^{(00)} \\ &= F_\perp^{(00)} + \sum_{k=1}^{l} c_k^{(00)}\tau_k, \\ \forall j = 1, \ldots, l : F_{xx}(\tilde{x})\Delta x\phi_j &= F_\perp^{(0j)} + F_\parallel^{(0j)} \\ &= F_\perp^{(0j)} + \sum_{k=1}^{l} c_k^{(0j)}\tau_k, \\ \forall i, j = 1, \ldots, l : F_{xx}(\tilde{x})\phi_i\phi_j &= F_\perp^{(ij)} + F_\parallel^{(ij)} \\ &= F_\perp^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)}\tau_k, \end{aligned}$$

with $\langle F_\perp^{(ij)}, \phi_k\rangle = 0$ $(\forall i, j, k = 1, \ldots, l)$. The values for $c_k^{(ij)}$ and b_k (in (4.40)) are calculated as

$$\begin{aligned} &\forall k = 1, \ldots, l : b_k = \langle F(\tilde{x}), \phi_k\rangle, \\ &\forall k = 1, \ldots, l : c_k^{(00)} = \langle F_{xx}(\tilde{x})\Delta x_\perp\Delta x_\perp, \phi_k\rangle, \\ &\forall j, k = 1, \ldots, l : c_k^{(0j)} = \langle F_{xx}(\tilde{x})\Delta x_\perp\phi_j, \phi_k\rangle, \\ &\forall i, j, k = 1, \ldots, l : c_k^{(ij)} = \langle F_{xx}(\tilde{x})\phi_i\phi_j, \phi_k\rangle. \end{aligned}$$

The terms $z_{11}, \ldots, z_{ll}$ of the additional part Δx_z of the update vector satisfy $\langle z_{ij}, \tau_k\rangle = 0$ $(\forall i, j, k = 1, \ldots, l)$, and are calculated by solving $(\forall i, j = 1, \ldots, l)$

$$P(\tilde{x})\mathcal{Q}(\tilde{x})F_x(\tilde{x})z_{ij} = P(\tilde{x})\mathcal{Q}(\tilde{x})F_{xx}(\tilde{x})\phi_i\phi_j \tag{4.69}$$

with deflated GMRES, where $\mathcal{Q}(\tilde{x})$ is defined as in (4.15). The $\Delta x_\perp$ part is solved by application of deflated GMRES to (4.47).

Application of a third order Taylor expansion of the term $F(\tilde{x}+\Delta x)$ to $P(\tilde{x})F(\tilde{x}+\Delta x)$ yields analogue results as in the unpreconditioned case. The influence of second order derivatives in this expansion is again reduced by choosing $\alpha_{11},\ldots,\alpha_{ll}$ as in (4.60), resulting in improved convergence. In practice $a_1,\ldots,a_l$ and $\alpha_{11},\ldots,\alpha_{ll}$ are calculated by minimizing the function

$$\begin{aligned} g :&\mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ &(a,\alpha) \to \left\| F\left(\tilde{x} + \Delta x_\perp + \sum_{j=1}^{l} a_j\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}z_{ij}\right)\right\|^2_{P(\tilde{x})}. \end{aligned} \tag{4.70}$$

with the nonlinear conjugate gradients method. The first partial derivatives are given by $(\forall i,j=1,\ldots,l)$

$$\begin{aligned} &\frac{\partial g}{\partial a_i} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a,\alpha) \to 2\langle F(x(a,\alpha)), F_x(x(a,\alpha))\phi_i\rangle_{P(\tilde{x})}, \\ &\frac{\partial g}{\partial \alpha_{ij}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a,\alpha) \to 2\langle F(x(a,\alpha)), F_x(x(a,\alpha))z_{ij}\rangle_{P(\tilde{x})}, \\ &\text{with } x(a,\alpha) = \tilde{x} + \Delta x_\perp + \sum_{j=1}^{l} a_j\phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}z_{ij}, \end{aligned} \tag{4.71}$$

the second by $(\forall i,j,k,q=1,\ldots,l)$

$$\begin{aligned} \frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a,\alpha) \to &2\langle F_x(x(a,\alpha))\phi_i, F_x(x(a,\alpha))\phi_k\rangle_{P(\tilde{x})} \\ &+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\phi_i\phi_k\rangle_{P(\tilde{x})}, \\ \frac{\partial^2 g}{\partial a_i \partial \alpha_{jk}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : (a,\alpha) \to &2\langle F_x(x(a,\alpha))\phi_i, F_x(x(a,\alpha))z_{jk}\rangle_{P(\tilde{x})} \\ &+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\phi_i z_{jk}\rangle_{P(\tilde{x})}, \\ \frac{\partial^2 g}{\partial \alpha_{ij} \partial \alpha_{kq}} :&\mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ &(a,\alpha) \to 2\langle F_x(x(a,\alpha))z_{ij}, F_x(x(a,\alpha))z_{kq}\rangle_{P(\tilde{x})} \\ &+ 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))z_{ij}z_{kq}\rangle_{P(\tilde{x})}. \end{aligned} \tag{4.72}$$

To construct an initial guess for the minimization of (4.70), we first minimize the function

$$\begin{aligned} f :&\mathbb{R}^l \to \mathbb{R} : \\ &a' \to \sum_{k=1}^{l}\left(b_k + a'_k\lambda_k + \frac{1}{2}c_k^{(00)} + \sum_{j=1}^{l}c_k^{(0j)}a'_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}c_k^{(ij)}a'_ia'_j\right)^2 \\ &+ \sum_{j=1}^{l}\left(a'_j\|\Delta x_\perp\|_{P(\tilde{x})^{-1}}\right)^2 + \sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l}(a'_ka'_ia'_j)^2 \end{aligned} \tag{4.73}$$

with nonlinear conjugate gradients. As an initial guess for this extra minimization, we use the values

$$\begin{aligned} a_s'^{(0)} &= -\operatorname{sgn}\left(\frac{b_s}{\lambda_s}\right)\min\left(\left|\frac{b_s}{\lambda_s}\right|, \sqrt{|\lambda_s|}, \left|\frac{b_s}{\|\Delta x_\perp\|_{P(\tilde{x})^{-1}}}\right|\right), \\ \forall j \neq s : a_j'^{(0)} &= -\operatorname{sgn}\left(\frac{b_j}{\lambda_j}\right)\min\left(\left|\frac{b_j}{\lambda_j}\right|, |a_s'^{(0)}|\right), \end{aligned} \tag{4.74}$$

with s again the index for which $|b_s|$ is maximal. The (approximate) arguments for the minimum of (4.73) are used as an initial guess for the $a_1, \ldots, a_l$ values that minimize (4.70). Initial guesses for $\alpha_{11}, \ldots, \alpha_{ll}$ are calculated by application of (4.60).

4.7 Reduction of the extra terms in presence of a dominant update direction

In the previous section an alternative Newton method (SNE) was derived that is capable of solving ill-conditioned problems. Though convergence is usually achieved in an acceptable amount of iterations, the number of linear systems that need to be solved grows exponentially with the amount of approximate null vectors. This strongly increases computational work, making the SNE method less practical. In the current section we will try to reduce the number of linear systems.

4.7.1 Creation of a new base for the approximate kernel

Consider two consecutive guesses $\tilde{x}^{(m)}$ and $\tilde{x}^{(m+1)}$ of the SNE method, we have (see (4.57))

$$\begin{aligned} \tilde{x}^{(m+1)} &= \tilde{x}^{(m)} + \Delta x^{(m)} = \tilde{x}^{(m)} + \Delta x_\perp^{(m)} + \Delta x_\parallel^{(m)} + \Delta x_z^{(m)} \\ &= \tilde{x}^{(m)} + \Delta x_\perp^{(m)} + \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)} + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij}^{(m)} z_{ij}^{(m)}. \end{aligned} \tag{4.75}$$

The vectors $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$ represent the approximate null vectors of $F_x(\tilde{x}^{(m)})$ with $\|\phi_1^{(m)}\| = \cdots = \|\phi_l^{(m)}\| = 1$, their respective eigenvalues are denoted by $\lambda_1^{(m)}, \ldots, \lambda_l^{(m)}$. The vectors $\Delta x_\perp^{(m)}, z_{11}^{(m)}, \ldots, z_{ll}^{(m)}$ are perpendicular to these. Together with the values $a_1^{(m)}, \ldots, a_l^{(m)}$ and $\alpha_{11}^{(m)}, \ldots, \alpha_{ll}^{(m)}$, these vectors are calculated as described in section 4.6 (see (4.30) and (4.55)).

Denote $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ the approximate null vectors of $F_x(\tilde{x}^{(m+1)})$ such that $\|\phi_1^{(m+1)}\| = \cdots = \|\phi_l^{(m+1)}\| = 1$. Since $\tilde{x}^{(m)}$ and $\tilde{x}^{(m+1)}$ are close we can choose these vectors such that

$$\forall j = 1, \ldots, l : \phi_j^{(m+1)} \approx \phi_j^{(m)}.$$

The eigenvalues corresponding to $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ are denoted respectively by $\lambda_1^{(m+1)}, \ldots, \lambda_l^{(m+1)}$.

We will first create a new base $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ for the space spanned by $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$. Define

$$\begin{aligned} \mathring{\phi} &= \sum_{j=1}^{l} \langle \phi_j^{(m+1)}, \Delta x_{\|}^{(m)} \rangle \phi_j^{(m+1)}, \\ \breve{\phi}_1^{(m+1)} &= \|\mathring{\phi}\|^{-1} \mathring{\phi}, \end{aligned} \tag{4.76}$$

with $\Delta x_{\|}^{(m)} = \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)}$, and choose the remaining vectors $\breve{\phi}_2^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ as linear combinations of $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ such that

$$\forall i, j = 1, \ldots, l : \langle \breve{\phi}_i^{(m+1)}, \breve{\phi}_j^{(m+1)} \rangle = \delta_{ij}.$$

Define the matrix D such that

$$\begin{pmatrix} \breve{\phi}_1^{(m+1)} & \breve{\phi}_2^{(m+1)} & \ldots & \breve{\phi}_l^{(m+1)} \end{pmatrix} = \begin{pmatrix} \phi_1^{(m+1)} & \phi_2^{(m+1)} & \ldots & \phi_l^{(m+1)} \end{pmatrix} D. \tag{4.77}$$

Note that D is a unitary matrix, implying $D^T = D^{-1}$. The elements of D are explicitly given by (using orthonormality of $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$)

$$\forall i, j = 1, \ldots, l : D_{ij} = \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m+1)} \rangle$$

and typically satisfy $|D_{ij}| \lesssim 1$ $(\forall i, j = 1, \ldots, l)$.

Similar as before we define $\mathcal{Q}(\tilde{x}^{(m+1)})$ as

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}) : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K \langle K, K \rangle^{-1} \langle K, y \rangle \\ &\text{with } K = \begin{pmatrix} \breve{\phi}_1^{(m+1)} & \breve{\phi}_2^{(m+1)} & \ldots & \breve{\phi}_l^{(m+1)} \end{pmatrix}, \end{aligned}$$

calculate $\Delta x_{\perp}^{(m+1)}$ by solving (4.30) and split the terms $(\forall i, j = 1, \ldots, l)$

$$F(\tilde{x}^{(m+1)}) = F_{\perp} + F_{\|} = F_{\perp} + \sum_{k=1}^{l} b_k \breve{\phi}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)}) \Delta x_{\perp}^{(m+1)} \Delta x_{\perp}^{(m+1)} = F_{\perp}^{(00)} + F_{\|}^{(00)} = F_{\perp}^{(00)} + \sum_{k=1}^{l} c_k^{(00)} \breve{\phi}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)}) \Delta x_{\perp}^{(m+1)} \breve{\phi}_j^{(m+1)} = F_{\perp}^{(0j)} + F_{\|}^{(0j)} = F_{\perp}^{(0j)} + \sum_{k=1}^{l} c_k^{(0j)} \breve{\phi}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)} = F_{\perp}^{(ij)} + F_{\|}^{(ij)} = F_{\perp}^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)} \breve{\phi}_k^{(m+1)}$$

with $F_{\perp}$, $F_{\perp}^{(00)}$, $F_{\perp}^{(0j)}$ and $F_{\perp}^{(ij)}$ orthogonal to $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ $(\forall i, j = 1, \ldots, l)$.

For a normal step of the split Newton method with extra terms (SNE), we would need to calculate $z_{11}^{(m+1)}, \ldots, z_{ll}^{(m+1)}$ (orthogonal to $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$) by solving $(\forall i, j = 1, \ldots, l)$

$$\mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m+1)}) z_{ij}^{(m+1)} = \mathcal{Q}(\tilde{x}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)} \quad (4.78)$$

and calculate the update vector as

$$\begin{aligned} \Delta x^{(m+1)} &= \Delta x_\perp^{(m+1)} + \Delta x_\parallel^{(m+1)} + \Delta x_z^{(m+1)} \\ &= \Delta x_\perp^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_{ij} z_{ij}^{(m+1)}, \end{aligned}$$

with $a_1, \ldots, a_l$ and $\alpha_{11}, \ldots, \alpha_{ll}$ to be determined by minimizing the norm of the new residual.

4.7.2 Elimination of less important terms in presence of a dominant update direction

An update performed by the SNE method requires an additional $\frac{1}{2}l(l+1)$ linear systems to be solved. We will first reduce this amount to l, by eliminating less important terms. We first make an assumption.

Assumption 4.15. The part $\Delta x_\parallel^{(m+1)}$ of the update vector $\Delta x^{(m+1)}$ in the null vector directions is dominated by $\breve{\phi}_1^{(m+1)}$, defined in (4.76). This implies

$$\forall i = 1, \ldots, l : i \neq 1 \Rightarrow |a_i| \ll |a_1| \quad (4.79)$$

in the update vector.

If the approximate null vector part of the current residual $F(\tilde{x}^{(m+1)})$ approximates that of the previous one $F(\tilde{x}^{(m)})$, which is expected when the values $a_1^{(m)}, \ldots, a_l^{(m)}$ used in the update from $\tilde{x}^{(m)}$ to $\tilde{x}^{(m+1)}$ do not satisfy (4.41) (due to bounds induced by the third derivative), we expect the approximate null vector part of the current update vector $\Delta x^{(m+1)}$ to approximate that of the previous one $\Delta x^{(m)}$ as well. In this case assumption 4.15 typically holds: the vector $\breve{\phi}_1^{(m+1)}$ was created to be similar to the approximate null vector part of the previous iteration (see (4.76)). Note that the assumption will not hold for each Newton iteration, this is further discussed in section 4.8. The remainder of the derivation in the current section is based on assumption 4.15.

The parts $z_{11}^{(m+1)}, \ldots, z_{ll}^{(m+1)}$ were introduced to the update vector to prevent dominance of 2nd order partial derivatives $a_i a_j F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)}$ $(\forall i, j = 1, \ldots, l)$ in the Taylor expansion of the new residual. If assumption 4.15 holds, the most dominant term between these derivatives is given by $a_1^2 F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_1^{(m+1)}$. Terms of the form $a_1 a_j F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)}$ (with $j = 2, \ldots, l$) are less dominant, but should not be dismissed. For $i \neq 1$ and $j \neq 1$, the influence of the $a_i a_j F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)}$ term will be small. Reducing these terms by adding $z_{ij}^{(m+1)}$ parts is not necessary anymore.

We will eliminate the $z_{ij}^{(m+1)}$ parts for $i \neq 1$ and $j \neq 1$, leading to a new update vector given by

$$\begin{aligned} \Delta x^{(m+1)} &= \Delta x_{\perp}^{(m+1)} + \Delta x_{\parallel}^{(m+1)} + \Delta x_z'^{(m+1)} && (4.80) \\ &= \Delta x_{\perp}^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} + \sum_{j=1}^{l} \alpha_j z_j^{(m+1)}, && (4.81) \end{aligned}$$

where we denoted $z_{1j}^{(m+1)}$ as $z_j^{(m+1)}$ for $j = 1, \ldots, l$. Only l additional linear systems of the form (4.78) need to be solved.

Lemma 4.16. Under assumption 4.15, the bound discussed in lemma 4.11 is eliminated by the additional part $\Delta x_z'^{(m+1)}$ defined in (4.81).

Proof. Using the update vector defined in (4.81), the third order Taylor expansion of the new residual becomes

$$\begin{aligned} & F(\tilde{x}^{(m+1)} + \Delta x^{(m+1)}) \\ &= \sum_{k=1}^{l} \left(b_k + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a_j + \frac{1}{2} c_k^{(11)} a_1^2 + \sum_{j=2}^{l} c_k^{(1j)} a_1 a_j \right) \breve{\phi}_k^{(m+1)} \\ &\quad + \sum_{j=1}^{l} a_j F_x(\tilde{x}^{(m+1)}) \breve{\phi}_j^{(m+1)} + \frac{1}{2} F_{\perp}^{(00)} + \sum_{j=1}^{l} a_j F_{\perp}^{(0j)} + \left(\alpha_1 + \frac{1}{2} a_1^2 \right) F_{\perp}^{(11)} \\ &\quad + 2 \sum_{j=2}^{l} \left(\alpha_j + \frac{1}{2} a_1 a_j \right) F_{\perp}^{(1j)} + \frac{1}{2} \sum_{i=2}^{l} \sum_{j=2}^{l} a_i a_j F_{\perp}^{(ij)} + R(\tilde{x}^{(m+1)}, \Delta x^{(m+1)}), \end{aligned}$$

with the remainder $R(\tilde{x}^{(m+1)}, \Delta x^{(m+1)})$ given by

$$\begin{aligned} & R(\tilde{x}^{(m+1)}, \Delta x^{(m+1)}) \\ &= F_{xx}(\tilde{x}^{(m+1)}) \Delta x_{\perp}^{(m+1)} \Delta x_z'^{(m+1)} + F_{xx}(\tilde{x}^{(m+1)}) \Delta x_{\parallel}^{(m+1)} \Delta x_z'^{(m+1)} \\ &\quad + \frac{1}{2} F_{xx}(\tilde{x}^{(m+1)}) \Delta x_z'^{(m+1)} \Delta x_z'^{(m+1)} \\ &\quad + \frac{1}{6} F_{xxx}(\tilde{x}^{(m+1)}) \Delta x^{(m+1)} \Delta x^{(m+1)} \Delta x^{(m+1)} + \mathcal{O}(\|\Delta x^{(m+1)}\|^4). \end{aligned}$$

Denote $C_{kj} = \sum_{i=1}^{l} D_{ij} \lambda_i^{(m+1)} D_{ik}$ for $k, j = 1, \ldots, l$ (note that $C_{kj} = C_{jk}$). We have

$$\begin{aligned} \sum_{j=1}^{l} a_j F_x(\tilde{x}^{(m+1)}) \breve{\phi}_j^{(m+1)} &= \sum_{j=1}^{l} \sum_{i=1}^{l} a_j D_{ij} F_x(\tilde{x}^{(m+1)}) \phi_i^{(m+1)} \\ = \sum_{j=1}^{l} \sum_{i=1}^{l} a_j D_{ij} \lambda_i^{(m+1)} \phi_i^{(m+1)} &= \sum_{j=1}^{l} \sum_{i=1}^{l} \sum_{k=1}^{l} a_j D_{ij} \lambda_i^{(m+1)} D_{ki}^{-1} \breve{\phi}_k^{(m+1)} \\ = \sum_{j=1}^{l} \sum_{i=1}^{l} \sum_{k=1}^{l} a_j D_{ij} \lambda_i^{(m+1)} D_{ik} \breve{\phi}_k^{(m+1)} &= \sum_{j=1}^{l} \sum_{k=1}^{l} a_j C_{kj} \breve{\phi}_k^{(m+1)}, \end{aligned}$$

where we used the fact that D is unitary. Substitution in the Taylor expansion eventually yields

$$\begin{aligned}
&F(\tilde{x}^{(m+1)} + \Delta x)^{(m+1)} \\
&= \sum_{k=1}^{l} \left(b_k + \sum_{j=1}^{l} a_j C_{kj} + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a_j \right. \\
&\qquad \left. + \frac{1}{2} c_k^{(11)} a_1^2 + \sum_{j=2}^{l} c_k^{(1j)} a_1 a_j \right) \breve{\phi}_k^{(m+1)} \\
&\quad + \frac{1}{2} F_\perp^{(00)} + \sum_{j=1}^{l} a_j F_\perp^{(0j)} + \left(\alpha_1 + \frac{1}{2} a_1^2 \right) F_\perp^{(11)} \\
&\quad + 2 \sum_{j=2}^{l} \left(\alpha_j + \frac{1}{2} a_1 a_j \right) F_\perp^{(1j)} + \frac{1}{2} \sum_{i=2}^{l} \sum_{j=2}^{l} a_i a_j F_\perp^{(ij)} + R(\tilde{x}^{(m+1)}, \Delta x^{(m+1)}).
\end{aligned} \tag{4.82}$$

The $a_i a_j F_\perp^{(ij)}$ terms (for $i, j = 1, \ldots, l$) induced the bound of lemma 4.11. Under assumption 4.15, the influence of these terms does not need to be eliminated if $i \neq 1$ and $j \neq 1$. For the other indices dominance of these terms is prevented by setting

$$\forall j = 1, \ldots, l : \alpha_j = -\frac{1}{2} a_1 a_j. \tag{4.83}$$

□

The Taylor expansion (4.82) will also be used in section 4.7.4 to derive initial guesses for $a_1, \ldots, a_l$, to use in the nonlinear conjugate gradients method.

4.7.3 Replacement of the additional update part based on the previous Newton iteration

We already reduced the amount of additional linear systems to l, the number of approximate null vectors. By replacing the terms $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ in the part $\Delta x_z'^{(m+1)}$ of the update vector we will reduce this amount even further. These terms were calculated by solving linear systems ($\forall j = 1, \ldots, l$)

$$\mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m+1)}) z_j^{(m+1)} = \mathcal{Q}(\tilde{x}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)}. \tag{4.84}$$

Before continuing, we make a second assumption.

Assumption 4.17. The part $\Delta x_\perp^{(m)}$, perpendicular to null vectors of $F_x(\tilde{x}^{(m)})$, of the previous update vector $\Delta x^{(m)}$ (see (4.75)) is negligible compared to the part $\Delta x_\parallel^{(m)}$, consisting of these null vectors. This implies

$$\|\Delta x_\perp^{(m)}\| \ll \max_{j=1,\ldots,l} |a_j^{(m)}|. \tag{4.85}$$

Note that assumption 4.17 typically only holds in later Newton iterations, when the residual is dominated by approximate null vector parts. In this case only a small update is performed in directions perpendicular to these vectors. If the assumption is not valid, the further derivation in the current section cannot be made. For this case the split Newton method (without extra terms) however yields decent updates, the part perpendicular to approximate null vectors still decreases quadratically for this method. The slowdown in convergence happens when the residual is dominated by approximate null vectors, for such iterations assumption 4.17 does hold. We can always perform updates with the split Newton method until the assumption is valid.

Lemma 4.18. Under assumptions 4.15 and 4.17, the terms $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ can approximately be solved from the linear systems $(\forall j = 1, \ldots, l)$

$$\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)})z_j^{(m+1)} \approx -\gamma^{-1}\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m)})\rho_j \tag{4.86}$$

with $\gamma = \| \sum_{j=1}^{l} a_j^{(m)}\phi_j^{(m)} \|$ and $\rho_1, \ldots, \rho_l$ defined by

$$\forall j = 1, \ldots, l : \rho_j = \breve{\phi}_j^{(m+1)} - \sum_{i=1}^{l} \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m)} \rangle \phi_i^{(m)}. \tag{4.87}$$

Proof. Choose $j \in \{1, \ldots, l\}$. Application of a finite difference method gives the approximation

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)})F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_1^{(m+1)}\breve{\phi}_j^{(m+1)} \\ &\quad\approx\quad \varepsilon^{-1}\mathcal{Q}(\tilde{x}^{(m+1)})\left(F_x(\tilde{x}^{(m+1)})\breve{\phi}_j^{(m+1)} - F_x(\tilde{x}^{(m+1)} - \varepsilon\breve{\phi}_1^{(m+1)})\breve{\phi}_j^{(m+1)}\right) \end{aligned} \tag{4.88}$$

for small $\varepsilon \in \mathbb{R}_0^+$. We will choose ε as $\gamma = \| \sum_{j=1}^{l} a_j^{(m)}\phi_j^{(m)} \|$, the norm of the approximate null vector part $\Delta x_{\|}^{(m)}$ of the previous update $\Delta x^{(m)}$. (4.88) becomes

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)})F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_1^{(m+1)}\breve{\phi}_j^{(m+1)} \\ &\qquad\approx\quad \gamma^{-1}\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)})\breve{\phi}_j^{(m+1)} \\ &\qquad\quad\ - \gamma^{-1}\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)} - \gamma\breve{\phi}_1^{(m+1)})\breve{\phi}_j^{(m+1)}. \end{aligned} \tag{4.89}$$

Since $\mathcal{Q}(\tilde{x}^{(m+1)})$ represents the projection perpendicular to eigenvectors of $F_x(\tilde{x}^{(m+1)})$, these two linear operators commute. By the definition of $\mathcal{Q}(\tilde{x}^{(m+1)})$, we further have $\mathcal{Q}(\tilde{x}^{(m+1)})\breve{\phi}_j^{(m+1)} = 0$. Application to (4.89) yields

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)})F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_1^{(m+1)}\breve{\phi}_j^{(m+1)} \\ &\qquad\approx\quad -\gamma^{-1}\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)} - \gamma\breve{\phi}_1^{(m+1)})\breve{\phi}_j^{(m+1)}. \end{aligned}$$

By its definition, the vector $\breve{\phi}_1^{(m+1)}$ approximates the null vector part $\Delta x_{\|}^{(m)}$ of the previous update $\Delta x^{(m)}$. Other parts of $\Delta x^{(m)}$ are negligible compared

to this part under assumption 4.17, and since the order of the introduced extra terms $\alpha_{ik}^{(m)} z_{ik}^{(m)}$ is approximately $|a_i^{(m)} a_k^{(m)}|$ ($\ll |a_i^{(m)}|$ since typically $|a_k^{(m)}| \ll 1$) for each $i, k = 1, \ldots, l$. Using $\breve{\phi}_1^{(m+1)} \approx \Delta x_{\|}^{(m)} \approx \Delta x^{(m)}$, we arrive at the approximation

$$\mathcal{Q}(\tilde{x}^{(m+1)}) F_{xx} \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)} \approx -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \breve{\phi}_j^{(m+1)}. \tag{4.90}$$

We now split the vector $\breve{\phi}_j^{(m+1)}$ as

$$\breve{\phi}_j^{(m+1)} = \sum_{i=1}^{l} \zeta_{ij} \phi_i^{(m)} + \rho_j \tag{4.91}$$

with $\zeta_{ij} = \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m)} \rangle$ ($\forall i = 1, \ldots, l$) and ρ_j defined by (4.87), we have $\langle \rho_j, \phi_i^{(m)} \rangle = 0$ ($\forall i = 1, \ldots, l$). Typically $|\zeta_{jj}| \approx 1$, $|\zeta_{ij}| \approx 0$ for $i \neq j$. We have

$$\begin{aligned}
&\mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \breve{\phi}_j^{(m+1)} \\
&\qquad = \sum_{i=1}^{l} \zeta_{ij} \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \phi_i^{(m)} + \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \rho_j \\
&\qquad = \sum_{i=1}^{l} \zeta_{ij} \lambda_i^{(m)} \mathcal{Q}(\tilde{x}^{(m+1)}) \phi_i^{(m)} + \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \rho_j
\end{aligned}$$

Since $\lambda_i^{(m)} \approx 0$ and $\mathcal{Q}(\tilde{x}^{(m+1)}) \phi_i^{(m)}$ is perpendicular to $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ for each $i = 1, \ldots, l$, the first term can be neglected:

$$\mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \breve{\phi}_j^{(m+1)} \approx \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \rho_j.$$

Substitution in (4.90) yields

$$\mathcal{Q}(\tilde{x}^{(m+1)}) F_{xx} \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)} \approx -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}) F_x(\tilde{x}^{(m)}) \rho_j.$$

The statement now follows by substituting this approximation in the linear system (4.84). □

Lemma 4.18 presents alternative linear systems to solve for the vectors $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$. Instead of solving these systems, we will simply replace $z_j^{(m+1)}$ by $-\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}) \rho_j$ ($\forall j = 1, \ldots, l$). If $\tilde{x}^{(m+1)} \approx \tilde{x}^{(m)}$, we indeed have $z_j^{(m+1)} \approx -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}) \rho_j$ ($\forall j = 1, \ldots, l$), since both vectors are perpendicular to the approximate null vectors $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ of $F_x(\tilde{x}^{(m+1)})$. The replacement of $z_j^{(m+1)}$ is done for every $j \in \{1, \ldots, l\}$. The eventual update vector we consider becomes

$$\Delta x^{(m+1)} = \Delta x_{\perp}^{(m+1)} + \Delta x_{\|}^{(m+1)} + \Delta x_{\rho}^{(m+1)} \tag{4.92}$$

$$= \Delta x_{\perp}^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho_j', \tag{4.93}$$

with $\rho'_j = \mathcal{Q}(\tilde{x}^{(m+1)})\rho_j$ $(\forall j = 1, \ldots, l)$. To create this update vector, no additional linear systems need to be solved. Instead the terms $\rho'_1, \ldots, \rho'_l$ need to be calculated, this requires the approximate null vectors of $F_x(\tilde{x}^{(m)})$. These vectors were, however, already calculated to use in the previous iteration. No additional use of the eigensolver is required either.

4.7.4 Minimization of the residual norm

The values for $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ in (4.93) have yet to be determined. Similar to previous methods, this is done by minimizing the norm of the new residual, given by the function

$$g : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} :$$
$$(a, \alpha) \to \left\| F\left(\tilde{x}^{(m+1)} + \Delta x_\perp^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho'_j \right) \right\|^2, \tag{4.94}$$

with the nonlinear conjugate gradients method. The first partial derivatives of g are given by $(\forall i = 1, \ldots, l)$

$$\begin{aligned}
&\frac{\partial g}{\partial a_i} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (a, \alpha) \to 2\langle F(x(a,\alpha)), F_x(x(a,\alpha))\breve{\phi}_i^{(m+1)} \rangle, \\
&\frac{\partial g}{\partial \alpha_i} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (a, \alpha) \to -2\gamma^{-1}\langle F(x(a,\alpha)), F_x(x(a,\alpha))\rho'_i \rangle, \\
&\text{with } x(a,\alpha) = \tilde{x}^{(m+1)} + \Delta x_\perp^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho'_j,
\end{aligned} \tag{4.95}$$

the second partial derivatives by $(\forall i, k = 1, \ldots, l)$

$$\begin{aligned}
\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (a, \alpha) \to & 2\langle F_x(x(a,\alpha))\breve{\phi}_i^{(m+1)}, F_x(x(a,\alpha))\breve{\phi}_k^{(m+1)} \rangle \\
& + 2\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\breve{\phi}_i^{(m+1)}\breve{\phi}_k^{(m+1)} \rangle, \\
\frac{\partial^2 g}{\partial a_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (a, \alpha) \to & -2\gamma^{-1}\langle F_x(x(a,\alpha))\breve{\phi}_i^{(m+1)}, F_x(x(a,\alpha))\rho'_k \rangle \\
& - 2\gamma^{-1}\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\breve{\phi}_i^{(m+1)}\rho'_k \rangle, \\
\frac{\partial^2 g}{\partial \alpha_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (a, \alpha) \to & 2\gamma^{-2}\langle F_x(x(a,\alpha))\rho'_i, F_x(x(a,\alpha))\rho'_k \rangle \\
& + 2\gamma^{-2}\langle F(x(a,\alpha)), F_{xx}(x(a,\alpha))\rho'_i\rho'_k \rangle.
\end{aligned} \tag{4.96}$$

Given an initial guess $a^{(0)}$ for a, the initial guess for α is calculated by $\alpha_j^{(0)} = -\frac{1}{2}a_1^{(0)}a_j^{(0)}$ $(\forall j = 1, \ldots, l)$. The guess $a^{(0)}$ is calculated by minimizing

$$
\begin{aligned}
&f : \mathbb{R}^l \to \mathbb{R} : \\
&a' \to \sum_{k=1}^{l} \left(b_k + \sum_{j=1}^{l} a'_j C_{kj} + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a'_j \right. \\
&\qquad\qquad \left. + \frac{1}{2} c_k^{(11)} a_1'^2 + \sum_{j=2}^{l} c_k^{(1j)} a'_1 a'_j \right)^2 \\
&\quad + \sum_{j=1}^{l} \left(a'_j \| \Delta x_\perp^{(m+1)} \| \right)^2 + \sum_{i=2}^{l} \sum_{j=2}^{l} (a'_i a'_j)^2 + \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} (a'_k a'_i a'_j)^2 .
\end{aligned}
\tag{4.97}
$$

The values $C_{11}, \ldots, C_{ll}$ that appear in the function are defined as in the proof of lemma 4.16. The function prevents the possible blow-up of terms in the Taylor expansion (4.82) of the new residual. Note that due to the replacement of the $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ terms of the update vector, this Taylor expansion is only an approximation.

To minimize (4.97) we need an initial guess $a'^{(0)}$. To construct this guess, we start from the one we used in the SNE method. With $b'_1, \ldots, b'_l$ defined by $b'_j = \langle F(\tilde{x}^{(m+1)}), \phi_j^{(m+1)} \rangle$ $(\forall j = 1, \ldots, l)$ and s the index such that $|b'_s|$ is maximal, this guess was given by

$$
\begin{aligned}
a_s'^{(e)} &= -\operatorname{sgn}\left(\frac{|b'_s|}{\lambda_s^{(m+1)}} \right) \min\left(\left| \frac{|b'_s|}{\lambda_s^{(m+1)}} \right|, \sqrt{|\lambda_s^{(m+1)}|}, \left| \frac{|b'_s|}{\| \Delta x_\perp^{(m+1)} \|} \right| \right), \\
\forall j \neq s : a_j'^{(e)} &= -\operatorname{sgn}\left(\frac{|b'_j|}{\lambda_j^{(m+1)}} \right) \min\left(\left| \frac{|b'_j|}{\lambda_j^{(m+1)}} \right|, |a_s'^{(e)}| \right).
\end{aligned}
$$

To account for the base transformation of the approximate null vectors $\phi_1^{(m+1)}$, $\ldots$, $\phi_l^{(m+1)}$ to $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$, we calculate

$$
\forall j = 1, \ldots, l : a_j'^{(b)} = \sum_{i=1}^{l} D_{ij} a_i'^{(e)}. \tag{4.98}
$$

The choice $a_1'^{(b)}, \ldots, a_l'^{(b)}$ prevents the blow-up of most terms in (4.97), except for the part $\sum_{i=2}^{l} \sum_{j=2}^{l} (a'_i a'_j)^2$. To prevent its blow-up, we require

$$
\max_{i,j=2,\ldots,l} |a'_i a'_j| \lesssim \max_{k=1,\ldots,l} \left| \sum_{j=1}^{l} a'_j C_{kj} \right|. \tag{4.99}
$$

We define

$$
\tilde{\kappa} = \min\left(1, \frac{\max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} a_j'^{(b)} C_{kj} \right|}{\max_{i,j=2,\ldots,l} |a_i'^{(b)} a_j'^{(b)}|} \right) \tag{4.100}
$$

and finally choose

$$
\forall j = 1, \ldots, l : a_j'^{(0)} = \tilde{\kappa} a_j'^{(b)} \tag{4.101}
$$

as an initial guess for the minimization of (4.97).

4.7.5 The method in practice

The method derived in the current section will be called the *split Newton method with reduced terms* (SNR). In each iteration information of the previous one is required, making it impossible to use for the first iteration. In practice, we will first perform a normal split Newton step, afterwards updates of the form (4.93) are used.

Each Newton iteration $(m+1)$ requires approximate null vectors of both the previous and current Jacobian. These are respectively reused from the previous iteration (iteration m), and calculated with an eigensolver (algorithm 3.3, page 43). A base transformation (4.76) is performed, yielding the vectors $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$. The part $\Delta x_\perp^{(m+1)}$ of the update vector is again calculated by solving (4.30) with deflated GMRES (algorithm 3.4, page 44), the $\rho'_1, \ldots, \rho'_l$ terms of the part $\Delta x_\rho^{(m+1)}$ by first orthogonalizing $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ to the approximate null vectors $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$ of the previous iteration (see (4.91)), and then applying $\mathcal{Q}(\tilde{x}^{(m+1)})$. The values for $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ are calculated by minimizing (4.102), the preconditioned version of (4.94). The initial guess $a^{(0)}$ for a is created by first minimizing (4.105), the one for α by setting $\alpha_j^{(0)} = -\frac{1}{2} a_1 a_j$ $(\forall j = 1, \ldots, l)$. To minimize (4.105), the initial guess given by (4.101) is used. The update vector is constructed by (4.93) and the guess is updated. The algorithm ends with the calculation of $\Delta x_\parallel^{(m+1)}$ and γ, and also stores the approximate null vectors, to use in the next iteration. The full SNR method is described by algorithm 4.6 on page 104 in appendix 4.10.

The algorithm is applied in example 4.19. The first iterations show a similar behaviour in convergence as for the SNE method (see section 4.6): assumption 4.15 holds, validating the reduction of the extra terms. After these iterations convergence however slows down, and becomes similar to that of the normal split Newton method (see section 4.5). For these iterations assumption 4.15 does not hold, the additional part $\Delta x_\rho^{(m+1)}$ does not yield a significant speed-up anymore.

Example 4.19. Consider the same set-up as in example 4.3. Application of the SNR method yields the residual plot in figure 4.9. After 8 Newton iterations the residual norm reaches a value of approximately 10^{-10}. Further decrease is however slow. ◇

Note that none of the iterations in the SNR method required additional applications of eigen- or linear solvers. In each iteration eigenpairs of only a single operator needed to be calculated, and only a single linear system had to be solved. The computational work is approximately the same as for the normal split Newton method.

4.7.6 The preconditioned case

Similar to previous methods, in the preconditioned case $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$ and $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ represent the approximate null vectors of respectively the preconditioned Jacobians $P(\tilde{x}^{(m)})F_x(\tilde{x}^{(m)})$ and $P(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)})$. The respective eigenvalues of $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ are denoted by $\lambda_1^{(m+1)}, \ldots, \lambda_l^{(m+1)}$,

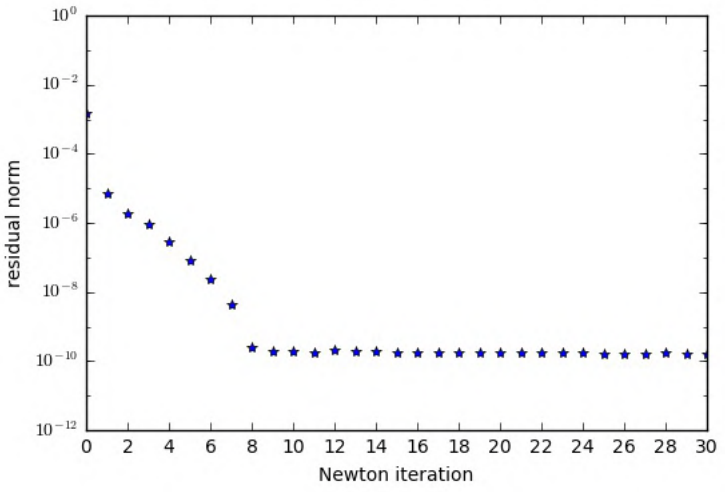


Figure 4.9: Residual plot of example 4.19. The SNR method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-10} after 8 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps.

and we denote $\tau_1^{(m)}, \ldots, \tau_l^{(m)}$, respectively $\tau_1^{(m+1)}, \ldots, \tau_l^{(m+1)}$, such that ($\forall j = 1, \ldots, l$)

$$P(\tilde{x}^{(m)})\tau_j^{(m)} = \phi_j^{(m)},$$
$$P(\tilde{x}^{(m+1)})\tau_j^{(m+1)} = \phi_j^{(m+1)}.$$

To create a new base for the space spanned by $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$, we define

$$\mathring{\phi} = \sum_{j=1}^{l} \langle \tau_j^{(m+1)}, \Delta x_{\|}^{(m)} \rangle \phi_j^{(m+1)}, \qquad \mathring{\tau} = \sum_{j=1}^{l} \langle \tau_j^{(m+1)}, \Delta x_{\|}^{(m)} \rangle \tau_j^{(m+1)},$$
$$\breve{\phi}_1^{(m+1)} = \left(\sqrt{|\langle \mathring{\phi}, \mathring{\tau} \rangle|} \right)^{-1} \mathring{\phi}, \qquad \breve{\tau}_1^{(m+1)} = \left(\sqrt{|\langle \mathring{\phi}, \mathring{\tau} \rangle|} \right)^{-1} \mathring{\tau},$$

with $\Delta x_{\|}^{(m)} = \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)}$. The vectors $\breve{\tau}_2^{(m+1)}, \ldots, \breve{\tau}_l^{(m+1)}$ and $\breve{\phi}_2^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ are chosen as linear combinations of respectively $\tau_1^{(m+1)}, \ldots, \tau_l^{(m+1)}$ and $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ such that $\langle \breve{\tau}_i^{(m+1)}, \breve{\phi}_j^{(m+1)} \rangle = \delta_{ij}$ ($\forall i, j = 1, \ldots, l$) and $P(\tilde{x}^{(m+1)})\breve{\tau}_i^{(m+1)} = \breve{\phi}_i^{(m+1)}$ ($\forall i = 1, \ldots, l$).

As in the unpreconditioned case, the part of the update vector in approximate null vector directions is dominated by $\breve{\phi}_1^{(m+1)}$ under assumption 4.15. The matrix D is defined as before (see (4.77)), the elements of this unitary

matrix are explicitly given by

$$\forall i,j=1,\ldots,l : D_{ij} = \langle \breve{\phi}_j^{(m+1)}, \tau_i^{(m+1)} \rangle.$$

If $P(\tilde{x}^{(m+1)})$ is self-adjoint with respect to $\langle \cdot, \cdot \rangle$, D is again a unitary matrix, implying the relation $D^{-1} = D^T$.

The $\Delta x_\perp^{(m+1)}$ part of the update vector is calculated by solving (4.47), with $\mathcal{Q}(\tilde{x}^{(m+1)})$ defined by (4.15). We split the current residual and second order partial derivatives ($\forall i,j=1,\ldots,l$)

$$F(\tilde{x}^{(m+1)}) = F_\perp + F_\parallel = F_\perp + \sum_{k=1}^{l} b_k \breve{\tau}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)})\Delta x_\perp^{(m+1)} \Delta x_\perp^{(m+1)} = F_\perp^{(00)} + F_\parallel^{(00)} = F_\perp^{(00)} + \sum_{k=1}^{l} c_k^{(00)} \breve{\tau}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)})\Delta x_\perp^{(m+1)} \breve{\phi}_j^{(m+1)} = F_\perp^{(0j)} + F_\parallel^{(0j)} = F_\perp^{(0j)} + \sum_{k=1}^{l} c_k^{(0j)} \breve{\tau}_k^{(m+1)},$$

$$F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)} = F_\perp^{(ij)} + F_\parallel^{(ij)} = F_\perp^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)} \breve{\tau}_k^{(m+1)}$$

with $\langle F_\perp^{(ij)}, \breve{\phi}_k^{(m+1)} \rangle = 0$ ($\forall i,j,k=1,\ldots,l$) and $\langle F_\perp, \breve{\phi}_k^{(m+1)} \rangle = 0$ ($\forall k = 1,\ldots,l$). The values for $b_1,\ldots,b_l$ and $c_k^{(ij)}$ ($\forall i,j,k=1,\ldots,l$) are calculated as

$$\begin{aligned}
&\forall k=1,\ldots,l : b_k = \langle F(\tilde{x}^{(m+1)}), \breve{\phi}_k^{(m+1)} \rangle,\\
&\forall k=1,\ldots,l : c_k^{(00)} = \langle F_{xx}(\tilde{x}^{(m+1)})\Delta x_\perp^{(m+1)} \Delta x_\perp^{(m+1)}, \breve{\phi}_k^{(m+1)} \rangle,\\
&\forall j,k=1,\ldots,l : c_k^{(0j)} = \langle F_{xx}(\tilde{x}^{(m+1)})\Delta x_\perp^{(m+1)} \breve{\phi}_j^{(m+1)}, \breve{\phi}_k^{(m+1)} \rangle,\\
&\forall i,j,k=1,\ldots,l : c_k^{(ij)} = \langle F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)}, \breve{\phi}_k^{(m+1)} \rangle.
\end{aligned}$$

Instead of solving linear systems of the form ($\forall j=1,\ldots,l$)

$$\begin{aligned}
&P(\tilde{x}^{(m+1)})\mathcal{Q}(\tilde{x}^{(m+1)})F_x(\tilde{x}^{(m+1)})z_j^{(m+1)}\\
&\qquad = \quad P(\tilde{x}^{(m+1)})\mathcal{Q}(\tilde{x}^{(m+1)})F_{xx}(\tilde{x}^{(m+1)})\breve{\phi}_1^{(m+1)}\breve{\phi}_j^{(m+1)},
\end{aligned}$$

we will again replace the extra parts $z_1^{(m+1)},\ldots,z_l^{(m+1)}$ of the update vector by introducing $\rho_1,\ldots,\rho_l$, defined in such a way that

$$\forall j=1,\ldots,l : \breve{\phi}_j^{(m+1)} = \sum_{i=1}^{l} \zeta_{ij}\phi_i^{(m)} + \rho_j$$

with $\zeta_{ij} = \langle \breve{\phi}_j^{(m+1)}, \tau_i^{(m)} \rangle$ ($\forall i,j=1,\ldots,l$) and $\langle \rho_j, \tau_i^{(m)} \rangle = 0$ ($\forall i,j=1,\ldots,l$). Similar analysis as for the unpreconditioned case shows that, under assumption 4.17, the $z_1^{(m+1)},\ldots,z_l^{(m+1)}$ terms can be replaced by $-\gamma^{-1}Q(\tilde{x}^{(m+1)})\rho_j$ ($\forall j=1,\ldots,l$), with $\gamma = \| \sum_{j=1}^{l} a_j^{(m)}\phi_j^{(m)} \|_{P(\tilde{x}^{(m)})^{-1}}$.

The update vector is constructed by (4.93), with $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ calculated by minimizing (with $\tilde{P} = P(\tilde{x}^{(m+1)})$)

$$\begin{aligned} &g : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\quad (a, \alpha) \to \left\| F\left(\tilde{x}^{(m+1)} + \Delta x_\perp^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho'_j \right) \right\|_{\tilde{P}}^2 \end{aligned} \tag{4.102}$$

with the nonlinear conjugate gradients method. The first partial derivatives of g are given by ($\forall i = 1, \ldots, l$)

$$\begin{aligned} &\frac{\partial g}{\partial a_i} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to 2 \langle F(x(a, \alpha)), F_x(x(a, \alpha)) \breve{\phi}_i^{(m+1)} \rangle_{\tilde{P}}, \\ &\frac{\partial g}{\partial \alpha_i} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to -2\gamma^{-1} \langle F(x(a, \alpha)), F_x(x(a, \alpha)) \rho'_i \rangle_{\tilde{P}}, \\ &\text{with } x(a, \alpha) = \tilde{x}^{(m+1)} + \Delta x_\perp^{(m+1)} + \sum_{j=1}^{l} a_j \breve{\phi}_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho'_j, \end{aligned} \tag{4.103}$$

the second partial derivatives by ($\forall i, k = 1, \ldots, l$)

$$\begin{aligned} &\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to 2 \langle F_x(x(a, \alpha)) \breve{\phi}_i^{(m+1)}, F_x(x(a, \alpha)) \breve{\phi}_k^{(m+1)} \rangle_{\tilde{P}} \\ &\qquad\qquad + 2 \langle F(x(a, \alpha)), F_{xx}(x(a, \alpha)) \breve{\phi}_i^{(m+1)} \breve{\phi}_k^{(m+1)} \rangle_{\tilde{P}}, \\ &\frac{\partial^2 g}{\partial a_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to -2\gamma^{-1} \langle F_x(x(a, \alpha)) \breve{\phi}_i^{(m+1)}, F_x(x(a, \alpha)) \rho'_k \rangle_{\tilde{P}} \\ &\qquad\qquad - 2\gamma^{-1} \langle F(x(a, \alpha)), F_{xx}(x(a, \alpha)) \breve{\phi}_i^{(m+1)} \rho'_k \rangle_{\tilde{P}}, \\ &\frac{\partial^2 g}{\partial \alpha_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to 2\gamma^{-2} \langle F_x(x(a, \alpha)) \rho'_i, F_x(x(a, \alpha)) \rho'_k \rangle_{\tilde{P}} \\ &\qquad\qquad + 2\gamma^{-2} \langle F(x(a, \alpha)), F_{xx}(x(a, \alpha)) \rho'_i \rho'_k \rangle_{\tilde{P}}. \end{aligned} \tag{4.104}$$

The initial guesses $a^{(0)}$ and $\alpha^{(0)}$ for the minimization of g are constructed similar to the unpreconditioned case, where the preconditioned alternatives for the values of D_{ij}, C_{kj}, b_k and $c_k^{(ij)}$ (with $i, j, k = 1, \ldots, l$) are used. First an

initial guess for the value of a is calculated by minimizing

$$\begin{aligned} f : \mathbb{R}^l \to \mathbb{R} : a' \to \sum_{k=1}^{l} \Bigg(b_k + \sum_{j=1}^{l} a'_j C_{kj} \; + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a'_j \\ + \frac{1}{2} c_k^{(11)} a_1'^2 + \sum_{j=2}^{l} c_k^{(1j)} a'_1 a'_j \Bigg)^2 \\ + \sum_{j=1}^{l} \left(a'_j \| \Delta x_\perp^{(m+1)} \|_{\tilde{P}^{-1}} \right)^2 + \sum_{i=2}^{l} \sum_{j=2}^{l} (a'_i a'_j)^2 \\ + \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} (a'_k a'_i a'_j)^2 \end{aligned} \tag{4.105}$$

with nonlinear conjugate gradients. The initial guess for α is then calculated by $\alpha_j^{(0)} = -\frac{1}{2} a_1^{(0)} a_j^{(0)}$ $(\forall j = 1, \ldots, l)$. To construct an initial guess for minimizing f, we first calculate

$$\begin{aligned} a_s'^{(e)} &= -\operatorname{sgn}\left(\frac{|b'_s|}{\lambda_s^{(m+1)}} \right) \min\left(\left| \frac{|b'_s|}{\lambda_s^{(m+1)}} \right|, \sqrt{|\lambda_s^{(m+1)}|}, \left| \frac{|b'_s|}{\| \Delta x_\perp^{(m+1)} \|_{\tilde{P}^{-1}}} \right| \right), \\ \forall j \neq s : a_j'^{(e)} &= -\operatorname{sgn}\left(\frac{|b'_j|}{\lambda_j^{(m+1)}} \right) \min\left(\left| \frac{|b'_j|}{\lambda_j^{(m+1)}} \right|, |a_s'^{(e)}| \right), \end{aligned}$$

with $b'_j = \langle F(\tilde{x}^{(m+1)}), \phi_j^{(m+1)} \rangle$ $(\forall j = 1, \ldots, l)$. The values $a_1'^{(e)}, \ldots, a_l'^{(e)}$ would be used as guess in the SNE method. The guess for the minimization of f is then calculated by application of (4.98), (4.100) and (4.101).

4.8 When is reduction justified?

The split Newton method with extra terms (SNE), introduced in section 4.6, introduces an additional part Δx_z to the update vector to prevent slow convergence in ill-conditioned problems. This extra part is calculated by solving additional linear systems. To bypass the extra amount of computational work, we approximate this part by a different one Δx_ρ in the split Newton method with reduced terms (SNR), as discussed in section 4.7. This alternative term is calculated by using approximate null vectors of the Jacobian from a previous Newton iteration, and does not require additional applications of linear solvers.

This last method is, however, based on an assumption (see assumption 4.15). The update vector Δx given by

$$\Delta x = \Delta x_\perp + \Delta x_\parallel + \Delta x_\rho \tag{4.106}$$

$$= \Delta x_\perp + \sum_{j=1}^{l} a_j \breve{\phi}_j - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \rho'_j \tag{4.107}$$

for a certain guess $\tilde{x} \in \mathbb{C}^n$, with $\Delta x_\perp$, $\breve{\phi}_1, \ldots, \breve{\phi}_l$, γ and $\rho'_1, \ldots, \rho'_l$ calculated as in section 4.7 (dropping the index $(m+1)$), is only expected to yield a

decent Newton update when $|a_i| \ll |a_1|$ for each $i \neq 1$. The part of the update vector that consists of approximate null vectors is dominated by $\breve{\phi}_1$ under this assumption.

If assumption 4.15 is valid, the SNR method is expected to yield a similar decrease in residual as for the SNE method. If the assumption does not hold, the decrease in residual should be similar to applying the normal split Newton method (SN). Instead of reducing the additional part to Δx_ρ, in this case an update with the part Δx_z, calculated by the SNE method, should be executed.

4.8.1 Derivation of a reduction criterion

Assumption 4.15 is required to neglect the term $\frac{1}{2}\sum_{i=2}^{l}\sum_{j=2}^{l} a_i a_j F_{xx}\breve{\phi}_i\breve{\phi}_j$ in the Taylor expansion of the new residual. To reduce its influence, we should assert that the values $a_1, \ldots, a_l$ satisfy the relation

$$\forall i, j = 2, \ldots, l : |a_i a_j| \lesssim \max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} a_j C_{ij} \right| \tag{4.108}$$

with $C_{11}, \ldots, C_{ll}$ defined as in section 4.7. We will now use this relation to derive a criterion that indicates whether assumption 4.15 holds. Let $\hat{a}'_1, \ldots, \hat{a}'_l$ be the arguments for which

$$\begin{aligned} f : \mathbb{R}^l \to \mathbb{R} : a' \to \sum_{k=1}^{l} \Bigg(b_k + \sum_{j=1}^{l} a'_j C_{kj} + \frac{1}{2} c_k^{(00)} + \sum_{j=1}^{l} c_k^{(0j)} a'_j \\ + \frac{1}{2} c_k^{(11)} a_1'^2 + \sum_{j=2}^{l} c_k^{(1j)} a'_1 a'_j \Bigg)^2 \\ + \sum_{j=1}^{l} \left(a'_j \|\Delta x_\perp\| \right)^2 + \sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l} (a'_k a'_i a'_j)^2, \end{aligned}$$

with $b_1, \ldots, b_l$ defined by (4.40), is minimal. This function contains the different scalars of the Taylor expansion of the new residual, ignoring terms of the form $a_i a_j F_\perp^{(ij)}$ for $i \neq 1$, $j \neq 1$. To check (4.108), we calculate the value

$$\kappa = \min \left(1, \frac{\max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} \hat{a}'_j C_{kj} \right|}{\max_{i,j=2,\ldots,l} |\hat{a}'_i \hat{a}'_j|} \right). \tag{4.109}$$

If $\kappa \approx 1$, relation (4.108) is valid. The $a_i a_j F_\perp^{(ij)}$ terms are indeed negligible for $i \neq 1$, $j \neq 1$. For $\kappa \ll 1$ these terms should not be ignored. Assumption 4.15 is not valid in this case, the reduction of the additional update vector part should not be executed.

4.8.2 The method in practice

We will extend the SNE/SNR methods with the derived criterion, we call the updated algorithm the *split Newton method with mixed terms* (SNM). In prac-

tice we will start each Newton iteration as if executing a SNR step, up to the calculation of $a_1'^{(b)}, \ldots, a_l'^{(b)}$ (used in the creation of an initial guess for the minimization of (4.97), see (4.98)). These values approximate the $\hat{a}_1', \ldots, \hat{a}_l'$ ones defined before. The value κ is now approximated by $\tilde{\kappa}$, calculated from (4.100), and the criterion is checked.

If $\tilde{\kappa} \approx 1$, the step of the SNR method is continued, it is expected to perform as well as a SNE step. If $\tilde{\kappa} \ll 1$, the values $a_1, \ldots, a_l$ that minimize the norm of the new residual are typically too small for a good update to be performed. In this case we do not continue the construction of the update vector used in the SNR method. Instead we shift to the calculation of the $z_{11}, \ldots, z_{ll}$ terms to use in a step of the SNE method.

The resulting algorithm is given by algorithm 4.7 on page 106 in appendix 4.10. Note that a normal split Newton step is executed in the first iteration, since we require information on the kernel of the Jacobian at a previous Newton iteration for the calculation of reduced terms.

The SNM method is applied in example 4.20. 24 Newton steps are required for convergence, which is slightly higher than when extra terms are used in each iteration (see example 4.12). The total computational work required to converge is however reduced: only in 13 Newton steps additional linear systems had to be solved. In example 4.12 each of the 19 steps had this requirement. For this example, application of the SNM method yields a converged approximation in less time than the SNE one.

Example 4.20. Consider the same set-up as in example 4.3. Application of the SNM method yields the residual plot in figure 4.10. Convergence (up to a tolerance of 10^{-12}) is reached in 24 Newton iterations. In 13 of the 24 iterations additional linear systems had to be solved, for 10 iterations reduced terms were used instead. The value $\epsilon_{\tilde{\kappa}} = 10^{-2}$ was used as a threshold for $\tilde{\kappa}$ to choose the sort of update. ◇

By adding a criterion, we combined both the robustness of the SNE method, and the low amount of computational work of the SNR one. The resulting algorithm is computationally more efficient than the SNE method, and retains its ability to efficiently solve ill-conditioned problems.

4.8.3 The preconditioned case

The analysis in the current section is nearly identical for the preconditioned case. We still calculate $\tilde{\kappa}$ and perform a SNE or SNR step depending on its value. The single difference is the use of preconditioned versions for the values of $C_{11}, \ldots, C_{ll}$ in the calculation of $\tilde{\kappa}$ (see (4.100)).

4.9 Discussion on convergence

To end the chapter we summarize the results on convergence of the derived Newton methods, and make a note on possible further adjustments.

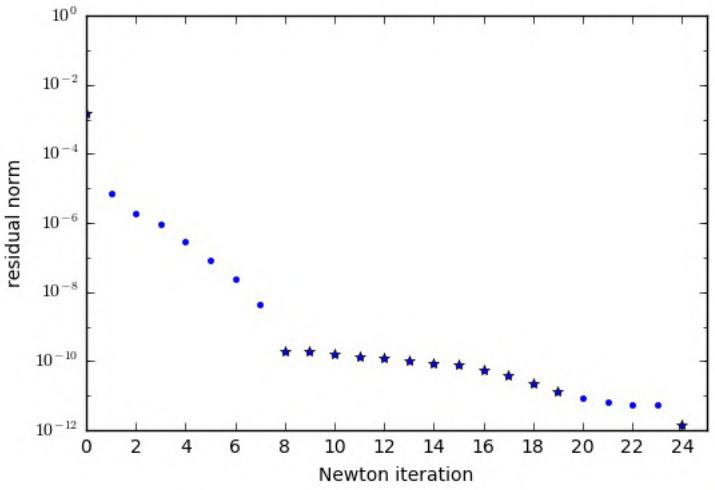


Figure 4.10: Residual plot of example 4.20. The SNM method is applied to an ill-conditioned nonlinear problem, where the Jacobian contains a three-dimensional kernel in the searched solution. 11 Newton steps were executed with reduced terms (illustrated by circles), 13 with extra terms (illustrated by stars). The residual norm converges to approximately 10^{-12} after 24 iterations, which is the attainable accuracy. Compared to figure 4.8 more iterations are required to reach this value, but the total computational work is reduced.

4.9.1 Convergence of the split Newton methods

In section 4.6 we introduced an update vector of the form

$$\Delta x = \Delta x_\perp + \Delta x_\parallel + \Delta x_z \tag{4.110}$$

$$= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} z_{ij}, \tag{4.111}$$

with $\phi_1, \ldots, \phi_l$ the null vectors of the Jacobian at the current guess $\tilde{x}$, $z_{11}, \ldots, z_{ll}$ the solutions of additional linear systems (4.55) and $a_1, \ldots, a_l$, α_{11}, α_{ll} determined by minimizing the new residual norm. The part Δx_z was introduced to counter the influence of second order derivatives of the form $a_i a_j F_{xx}(\tilde{x})\phi_i\phi_j$ on the new residual (for $i, j = 1, \ldots, l$). In section 4.5 we showed that these derivatives imposed the condition

$$\max_{i,j=1,\ldots,l} |a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \lambda_j| \tag{4.112}$$

on the values $a_1, \ldots, a_l$. $\lambda_1, \ldots, \lambda_l$ denote the eigenvalues corresponding to the respective approximate null vectors $\phi_1, \ldots, \phi_l$. With $s \in \{1, \ldots, l\}$ the index for which $|a_s\lambda_s|$ is maximal, this condition lead to the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim |\lambda_s|. \tag{4.113}$$

Since $|\lambda_j| \ll 1$ ($\forall j = 1, \ldots, l$), this imposes $|a_j| \ll 1$ ($\forall j = 1, \ldots, l$), resulting in small updates to the residual and overall slow convergence.

By introducing the $\alpha_{ij} z_{ij}$ terms (for $i, j = 1, \ldots, l$) to the update vector, condition (4.112) was removed by choosing $\alpha_{ij} \approx -\frac{1}{2} a_i a_j$ ($\forall i, j = 1, \ldots, l$). A different bound is then however induced by other derivatives, of the forms $a_k \alpha_{ij} F_{xx}(\tilde{x}) \phi_k z_{ij}$ and $a_k a_i a_j F_{xxx}(\tilde{x}) \phi_k \phi_i \phi_j$, which both are of order $\mathcal{O}(a_k a_i a_j)$ (with $i, j, k = 1, \ldots, l$). The new condition was given by

$$\max_{k,i,j=1,\ldots,l} |a_k a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \lambda_j| \tag{4.114}$$

and lead to the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt{|\lambda_s|}. \tag{4.115}$$

This new bound is not as strict as (4.113), and does not hamper convergence as much.

4.9.2 Possible further adjustments

Though updates with (4.111) usually yield decent convergence behaviour, we note that the analysis can be further extended. To counter the condition (4.114), more parts can be introduced to the update vector by solving the linear systems ($\forall i, j, k = 1, \ldots, l$)

$$\mathcal{Q}(\tilde{x}) F_x(\tilde{x}) v_{ijk} = \mathcal{Q}(\tilde{x}) F_{xx}(\tilde{x}) \phi_k z_{ij}, \tag{4.116}$$

$$\mathcal{Q}(\tilde{x}) F_x(\tilde{x}) w_{ijk} = \mathcal{Q}(\tilde{x}) F_{xx}(\tilde{x}) \phi_i \phi_j \phi_k. \tag{4.117}$$

The update vector would become

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel + \Delta x_z + \Delta x_v + \Delta x_w \\ &= \Delta x_\perp + \sum_{j=1}^{l} a_j \phi_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_{ij} z_{ij} + \sum_{i=1}^{l} \sum_{j=1}^{l} \sum_{k=1}^{l} (\eta_{ijk} v_{ijk} + \sigma_{ijk} w_{ijk}) \end{aligned} \tag{4.118}$$

in this case. Condition (4.114) can be reduced by choosing $\eta_{ijk} = -\frac{1}{2} a_k \alpha_{ij}$ and $\sigma_{ijk} = -\frac{1}{6} a_i a_j a_k$ ($\forall i, j, k = 1, \ldots, l$).

Though (4.114) is countered by introducing (4.118), a new condition is again introduced by derivatives of order $\mathcal{O}(a_q a_k a_i a_j)$ (with $i, j, k, q = 1, \ldots, l$). This new condition is given by

$$\max_{q,k,i,j=1,\ldots,l} |a_q a_k a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \lambda_j| \tag{4.119}$$

and would yield the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt[3]{|\lambda_s|}. \tag{4.120}$$

The difference with bound (4.115) is not as profound as the one between (4.115) and (4.113), but a small increase in $|a_1|, \ldots, |a_l|$ values is possible. The process to remove (4.114) can be repeated: even further update vector parts can be

introduced to exchange (4.119) for a slightly milder condition. By repeatedly adding extra terms, the condition on $|a_1|, \ldots, |a_l|$ is relaxed.

Though an extension to the split Newton method with extra terms (SNE) is possible, it is not practical: to reduce condition (4.114) to (4.119) an additional $\frac{1}{6}l(l+1)(4l+2)$ linear systems need to be solved: $\frac{1}{2}l^2(l+1)$ of form (4.116) and $\frac{1}{6}l(l+1)(l+2)$ of form (4.117). The milder condition we achieve by introducing further extra parts does not outweigh the cost of solving the required additional linear systems. Furthermore, the analysis done in sections 4.7 and 4.8, which was used to reduce the amount of linear solver applications, cannot be extended to this alternative. In our applications the split Newton method with mixed terms (SNM) will be used.

4.10 Appendix

The standard Newton method

Algorithm 4.1 NewtonStandard

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

1: Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
2: Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: $\tilde{x} = x^{(0)}$
5: $r = F(\tilde{x})$
6: $i = 0$
7: **while** $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
8: $\quad i \leftarrow i + 1$
9: $\quad$ Calculate Δx by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$
10: $\quad \tilde{x} \leftarrow \tilde{x} + \Delta x$
11: $\quad r \leftarrow F(\tilde{x})$
12: **end while**
13: Return $\tilde{x}$

The deflated Newton method

Algorithm 4.2 NewtonDeflated

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

1: Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
2: Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: $\tilde{x} = x^{(0)}$
5: $r = F(\tilde{x})$
6: $i = 0$
7: **while** $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
8: $\quad i \leftarrow i + 1$
9: $\quad$ Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
10: $\quad$ Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x})$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$ and K
11: $\quad U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
12: $\quad$ Calculate Δx by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$

13: $\tilde{x} \leftarrow \tilde{x} + \Delta x$
14: $r \leftarrow F(\tilde{x})$
15: **end while**
16: Return $\tilde{x}$

The Newton method with line search (NLS)

Algorithm 4.3 NewtonLS

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

1: Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
2: Define $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$ by (4.3) if not specified
3: Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
4: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
5: $\tilde{x} = x^{(0)}$
6: $r = F(\tilde{x})$
7: $i = 0$
8: **while** $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
9: $i \leftarrow i + 1$
10: Calculate Δx by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$
11: Define g, g' and g'' as in (4.24), (4.25) and (4.26)
12: $\xi^{(0)} = \min(1, 2\|r\|_{P(\tilde{x})}\|\Delta x\|^{-2}_{P(\tilde{x})^{-1}})$
13: Calculate $\tilde{\xi}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \xi^{(0)}$ and given g
14: $\tilde{x} \leftarrow \tilde{x} + \tilde{\xi}\Delta x$
15: $r \leftarrow F(\tilde{x})$
16: **end while**
17: Return $\tilde{x}$

The split Newton method (SN)

Algorithm 4.4 NewtonSplit

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

1: Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
2: Define $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$ by (4.3) if not specified
3: Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
4: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified

$\tilde{x} = x^{(0)}$
$r = F(\tilde{x})$
$i = 0$
while $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$
  Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
  Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x})$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$ and K
  $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
  $W \leftarrow \begin{pmatrix} P(\tilde{x})K & W \end{pmatrix}$
  Calculate $\Delta x_\perp$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  for $j = 1, \ldots, l$ **do**
    $b_j = \langle r, W_j \rangle$
  end for
  Calculate $a'^{(0)}$ by formula (4.52) (with $\lambda_j = L_{jj}$)
  Define f by (4.51) (with $\lambda_j = L_{jj}$)
  Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
  Define g, g' and g'' by (4.48), (4.49) and (4.50) (with $\phi_j = W_j$)
  Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
  $\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j W_j$
  $\tilde{x} \leftarrow \tilde{x} + \Delta x$
  $r \leftarrow F(\tilde{x})$
end while
Return $\tilde{x}$

The split Newton method with extra terms (SNE)

Algorithm 4.5 NewtonExtra

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
Define $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$ by (4.3) if not specified
Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
$\tilde{x} = x^{(0)}$
$r = F(\tilde{x})$
$i = 0$
while $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$

Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x})$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$ and K
$U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
$W \leftarrow \begin{pmatrix} P(\tilde{x})K & W \end{pmatrix}$
Calculate $\Delta x_\perp$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
for $j = 1, \ldots, l$ **do**
  for $k = 1, \ldots, j$ **do**
    $v_{jk} = F_{xx}(\tilde{x}) W_j W_k$
    Calculate z_{jk} by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = v_{jk}$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
    $z_{kj} = z_{jk}$
  end for
end for
$v_{00} = F_{xx}(\tilde{x}) \Delta x_\perp \Delta x_\perp$
for $k = 1, \ldots, l$ **do**
  $b_k = \langle r, W_k \rangle$
  $c_k^{(00)} = \langle v_{00}, W_k \rangle$
  $v_{0k} = F_{xx}(\tilde{x}) \Delta x_\perp W_k$
  for $j = 1, \ldots, l$ **do**
    $c_j^{(0k)} = \langle v_{0k}, W_j \rangle$
    for $p = 1, \ldots, k$ **do**
      $c_j^{(pk)} = \langle v_{pk}, W_j \rangle$
      $c_j^{(kp)} = c_j^{(pk)}$
    end for
  end for
end for
Calculate $a'^{(0)}$ by formula (4.74) (with $\lambda_j = L_{jj}$)
Define f by (4.73) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
for $j = 1, \ldots, l$ **do**
  for $k = 1, \ldots, j$ **do**
    $\alpha_{jk}^{(0)} = -\frac{1}{2} \tilde{a}_j^{(0)} \tilde{a}_k^{(0)}$
    $\alpha_{kj}^{(0)} = \alpha_{jk}^{(0)}$
  end for
end for
Define g, g' and g'' by (4.70), (4.71) and (4.72) (with $\phi_j = W_j$)
Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
$\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j W_j + \sum_{j=1}^{l} \sum_{k=1}^{l} \tilde{\alpha}_{jk} z_{jk}$
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$r \leftarrow F(\tilde{x})$
end while
Return $\tilde{x}$

The split Newton method with reduced terms (SNR)

Algorithm 4.6 NewtonReduced

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
Define $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$ by (4.3) if not specified
Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
$\tilde{x} = x^{(0)}$
$r = F(\tilde{x})$
$i = 0$
while $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$
  Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
  Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x})$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$ and K
  $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
  $W \leftarrow \begin{pmatrix} P(\tilde{x})K & W \end{pmatrix}$
  Calculate $\Delta x_\perp$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  if $i = 1$ **then**
    $\breve{W} = W$
    $\breve{U} = U$
    for $j = 1, \ldots, l$ **do**
      $b_j = \langle r, W_j \rangle$
    end for
    Calculate $a'^{(0)}$ by formula (4.52) (with $\lambda_j = L_{jj}$)
    Define f by (4.51) (with $\lambda_j = L_{jj}$)
    Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
    Define g, g' and g'' by (4.48), (4.49) and (4.50) (with $\phi_j = W_j$)
    Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
    $\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j W_j$
  else
    $\mathring{W} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle W_j$
    $\mathring{U} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle U_j$
    $\beta = \sqrt{|\langle \mathring{W}, \mathring{U} \rangle|}$
    $\breve{W}_1 = \beta^{-1} \mathring{W}$
    $\breve{U}_1 = \beta^{-1} \mathring{U}$
    Create $\breve{W}_2, \ldots, \breve{W}_l$ and $\breve{U}_2, \ldots, \breve{U}_l$ as linear combinations of respectively $W_1, \ldots, W_l$ and $U_1, \ldots, U_l$ such that $\langle \breve{U}_j, \breve{W}_k \rangle = \delta_{jk}$ and $P(\tilde{x})\breve{U}_j = \breve{W}_j$ for each $j, k = 1, \ldots, l$

for $j = 1, \dots, l$ **do**
  for $k = 1, \dots, l$ **do**
    $D_{jk} = \langle \breve{W}_k, U_j \rangle$
  end for
end for
$C = D^T L D$
for $j = 1, \dots, l$ **do**
  $\rho'_j = \breve{W}_j$
  for $k = 1, \dots, l$ **do**
    $\rho'_j \leftarrow \rho'_j - \hat{W}_k \langle \rho'_j, \hat{U}_k \rangle$
  end for
  for $k = 1, \dots, l$ **do**
    $\rho'_j \leftarrow \rho'_j - W_k \langle \rho'_j, U_k \rangle$
  end for
end for
$v_{00} = F_{xx}(\tilde{x}) \Delta x_\perp \Delta x_\perp$
for $k = 1, \dots, l$ **do**
  $b_k = \langle r, \breve{W}_k \rangle$
  $c_k^{(00)} = \langle v_{00}, \breve{W}_k \rangle$
  $v_{0k} = F_{xx}(\tilde{x}) \Delta x_\perp \breve{W}_k$
  $v_{1k} = F_{xx}(\tilde{x}) \breve{W}_1 \breve{W}_k$
  for $j = 1, \dots, l$ **do**
    $c_j^{(0k)} = \langle v_{0k}, \breve{W}_j \rangle$
    $c_j^{(1k)} = \langle v_{1k}, \breve{W}_j \rangle$
  end for
end for
Calculate $a'^{(0)}$ by formula (4.101) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
Define f by (4.105) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
for $j = 1, \dots, l$ **do**
  $\alpha_j^{(0)} = -\frac{1}{2} \tilde{a}_1^{(0)} \tilde{a}_j^{(0)}$
end for
Define g, g' and g'' by (4.102), (4.103) and (4.104) (with $\breve{\phi}_j = \breve{W}_j$)
Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
$\Delta x = \Delta x_\perp + \sum_{j=1}^l \tilde{a}_j \breve{W}_j - \gamma^{-1} \sum_{j=1}^l \tilde{\alpha}_j \rho'_j$
end if
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$r \leftarrow F(\tilde{x})$
$\hat{U} \leftarrow U$
$\hat{W} \leftarrow W$
$\Delta x_\parallel = \sum_{j=1}^l \tilde{a}_j \breve{W}_j$
$\gamma \leftarrow \sqrt{\langle \sum_{j=1}^l \tilde{a}_j \breve{W}_j, \sum_{j=1}^l \tilde{a}_j \breve{U}_j \rangle}$
end while
Return $\tilde{x}$

The split Newton method with mixed terms (SNM)

Algorithm 4.7 NewtonMixed

Input $m_{New} \in \mathbb{N}$, tolerances $\epsilon_{New}, \epsilon_{\tilde{\kappa}} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \to \mathbb{C}^n$, $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$, $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, initial guess $x^{(0)} \in \mathbb{C}^n$

Output Approximation $\tilde{x}$ for $F(x) = 0$

Define $F_x : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n)$ by (4.2) if not specified
Define $F_{xx} : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n \times \mathbb{C}^n)$ by (4.3) if not specified
Set $P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : x \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
$\tilde{x} = x^{(0)}$
$r = F(\tilde{x})$
$i = 0$
while $i < m_{New}$ and $\|r\|_{P(\tilde{x})} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$
  Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
  Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x})$, $\mathcal{P} = P(\tilde{x})$ and given $\langle \cdot, \cdot \rangle$ and K
  $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
  $W \leftarrow \begin{pmatrix} P(\tilde{x})K & W \end{pmatrix}$
  Calculate $\Delta x_\perp$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = -r$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  if $i = 1$ **then**
    $\breve{W} = W$
    $\breve{U} = U$
    for $j = 1, \ldots, l$ **do**
      $b_j = \langle r, W_j \rangle$
    end for
    Calculate $a'^{(0)}$ by formula (4.52) (with $\lambda_j = L_{jj}$)
    Define f by (4.51) (with $\lambda_j = L_{jj}$)
    Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
    Define g, g' and g'' by (4.48), (4.49) and (4.50) (with $\phi_j = W_j$)
    Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
    $\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j W_j$
  else
    $\mathring{W} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle W_j$
    $\mathring{U} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle U_j$
    $\beta = \sqrt{|\langle \mathring{W}, \mathring{U} \rangle|}$
    $\breve{W}_1 = \beta^{-1} \mathring{W}$
    $\breve{U}_1 = \beta^{-1} \mathring{U}$
    Create $\breve{W}_2, \ldots, \breve{W}_l$ and $\breve{U}_2, \ldots, \breve{U}_l$ as linear combinations of respectively $W_1, \ldots, W_l$ and $U_1, \ldots, U_l$ such that $\langle \breve{U}_j, \breve{W}_k \rangle = \delta_{jk}$ and $P(\tilde{x})\breve{U}_j = \breve{W}_j$ for each $j, k = 1, \ldots, l$

for $j = 1, \dots, l$ **do**
  for $k = 1, \dots, l$ **do**
    $D_{jk} = \langle \breve{W}_k, U_j \rangle$
  end for
end for
$C = D^T L D$
Calculate $\tilde{\kappa}$ by formula (4.100) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
if $\tilde{\kappa} > \epsilon_{\tilde{\kappa}}$ **then**
  for $j = 1, \dots, l$ **do**
    $\rho'_j = \breve{W}_j$
    for $k = 1, \dots, l$ **do**
      $\rho'_j \leftarrow \rho'_j - \hat{W}_k \langle \rho'_j, \hat{U}_k \rangle$
    end for
    for $k = 1, \dots, l$ **do**
      $\rho'_j \leftarrow \rho'_j - W_k \langle \rho'_j, U_k \rangle$
    end for
  end for
  $v_{00} = F_{xx}(\tilde{x}) \Delta x_\perp \Delta x_\perp$
  for $k = 1, \dots, l$ **do**
    $b_k = \langle r, \breve{W}_k \rangle$
    $c_k^{(00)} = \langle v_{00}, \breve{W}_k \rangle$
    $v_{0k} = F_{xx}(\tilde{x}) \Delta x_\perp \breve{W}_k$
    $v_{1k} = F_{xx}(\tilde{x}) \breve{W}_1 \breve{W}_k$
    for $j = 1, \dots, l$ **do**
      $c_j^{(0k)} = \langle v_{0k}, \breve{W}_j \rangle$
      $c_j^{(1k)} = \langle v_{1k}, \breve{W}_j \rangle$
    end for
  end for
  Calculate $a'^{(0)}$ by formula (4.101) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
  Define f by (4.105) (with $\lambda_j = L_{jj}$)
  Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
  for $j = 1, \dots, l$ **do**
    $\alpha_j^{(0)} = -\frac{1}{2} \tilde{a}_1^{(0)} \tilde{a}_j^{(0)}$
  end for
  Define g, g' and g'' by (4.102), (4.103) and (4.104) (with $\breve{\phi}_j = \breve{W}_j$)
  Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
  $\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j \breve{W}_j - \gamma^{-1} \sum_{j=1}^{l} \tilde{\alpha}_j \rho'_j$
else
  for $j = 1, \dots, l$ **do**
    for $k = 1, \dots, j$ **do**
      $v_{jk} = F_{xx}(\tilde{x}) W_j W_k$
      Calculate z_{jk} by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x})$, $b = v_{jk}$, $\mathcal{P} = P(\tilde{x})$, $K = U$ and given $\langle \cdot, \cdot \rangle$

$z_{kj} = z_{jk}$
end for
end for
$v_{00} = F_{xx}(\tilde{x})\Delta x_\perp \Delta x_\perp$
for $k = 1, \dots, l$ **do**
$b_k = \langle r, W_k \rangle$
$c_k^{(00)} = \langle v_{00}, W_k \rangle$
$v_{0k} = F_{xx}(\tilde{x})\Delta x_\perp W_k$
for $j = 1, \dots, l$ **do**
$c_j^{(0k)} = \langle v_{0k}, W_j \rangle$
for $p = 1, \dots, k$ **do**
$c_j^{(pk)} = \langle v_{pk}, W_j \rangle$
$c_j^{(kp)} = c_j^{(pk)}$
end for
end for
end for
Calculate $a'^{(0)}$ by formula (4.74) (with $\lambda_j = L_{jj}$)
Define f by (4.73) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
for $j = 1, \dots, l$ **do**
for $k = 1, \dots, j$ **do**
$\alpha_{jk}^{(0)} = -\frac{1}{2}\tilde{a}_j^{(0)}\tilde{a}_k^{(0)}$
$\alpha_{kj}^{(0)} = \alpha_{jk}^{(0)}$
end for
end for
Define g, g' and g'' by (4.70), (4.71) and (4.72) (with $\phi_j = W_j$)
Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
$\Delta x = \Delta x_\perp + \sum_{j=1}^{l} \tilde{a}_j W_j + \sum_{j=1}^{l}\sum_{k=1}^{l} \tilde{\alpha}_{jk} z_{jk}$
end if
end if
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$r \leftarrow F(\tilde{x})$
$\hat{U} \leftarrow U$
$\hat{W} \leftarrow W$
$\Delta x_\parallel = \sum_{j=1}^{l} \tilde{a}_j \breve{W}_j$
$\gamma \leftarrow \sqrt{\langle \sum_{j=1}^{l} \tilde{a}_j \breve{W}_j, \sum_{j=1}^{l} \tilde{a}_j \breve{U}_j \rangle}$
end while
Return $\tilde{x}$

CHAPTER 5

The Newton-Krylov method for extended nonlinear systems

"In mathematics you don't understand things. You just get used to them."

– ***John von Neumann*** –

Chapter highlights:

- We review the block elimination technique, used to solve nonlinear systems extended with a single linear equation.
- We show how to combine the derived methods of chapter 4 with block elimination, preparing them for use in a numerical continuation setting.
- The results in the first part of this chapter are mainly based on the following references: [113, 18, 50].
- A journal article about the contents of this chapter and the previous one is in preparation.

5.1 Introduction

In chapter 4 we derived several versions of the Newton-Krylov method that find the zeros of a nonlinear function $F : \mathbb{C}^n \to \mathbb{C}^n$. In multiple applications, for example pseudo-arclength continuation (see section 6.4) or Newton step length adaptation (see section 6.6.2), we are interested in the zeros of a nonlinear function $H : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R}$ of the form

$$H(x,p) = \begin{cases} F(x,p) \\ G(x,p) \end{cases} \tag{5.1}$$

with $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$ a nonlinear and $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$ a linear function. Typically the partial Jacobian F_x of F contains desirable properties, that improve

the convergence speed when solving linear systems with this operator specifically. For example, a preconditioner might be given for F_x, or the operator might be self-adjoint with respect to a certain inner product.

However, direct application of the Newton-Krylov method to the function H nullifies these properties. As a consequence solving linear systems with the Jacobian of H typically requires a much larger amount of computational work than solving ones with F_x. The computational time required to converge is often more than twice as long.

Instead of a direct application of the Newton-Krylov method to (5.1), we will combine the method with block elimination [50]. Instead of solving a single linear system with the Jacobian of H, this technique allows for a replacement by two linear systems with the partial Jacobian F_x. The desirable properties of F_x are not nullified and can still be exploited when solving these two systems. The two approximate solutions are then combined to obtain one for the original linear system. By exploiting the properties of F_x, this technique typically leads to a reduction of total computational work.

In this chapter we will combine the refined Newtom methods of chapter 4 with a block elimination strategy. Possible ill-conditionedness of F_x near the solution can lead to diverging convergence behaviour when approximating the zeros of (5.1) with the standard Newton-Krylov method, even when block elimination is applied. Convergence will be improved by making similar adjustments as before. Note that we will restrict ourselves to problems with a Hermitian partial Jacobian F_x.

5.1.1 Preliminaries

In the remainder of the chapter we will consider a nonlinear function of the form (5.1) and try to find approximations $\tilde{x} \in \mathbb{C}^n$, $\tilde{p} \in \mathbb{R}$ of

$$H(x, p) = 0.$$

The partial derivatives $F_x(x, p) : \mathbb{C}^n \to \mathbb{C}^n$ and $G_x(x, p) : \mathbb{C}^n \to \mathbb{R}$ are assumed to be known as linear operators for each $x \in \mathbb{C}^n, p \in \mathbb{R}$, partial derivatives $F_p(x, p) \in \mathbb{C}^n$ and $G_p(x, p) \in \mathbb{R}$ as vectors. If not available, these derivatives are approximated by first-order central finite difference schemes [43]:

$$F_x(x, p) : \mathbb{C}^n \to \mathbb{C}^n : v \to \frac{1}{2\varepsilon} \left(F(x + \varepsilon v, p) - F(x - \varepsilon v, p) \right), \tag{5.2}$$

$$G_x(x, p) : \mathbb{C}^n \to \mathbb{C}^n : v \to \frac{1}{2\varepsilon} \left(G(x + \varepsilon v, p) - G(x - \varepsilon v, p) \right), \tag{5.3}$$

$$F_p(x, p) = \frac{1}{2\varepsilon} \left(F(x, p + \varepsilon) - F(x, p - \varepsilon) \right), \tag{5.4}$$

$$G_p(x, p) = \frac{1}{2\varepsilon} \left(G(x, p + \varepsilon) - G(x, p - \varepsilon) \right), \tag{5.5}$$

with $\varepsilon \in \mathbb{R}_0^+$ a small real number, typically chosen as $\varepsilon = \sqrt[3]{\varepsilon_{mach}}$ [63]. Note that the approximations of $G_x(x, p)$ and $G_p(x, p)$ are exact since G is a linear

function. As in chapter 4 we further assume $F_x(x,p)$ to be self-adjoint. The extension of the analysis in this chapter to non-Hermitian problems is again straightforward.

The second partial derivatives $F_{xx}(x,p) : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n$ and $F_{xp}(x,p) : \mathbb{C}^n \to \mathbb{C}^n$ are assumed to be known as a bilinear, respectively linear, operator. The partial derivative $F_{pp}(x,p) \in \mathbb{C}^n$ again as a vector. When not given, second-order central schemes are used as approximation [43]:

$$\begin{aligned} F_{xx}(x,p) :&\mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n : \\ &(v,w) \to \frac{1}{4\varepsilon^2}\left(F(x+\varepsilon v+\varepsilon w,p) - F(x+\varepsilon v-\varepsilon w,p)\right. \\ &\qquad \left. -F(x-\varepsilon v+\varepsilon w,p) + F(x-\varepsilon v-\varepsilon w,p)\right), \end{aligned} \tag{5.6}$$

$$\begin{aligned} F_{xp}(x,p) : \mathbb{C}^n \to \mathbb{C}^n : v \to \frac{1}{4\varepsilon^2}&\left(F(x+\varepsilon v,p+\varepsilon) - F(x+\varepsilon v,p-\varepsilon)\right. \\ &\left. -F(x-\varepsilon v,p+\varepsilon) + F(x-\varepsilon v,p-\varepsilon)\right), \end{aligned} \tag{5.7}$$

$$F_{pp}(x,p) = \frac{1}{\varepsilon^2}\left(F(x,p+\varepsilon) - 2F(x,p) + F(x,p-\varepsilon)\right). \tag{5.8}$$

The value ε is typically chosen as $\varepsilon = \sqrt[4]{\varepsilon_{mach}}$ [63]. Note that second and higher partial derivatives of G equal zero by the functions linearity.

The full derivatives of F will be denoted by $F_X(x,p)$, $F_{XX}(x,p)$, etc. The first derivative is defined as

$$F_X(x,p) : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (v,q) \to F_x(x,p)v + qF_p(x,p),$$

the second as

$$\begin{aligned} F_{XX}(x,p) &: (\mathbb{C}^n \times \mathbb{R}) \times (\mathbb{C}^n \times \mathbb{R}) \to \mathbb{C}^n : \\ &((v,q),(w,r)) \to F_{xx}(x,p)vw + qF_{xp}(x,p)w + rF_{xp}(x,p)v + qrF_{pp}(x,p). \end{aligned}$$

Other derivatives are defined analogously.

The algorithms described in this chapter will again be given for the preconditioned case, where we assume a Hermitian, positive definite preconditioner is given for linear systems with the operator $F_x(x,p)$. The preconditioner is again constructed by a single function $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$. The unpreconditioned case is derived by defining

$$P : \mathbb{C}^n \to \mathcal{C}(\mathbb{C}^n) : (x,p) \to \mathcal{I},$$

with $\mathcal{I}$ the identity operator. As for the algorithms described in chapter 4, it is again possible to provide null vectors of $F_x(x,p)$ that are known in advance, which reduces computational work.

5.2 The standard method

5.2.1 Description of the method

We first describe the combination of the standard Newton-Krylov method with block elimination, without the inclusion of any deflation or splitting techniques.

Starting from initial guesses $x^{(0)} \in \mathbb{C}^n$ and $p^{(0)} \in \mathbb{R}$, each Newton iteration i calculates update vectors Δx and Δp, and updates the guesses by

$$x^{(i)} = x^{(i-1)} + \Delta x, \tag{5.9}$$

$$p^{(i)} = p^{(i-1)} + \Delta p. \tag{5.10}$$

As in the case without block elimination (see section 4.2), Δx and Δp are calculated from a linear system derived from the Taylor expansion of $H(x, p)$. For general guesses $\tilde{x}$ and $\tilde{p}$, this linear system is given by

$$\begin{pmatrix} F_x(\tilde{x},\tilde{p}) & F_p(\tilde{x},\tilde{p}) \\ G_x(\tilde{x},\tilde{p}) & G_p(\tilde{x},\tilde{p}) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta p \end{pmatrix} = - \begin{pmatrix} F(\tilde{x},\tilde{p}) \\ G(\tilde{x},\tilde{p}) \end{pmatrix}. \tag{5.11}$$

Instead of approximating Δx and Δp by a direct application of a Krylov method, we apply a block elimination technique [50]. The update vector Δx is written as

$$\Delta x = y^{(1)} - \Delta p y^{(2)} \tag{5.12}$$

with $y^{(1)}, y^{(2)} \in \mathbb{C}^n$ to be determined. Substitution of (5.12) in (5.11) yields

$$\Big(F_x(\tilde{x},\tilde{p})y^{(1)} + F(\tilde{x},\tilde{p})\Big) + \Delta p \Big(-F_x(\tilde{x},\tilde{p})y^{(2)} + F_p(\tilde{x},\tilde{p})\Big) = 0, \tag{5.13}$$

$$\Big(G_x(\tilde{x},\tilde{p})y^{(1)} + G(\tilde{x},\tilde{p})\Big) + \Delta p \Big(-G_x(\tilde{x},\tilde{p})y^{(2)} + G_p(\tilde{x},\tilde{p})\Big) = 0. \tag{5.14}$$

Equation (5.13) is now solved by defining $y^{(1)}$ and $y^{(2)}$ as the solutions of the linear systems

$$F_x(\tilde{x},\tilde{p})y^{(1)} = -F(\tilde{x},\tilde{p}), \tag{5.15}$$

$$F_x(\tilde{x},\tilde{p})y^{(2)} = F_p(\tilde{x},\tilde{p}). \tag{5.16}$$

Equation (5.14) is solved by calculating Δp as

$$\Delta p = \frac{-G(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(2)}}. \tag{5.17}$$

Together with (5.12), this yields the solution of the linear system (5.11), used as update vector in (5.9) and (5.10).

In practice the linear systems (5.15) and (5.16) are approximately solved with a Krylov method. We will use GMRES (see algorithm 3.4, page 44) for this purpose. Instead of a single, direct application of GMRES to (5.11), two linear systems with the partial Jacobian $F_x(\tilde{x},\tilde{p})$ need to be solved.

Pseudo-code for the standard Newton method with block elimination (abbreviated as the *standard block Newton method*) is given by algorithm 5.1 on page 157 in appendix 5.10. Each Newton iteration solves (5.15) and (5.16) with GMRES. The vectors $y^{(1)}$ and $y^{(2)}$ are then used to calculate update vectors Δx and Δp by (5.17) and (5.12). The guess is updated by applying (5.9) and (5.10), and a new iteration is started.

Note that, by the linearity of G, we have $G(\tilde{x},\tilde{p}) = 0$ for each guess except possibly the initial one: By calculating the update vector Δp by (5.17), we assert the next guess $(\tilde{x} + \Delta x, \tilde{p} + \Delta p)$ to be a zero of G.

5.2.2 Analysis of convergence for ill-conditioned problems

The method is applied to examples 5.1, 5.2 and 5.3.

Example 5.1. We consider the Liouville-Bratu-Gelfand equation (see section 2.4), applied to a square domain with 900 discretization points. We will search a zero of the function $\mathcal{H}$, defined as

$$\mathcal{H}(\psi, \mu) = \begin{cases} \mathcal{F}(\psi, \mu) \\ G(\psi, \mu) \end{cases}$$

with $\mathcal{F}$ defined by (2.7), and G to be determined. A point $(\psi^{(0)}, \mu^{(0)})$ is chosen on the line between points $(\psi'^{(1)}, \mu'^{(1)})$ and $(\psi'^{(2)}, \mu'^{(2)})$, given by

$$\psi'^{(1)} = \begin{pmatrix} 0.372 \\ 0.372 \\ \vdots \\ 0.372 \end{pmatrix}, \qquad \psi'^{(2)} = \begin{pmatrix} 0.403 \\ 0.403 \\ \vdots \\ 0.403 \end{pmatrix},$$

$\mu'^{(1)} = 0.256$ and $\mu'^{(2)} = 0.269$. Note that both $(\psi'^{(1)}, \mu'^{(1)})$ and $(\psi'^{(2)}, \mu'^{(2)})$ are solutions of (2.7). The point $(\psi^{(0)}, \mu^{(0)})$ is chosen such that $\mu^{(0)} = 0.3$. The pseudo-arclength condition (see (6.9)) will be used as the linear function G.

The standard block Newton algorithm is applied to find a zero of $\mathcal{H}$, with $(\psi^{(0)}, \mu^{(0)})$ as the initial guess. As an inner product (2.9) is used, we do not apply a preconditioner. The residual plot of the example is given by figure 5.1. Convergence (up to a tolerance of 10^{-12}) is reached after 3 Newton iterations. ◇

Example 5.2. We consider a similar set-up as in example 5.1, this time for the Ginzburg-Landau equation (see section 2.5) applied to a pentagon-shaped material with $n = 10401$ discretization points. The function $\mathcal{H}$ is created as in example 5.1, with $\mathcal{F}$ given by (2.20) and G again the pseudo-arclength condition. The point $(\psi^{(0)}, \mu^{(0)})$ is chosen on the line between points $(\psi'^{(1)}, \mu'^{(1)})$ and $(\psi'^{(2)}, \mu'^{(2)})$ such that $\mu^{(0)} = 1.058$, points $(\psi'^{(1)}, \mu'^{(1)})$ and $(\psi'^{(2)}, \mu'^{(2)})$ represent solutions of (2.20) for respectively $\mu'^{(1)} = 1.063$ and $\mu'^{(2)} = 1.061$ (both lying on branch B in figure 9.23, see section 9.5.3).

Application of the standard block Newton algorithm yields the residual plot of figure 5.2. The inner product (2.21) and preconditioner (2.25) were used. As an initial guess the point $(\tilde{\psi}^{(0)}, \mu^{(0)})$ was taken, with $\tilde{\psi}^{(0)}$ a small perturbation of $\psi^{(0)}$. Though convergence (up to a tolerance of 10^{-12}) is reached in 14 Newton iterations, the behaviour is not optimal: for multiple steps the residual norm shows a strong increase instead of a decrease. As a consequence, it's possible that the solution found by the algorithm lies far from the initial guess $(\psi^{(0)}, \mu^{(0)})$. In applications this is often not desirable. ◇

Example 5.3. We consider the same equation $\mathcal{H}$, inner product and preconditioner as for example 5.2. This time we choose the point $(\psi^{(0)}, \mu^{(0)})$ on the line between $(\psi'^{(1)}, \mu'^{(1)})$, a solution of (2.20) for $\mu'^{(1)} = 1.531$, and $(\psi'^{(2)}, \mu'^{(2)})$,

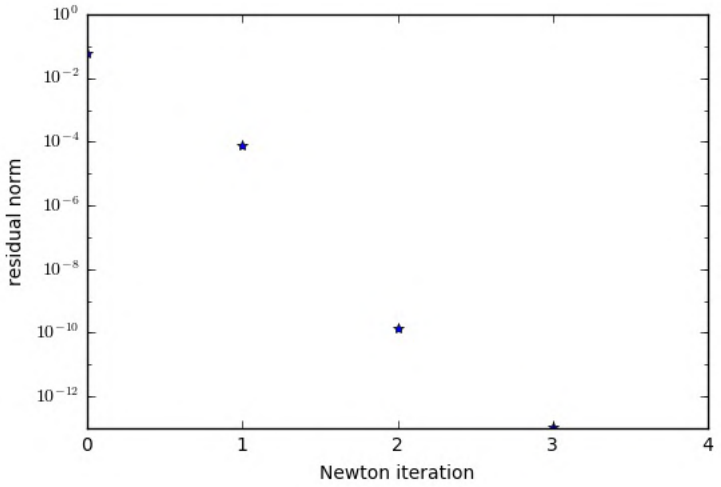


Figure 5.1: Residual plot of example 5.1. The standard block Newton method is applied to a well-conditioned nonlinear problem. The residual norm converges to approximately 10^{-12} after 3 iterations, which is the attainable accuracy. Quadratic convergence is observed in the iterations.

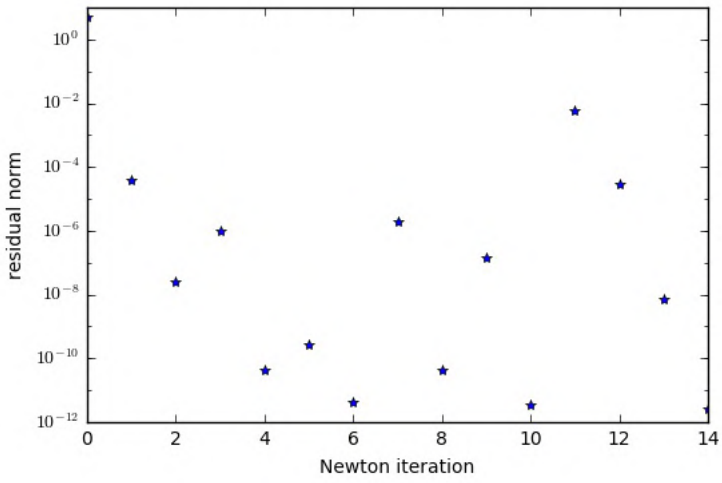


Figure 5.2: Residual plot of example 5.2. The standard block Newton method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-12} after 14 iterations, which is the attainable accuracy. Some of the iterations show sudden jumps that increase the residual norm.

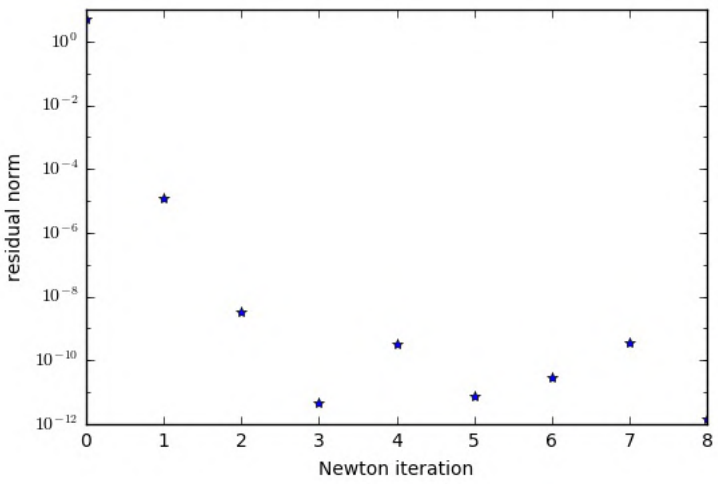


Figure 5.3: Residual plot of example 5.3. The standard block Newton method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a one-dimensional kernel in the searched solution. This kernel is induced by a continuous symmetry. The residual norm converges to approximately 10^{-12} after 8 iterations, which is the attainable accuracy. Some of the iterations show sudden jumps that increase the residual norm. Note that these jumps are less profound than in figure 5.2.

a solution for $\mu'^{(2)} = 1.533$. Both solutions lie on branch B in figure 9.23 (see section 9.5.3). The point $(\psi^{(0)}, \mu^{(0)})$ is chosen such that $\mu^{(0)} = 1.5348$. As an initial guess we consider $(\tilde{\psi}^{(0)}, \mu^{(0)})$, where $\tilde{\psi}^{(0)}$ again represents a small perturbation of $\psi^{(0)}$.

Application of the standard block Newton algorithm yields the residual plot of figure 5.3. Convergence (up to a tolerance of 10^{-12}) is again reached, after 9 Newton iterations. In several steps there is a strong increase in residual norm however. ◇

In example 5.1 the partial Jacobian $F_x(\tilde{x}, \tilde{p})$ is well-conditioned for each Newton guess $\tilde{x}, \tilde{p}$, leading to the typical quadratic convergence seen in figure 5.1. For examples 5.2 and 5.3 convergence does not behave well. The operator $F_x(\tilde{x}, \tilde{p})$ is ill-conditioned near the solution of $H(x, p) = 0$ for these problems.

Lemma 5.4. An iteration of the standard block Newton method possibly leads to a diverging update if executed in a guess $\tilde{x}, \tilde{p}$ for which the partial Jacobian $F_x(\tilde{x}, \tilde{p})$ is approximately singular. This is caused by a blow-up of the update vector Δx in the directions of the null vectors.

Proof. Similar analysis as in the proof of lemma 4.2 (see section 4.2.2) shows that, when $F_x(\tilde{x}, \tilde{p})$ is ill-conditioned, the solutions $y^{(1)}$ and $y^{(2)}$ of the linear systems (5.15) and (5.16) both contain a part that consists of approximate null

vectors of this operator. The linear model the Newton method bases itself upon is not valid for these vectors, typically leading to the parts being blown up. By (5.12) the approximate null vector part of the update vector Δx is blown up as well, possibly resulting in a worse Newton guess. □

Lemma 5.4 clarifies how the approximate singularity of $F_x(\tilde{x}, \tilde{p})$ causes the deterioration of convergence seen in examples 5.2 and 5.3. Far from the solution good Newton updates are still performed, for approximations close to the solution this is not the case and divergence is even observed in figures 5.2 and 5.3. For example 5.3 the ill-conditionedness of $F_x(\tilde{x}, \tilde{p})$ is due to a continuous symmetry of $F(x, p)$, for example 5.2 it is caused by a combination of this continuous symmetry and a nearby bifurcation.

5.2.3 The preconditioned case

The preconditioned case is adapted in a similar way as in section 4.2.3. The linear systems (5.15) and (5.16) are adjusted to

$$P(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})y^{(1)} = -P(\tilde{x}, \tilde{p})F(\tilde{x}, \tilde{p}), \tag{5.18}$$

$$P(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})y^{(2)} = P(\tilde{x}, \tilde{p})F_p(\tilde{x}, \tilde{p}). \tag{5.19}$$

Together with a small adjustment to the convergence criterium, this is the only difference between the preconditioned and unpreconditioned cases. Approximate singularity of the partial Jacobian $F_x(\tilde{x}, \tilde{p})$ is still a cause for possible diverging convergence behaviour.

5.3 The use of deflation to prevent divergence

5.3.1 Description of the method

The blow-up of the parts in $y^{(1)}$ and $y^{(2)}$ (and Δx) consisting of approximate null vectors, is prevented when deflation is applied to linear systems (5.15) and (5.16). Consider a guess $\tilde{x} \in \mathbb{C}^n$, $\tilde{p} \in \mathbb{R}$. Deflation is realized by adjusting these linear systems to [17, 18]

$$\mathcal{Q}(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})y^{(1)} = -\mathcal{Q}(\tilde{x}, \tilde{p})F(\tilde{x}, \tilde{p}), \tag{5.20}$$

$$\mathcal{Q}(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})y^{(2)} = \mathcal{Q}(\tilde{x}, \tilde{p})F_p(\tilde{x}, \tilde{p}), \tag{5.21}$$

with $\mathcal{Q}(\tilde{x}, \tilde{p})$ the operator that projects vectors away from the approximate null vectors $\phi_1, \ldots, \phi_l$ of $F_x(\tilde{x}, \tilde{p})$. This operator is defined similarly as in section 4.3.1. With K the matrix whose columns are given by $\phi_1, \ldots, \phi_l$, this operator is defined by

$$\mathcal{Q}(\tilde{x}, \tilde{p}) : \mathbb{C}^n \to \mathbb{C}^n : w \to w - K\langle K, K\rangle^{-1}\langle K, w\rangle. \tag{5.22}$$

The update vectors Δx and Δp are calculated as before (see (5.12) and (5.17)). When the linear systems (5.20) and (5.21) are used instead of (5.15) and (5.16), the calculated vectors $y^{(1)}$ and $y^{(2)}$ will be orthogonal to $\phi_1, \ldots, \phi_l$. By (5.12) the update vector Δx is orthogonal to these vectors as well. This

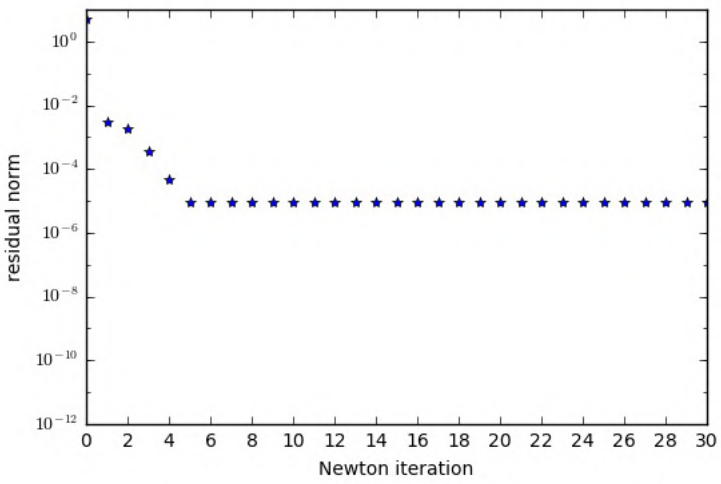


Figure 5.4: Residual plot of example 5.5. The deflated block Newton method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-5} after 5 iterations, the attainable accuracy for well-conditioned problems is however 10^{-12}. After 5 iterations there is no further significant decrease.

prevents the possible blow-up in the parts consisting of approximate null vectors, which caused the unwanted convergence behaviour described by lemma 5.4.

Algorithm 5.2 (see page 157 in appendix 5.10) contains pseudo-code for the updated Newton method, which we will call the *deflated block Newton method*. It is similar to algorithm 5.1 (page 157), except for the use of deflation when solving linear systems. In each iteration the approximate null vectors of $F_x(\tilde{x}, \tilde{p})$ are calculated first by application of algorithm 3.3 (page 43). Next, the updated linear systems (5.20) and (5.21) are solved with deflated GMRES (algorithm 3.4, page 44) for $y^{(1)}$ and $y^{(2)}$. Equations (5.17) and (5.12) eventually yield the update vectors used to calculate the next guess.

5.3.2 Analysis of convergence for ill-conditioned problems

The method is applied in examples 5.5 and 5.6, both problems contain an ill-conditioned Jacobian near the solution.

Example 5.5. Consider the same set-up as in example 5.2. The deflated block Newton method is applied, yielding the residual plot of figure 5.4. Stagnation at approximately 10^{-5} occurs after 5 Newton iterations. ◇

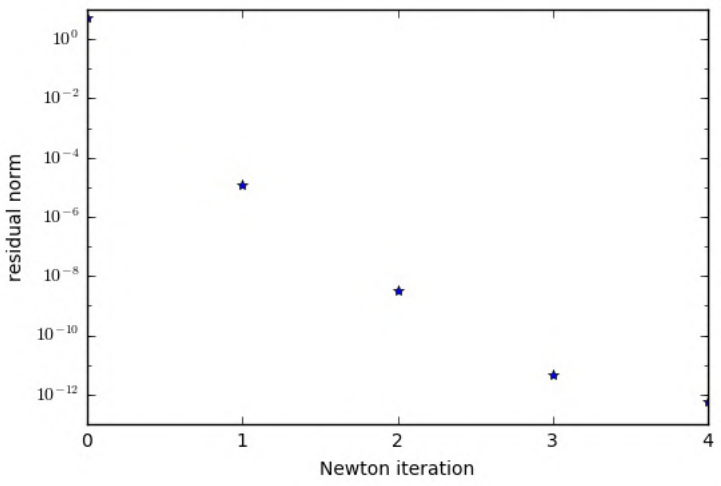


Figure 5.5: Residual plot of example 5.6. The deflated block Newton method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a one-dimensional kernel in the searched solution. This kernel is induced by a continuous symmetry. The residual norm converges to approximately 10^{-12} after 4 iterations, which is the attainable accuracy.

Example 5.6. Consider the same set-up as in example 5.3. Application of the deflated block Newton method yields the residual plot of figure 5.5. Convergence (up to a tolerance of 10^{-12}) is reached in 4 Newton iterations. ◇

The residual plot that corresponds to example 5.5 (see figure 5.4) shows stagnation of the residual norm after only a few iterations.

Lemma 5.7. The deflated block Newton method typically leads to stagnation of residuals when applied to a problem for which the partial Jacobian F_x is (approximately) singular in the searched solution. This is caused by the update part of Δx consisting of approximate null vectors being ignored.

Proof. Denote $\tilde{x}, \tilde{p}$ the current guess for the solution. When deflation is used, the update vector Δx is orthogonal to the approximate null vectors of $F_x(\tilde{x}, \tilde{p})$. Stagnation occurs after the part perpendicular to these vectors has vanished: in this case the residual $F(\tilde{x}, \tilde{p})$ consists entirely of approximate null vectors and $y^{(1)}$ will be calculated as the zero vector. If $G(\tilde{x}, \tilde{p}) = 0$, this leads to no update being performed. □

If the approximate singularity of the partial Jacobian is solely caused by null vectors induced by continuous symmetries, the method often works fine. This is the case for example 5.6, where the method converges fast (see figure 5.5). As noted in section 4.3.2, this is a consequence of each solution being a representative for a complete family of solutions.

5.3.3 The preconditioned case

When a preconditioner is provided, the projection operator is adjusted to

$$\mathcal{Q}(\tilde{x},\tilde{p}) : \mathbb{C}^n \to \mathbb{C}^n : w \to w - K\langle K, L\rangle^{-1}\langle L, w\rangle, \tag{5.23}$$

with

$$K = \begin{pmatrix}\tau_1 & \tau_2 & \dots & \tau_l\end{pmatrix},$$
$$L = \begin{pmatrix}\phi_1 & \phi_2 & \dots & \phi_l\end{pmatrix},$$

such that $P(\tilde{x},\tilde{p})K = L$. Approximate null vectors $\phi_1, \dots, \phi_l$ of $P(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})$ are used. The linear systems (5.20) and (5.21) are adjusted to

$$P(\tilde{x},\tilde{p})\mathcal{Q}(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})y^{(1)} = -P(\tilde{x},\tilde{p})\mathcal{Q}(\tilde{x},\tilde{p})F(\tilde{x},\tilde{p}), \tag{5.24}$$
$$P(\tilde{x},\tilde{p})\mathcal{Q}(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})y^{(2)} = P(\tilde{x},\tilde{p})\mathcal{Q}(\tilde{x},\tilde{p})F_p(\tilde{x},\tilde{p}). \tag{5.25}$$

The preconditioned version of the algorithm still typically leads to stagnation when the (preconditioned) partial Jacobian $P(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})$ is approximately singular in the wanted solution.

5.4 The use of line search to prevent divergence

5.4.1 Description of the method

To prevent the blow-up of the part consisting of approximate null vectors in the update vector, without entirely removing this part (as was done in section 5.3), the standard block Newton method can again be adapted with a line search technique [113, 67], similar to section 4.4. For guesses $\tilde{x} \in \mathbb{C}^n$, $\tilde{p} \in \mathbb{R}$, we calculate the solutions $y^{(1)}$ and $y^{(2)}$ of the linear systems (5.15) and (5.16) as in section 5.2. In practice these systems are solved with GMRES (without deflation). The update vector Δx is calculated from these by

$$\Delta x = y^{(1)} - \frac{-G(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(2)}}y^{(2)}. \tag{5.26}$$

Instead of immediately calculating Δp by (5.17), the values

$$c_0 = \frac{-G(\tilde{x},\tilde{p})}{G_p(\tilde{x},\tilde{p})}, \qquad c_1 = \frac{-G_x(\tilde{x},\tilde{p})\Delta x}{G_p(\tilde{x},\tilde{p})}$$

are calculated. We now consider an update of the form [113, 67]

$$\tilde{x} \leftarrow \tilde{x} + \xi\Delta x, \tag{5.27}$$
$$\tilde{p} \leftarrow \tilde{p} + c_0 + \xi c_1, \tag{5.28}$$

where we still need to determine $\xi \in \mathbb{R}$. By defining the update to $\tilde{p}$ as in (5.28), we assert the next guess to be a zero of G. We, indeed, have for any

$\xi \in \mathbb{R}$, by the definition of c_0 and c_1:

$$\begin{aligned} & G(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1) \\ & \qquad = \quad G(\tilde{x}, \tilde{p}) + \xi G_x(\tilde{x}, \tilde{p})\Delta x + c_0 G_p(\tilde{x}, \tilde{p}) + \xi c_1 G_p(\tilde{x}, \tilde{p}) \\ & \qquad = \quad G(\tilde{x}, \tilde{p}) + \xi G_x(\tilde{x}, \tilde{p})\Delta x - G(\tilde{x}, \tilde{p}) - \xi G_x(\tilde{x}, \tilde{p})\Delta x \\ & \qquad = \quad 0. \end{aligned}$$

The update (5.27) to $\tilde{x}$ is similar to the one used in section 4.4. It is possible that approximate singularity of $F_x(\tilde{x}, \tilde{p})$ yields a blown-up update vector Δx. By choosing ξ sufficiently small, this does however not lead to a worse Newton guess.

In practice the value for ξ is calculated by application of the nonlinear conjugate gradients (NCG) algorithm (see section 3.4) to minimize the function

$$g : \mathbb{R} \to \mathbb{R} : \xi \to \left\| F(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1) \right\|^2 . \tag{5.29}$$

The first and second derivatives of g are derived later for the preconditioned case, see (5.34) and (5.35) in section 5.4.3. The ones required for the minimization of (5.29) are derived from these by setting $P(\tilde{x}, \tilde{p}) = \mathcal{I}$, the identity operator. As initial guess for the minimization of (5.29) the value

$$\begin{aligned} & \xi^{(0)} = \min\left(1, \|F(\tilde{x}, \tilde{p})\|\gamma\right), \\ & \text{with } \gamma = \min(2\|\Delta x\|^{-2}, |c_1|^{-1}\|\Delta x\|^{-1}, 2|c_1|^{-2}, |c_0|^{-1}\|\Delta x\|^{-1}, |c_0|^{-1}|c_1|^{-1}), \end{aligned} \tag{5.30}$$

is used. This choice prevents second order derivatives from dominating the residual of the next guess, as will be explained section 5.4.2 (see the bound given in lemma 5.9).

Linear systems (5.15) and (5.16) remain unchanged when adapting the Newton method with a line search technique, no deflation is used to calculate $y^{(1)}$ and $y^{(2)}$. The only difference with the standard method (described in section 5.2.1) is the introduction of ξ, and the manner in which the updates to $\tilde{x}$ and $\tilde{p}$ are executed.

Note that the standard method is derived from the one with line search by setting $\xi = 1$. The update vector Δx remains unchanged, equation (5.26) is derived from substituting (5.17) in (5.12). The update (5.27) to $\tilde{x}$ is equivalent to (5.9) for $\xi = 1$, the same is true for the update Δp to $\tilde{p}$: by the definition of c_0 and c_1, (5.28) and (5.10) are equivalent for $\xi = 1$.

The *block Newton method with line search* (BNLS) is given by algorithm 5.3 on page 158 in appendix 5.10. Linear systems (5.15) and (5.16) are solved for $y^{(1)}$ and $y^{(2)}$ by application of GMRES (algorithm 3.4, page 44), which are then used to calculate the update vector Δx by (5.26). The guesses are updated by (5.27) and (5.28), where ξ is determined by applying the nonlinear conjugate gradients method (algorithm 3.5, page 45) to (5.33), the preconditioned version of (5.29). After updating the guesses, a new iteration is started.

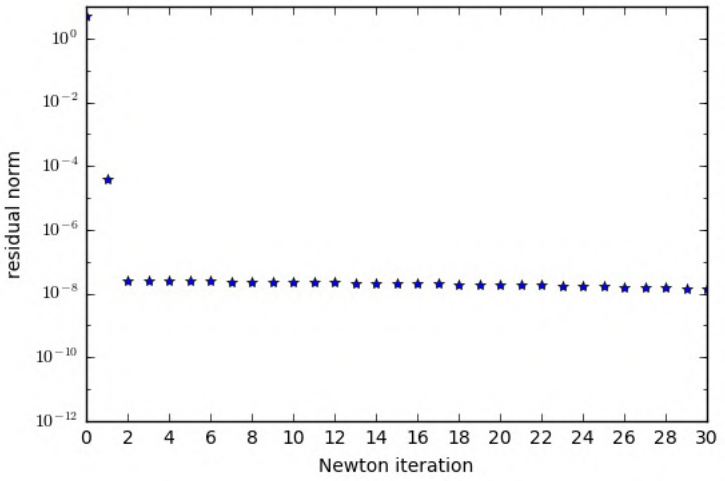


Figure 5.6: Residual plot of example 5.8. The BNLS method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm decreases to approximately 10^{-8} after 2 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps.

5.4.2 Analysis of convergence for ill-conditioned problems

The algorithm is applied in example 5.8. Figure 5.6 shows that though no diverging updates are performed, convergence is slow.

Example 5.8. Consider the same set-up as in example 5.2. Application of the BNLS method yields the residual plot of figure 5.6. Apparent stagnation occurs at a residual norm of approximately 10^{-8}, after 2 Newton iterations. ◇

Slow convergence of the BNLS method is clarified by similar analysis as was done for the method without block elimination (lemma 4.9 in section 4.4.2).

Lemma 5.9. After some initial iterations, convergence of the BNLS method is typically slow when applied to an ill-conditioned problem. The slowdown of convergence has multiple causes:

- The update part of Δx perpendicular to the null vectors of $F_x(\tilde{x}, \tilde{p})$ is damped due to the multiplication with ξ. This part does not cause any problems and is actually approximated well, its damping might lead to an unnecessarily small update.
- The value ξ used in updating the guesses $\tilde{x}$ and $\tilde{p}$ typically satisfies the bound

$$\xi \lesssim \|F(\tilde{x}, \tilde{p})\| \min\left(2\|\Delta x\|^{-2}, |c_1|^{-1}\|\Delta x\|^{-1}, 2c_1^{-2}\right). \tag{5.31}$$

The update vector Δx consists of vectors $y^{(1)}$ and $y^{(2)}$, calculated from (5.15) and (5.16). These vectors typically blow up if the partial Jacobian $F_x(\tilde{x}, \tilde{p})$ is approximately singular.

Proof. The proof for the first cause is entirely similar to the one presented in lemma 4.9. For the second cause, consider the second-order Taylor expansion of $F(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1)$, given by:

$$\begin{aligned} &F(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1) \\ &\quad = F(\tilde{x}, \tilde{p}) + \xi F_x(\tilde{x}, \tilde{p})\Delta x + (c_0 + \xi c_1)F_p(\tilde{x}, \tilde{p}) + \frac{1}{2}\xi^2 F_{xx}(\tilde{x}, \tilde{p})\Delta x\Delta x \\ &\qquad + \xi(c_0 + \xi c_1)F_{xp}(\tilde{x}, \tilde{p})\Delta x + \frac{1}{2}(c_0 + \xi c_1)^2 F_{pp}(\tilde{x}, \tilde{p}) + \mathcal{O}(\|\Delta x\|^3) \\ &\qquad + \mathcal{O}((c_0 + \xi c_1)^3). \end{aligned}$$

Substitution of $F_x(\tilde{x}, \tilde{p})\Delta x = -F(\tilde{x}, \tilde{p}) - (c_0 + c_1)F_p(\tilde{x}, \tilde{p})$ (which follows from the definition of Δx) and rewriting the terms yields

$$\begin{aligned} &F(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1) \\ &\quad = (1 - \xi)F(\tilde{x}, \tilde{p}) + (1 - \xi)c_0 F_p(\tilde{x}, \tilde{p}) + \frac{1}{2}c_0^2 F_{pp}(\tilde{x}, \tilde{p}) \\ &\qquad + \xi\left(c_0 F_{xp}(\tilde{x}, \tilde{p})\Delta x + c_0 c_1 F_{pp}(\tilde{x}, \tilde{p})\right) \\ &\qquad + \xi^2\left(\frac{1}{2}F_{xx}(\tilde{x}, \tilde{p})\Delta x\Delta x + c_1 F_{xp}(\tilde{x}, \tilde{p})\Delta x + \frac{1}{2}c_1^2 F_{pp}(\tilde{x}, \tilde{p})\right) \\ &\qquad + \mathcal{O}(\|\Delta x\|^3) + \mathcal{O}((c_0 + \xi c_1)^3). \end{aligned}$$

To prevent the second partial derivatives of order proportional to ξ^2 from dominating the residual of the next iteration, the condition

$$\xi^2 \max\left(\frac{1}{2}\|\Delta x\|^2, c_1\|\Delta x\|, \frac{1}{2}c_1^2\right) \lesssim \xi\|F(\tilde{x}, \tilde{p})\| \tag{5.32}$$

is required. This choice assures the dominance of the term $\xi F_x(\tilde{x}, \tilde{p})\Delta x$. Rewriting the condition yields the bound (5.31) on ξ. □

The first cause mentioned in lemma 5.9 concerns the part of the update vector Δx perpendicular to the null vectors of $F_x(\tilde{x}, \tilde{p})$. This part is possibly much smaller than required when line search is applied.

The second cause concerns a bound on ξ. If $\|\Delta x\| \gg 1$ (which is typical when $F_x(\tilde{x}, \tilde{p})$ becomes ill-conditioned) bound (5.31) is very strict, typically $\xi \ll 1$ is chosen. The residual term $F(\tilde{x}, \tilde{p})$ of the Taylor expansion only reduces slightly, leading to slow convergence of the method.

The choice (5.30), used in the minimization of (5.29), is based on the bound (5.31), but also prevents the possible blow-up of the parts $c_0 F_{xp}(\tilde{x}, \tilde{p})\Delta x$ and $c_0 c_1 F_{pp}(\tilde{x}, \tilde{p})$. Note that, by choosing the update (5.28) such that each updated guess becomes a zero of G, we have $c_0 = 0$ for each Newton guess except possibly the initial one.

5.4.3 The preconditioned case

When a preconditioner is provided, the linear systems to solve for $y^{(1)}$ and $y^{(2)}$ again need to be adapted to (5.18) and (5.19). The update vector Δx is still defined by application of (5.26). Updates of the form (5.27) and (5.28) are again considered, with ξ this time determined by minimizing the function

$$g : \mathbb{R} \to \mathbb{R} : \xi \to \|F(\tilde{x} + \xi\Delta x, \tilde{p} + c_0 + \xi c_1)\|^2_{P(\tilde{x},\tilde{p})}\,. \tag{5.33}$$

This minimization is executed by the nonlinear conjugate gradients algorithm (see section 3.4) in practice, the derivatives of g are given by

$$g' : \mathbb{R} \to \mathbb{R} : \xi \to 2\langle F(\tilde{x} + \xi\Delta x), F_x(\tilde{x} + \xi\Delta x)\Delta x\rangle_{P(\tilde{x},\tilde{p})}, \tag{5.34}$$

$$\begin{aligned} g'' : \mathbb{R} \to \mathbb{R} : \xi \to & 2\langle F_x(\tilde{x} + \xi\Delta x)\Delta x, F_x(\tilde{x} + \xi\Delta x)\Delta x\rangle_{P(\tilde{x},\tilde{p})} \\ & + 2\langle F(\tilde{x} + \xi\Delta x), F_{xx}(\tilde{x} + \xi\Delta x)\Delta x\Delta x\rangle_{P(\tilde{x},\tilde{p})}. \end{aligned} \tag{5.35}$$

As an initial guess for the minimization of g, the value

$$\begin{aligned} &\xi^{(0)} = \min\left(1, \|F(\tilde{x},\tilde{p})\|_{P(\tilde{x},\tilde{p})}\gamma\right), \\ &\text{with } \gamma = \min(2\|\Delta x\|^{-2}_{P(\tilde{x},\tilde{p})^{-1}}, |c_1|^{-1}\|\Delta x\|^{-1}_{P(\tilde{x},\tilde{p})^{-1}}, 2|c_1|^{-2}, \\ &\qquad\qquad |c_0|^{-1}\|\Delta x\|^{-1}_{P(\tilde{x},\tilde{p})^{-1}}, |c_0|^{-1}|c_1|^{-1}), \end{aligned} \tag{5.36}$$

is used. The preconditioned method still converges slowly when the partial Jacobian $F_x(\tilde{x},\tilde{p})$ becomes ill-conditioned, due to similar causes as for the unpreconditioned case.

5.5 Splitting the update vector

5.5.1 Description of the method

To counter the slow convergence of both the deflated block Newton method (see section 5.3), and the method adapted with line search (see section 5.4), we first combine both techniques in the current section. Consider a guess $\tilde{x} \in \mathbb{C}^n$, $\tilde{p} \in \mathbb{R}$ for which the partial Jacobian $F(\tilde{x},\tilde{p})$ becomes ill-conditioned, and denote the eigenvalues and -vectors of this operator by $\lambda_1, \ldots, \lambda_l$, respectively $\phi_1, \ldots, \phi_l$ (with $\|\phi_1\| = \cdots = \|\phi_l\| = 1$).

Similar to section 4.5, we will split the update vector Δx in a part $\Delta x_\parallel$ consisting of the vectors $\phi_1, \ldots, \phi_l$, and a perpendicular part $\Delta x_\perp$. The update Δp is split into two parts (denoted $\Delta p_\perp$ and $\Delta p_\parallel$) as well. With

$$K = \begin{pmatrix}\phi_1 & \phi_2 & \ldots & \phi_l\end{pmatrix},$$

the projection operator $\mathcal{Q}(\tilde{x},\tilde{p})$ is again defined as

$$\mathcal{Q}(\tilde{x},\tilde{p}) : \mathbb{C}^n \to \mathbb{C}^n : w \to w - K\langle K, K\rangle^{-1}\langle K, w\rangle. \tag{5.37}$$

The part $\Delta x_\perp$ of Δx perpendicular to $\phi_1, \ldots, \phi_l$ will be calculated from $y^{(1)}_\perp$ and $y^{(2)}_\perp$, defined as the solutions of the deflated linear systems

$$\mathcal{Q}(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})y^{(1)}_\perp = -\mathcal{Q}(\tilde{x},\tilde{p})F(\tilde{x},\tilde{p}), \tag{5.38}$$

$$\mathcal{Q}(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})y^{(2)}_\perp = \mathcal{Q}(\tilde{x},\tilde{p})F_p(\tilde{x},\tilde{p}), \tag{5.39}$$

such that $\langle y_\perp^{(1)}, \phi_j \rangle = 0$ and $\langle y_\perp^{(2)}, \phi_j \rangle = 0$ ($\forall j = 1, \ldots, l$). With

$$c_0 = \frac{-G(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}, \tag{5.40}$$

$$\forall j = 1, \ldots, l : c_j = \frac{-G_x(\tilde{x}, \tilde{p}) \phi_j}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}, \tag{5.41}$$

the complete update vectors Δx and Δp are defined by

$$\begin{aligned} \Delta x &= \Delta x_\perp + \Delta x_\parallel & (5.42)\\ &= y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right), & (5.43)\\ \Delta p &= \Delta p_\perp + \Delta p_\parallel & (5.44)\\ &= c_0 + \sum_{j=1}^{l} a_j c_j, & (5.45) \end{aligned}$$

where $a_1, \ldots, a_l \in \mathbb{R}$ have still to be determined. The Newton guesses are updated by these vectors by application of (5.9) and (5.10). The values $a_1, \ldots, a_l$ are calculated by minimizing the function

$$g : \mathbb{R}^l \to \mathbb{R} : a \to \left\| F\left(\tilde{x} + y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right), \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j \right) \right\|^2 \tag{5.46}$$

with the nonlinear conjugate gradients method (see section 3.4). Details on how this minimization is executed in practice are given in section 5.5.3.

By splitting both the update vectors Δx and Δp as in (5.43) and (5.45), the next guess is asserted to be a zero of the function G:

$$\begin{aligned} G(\tilde{x} + \Delta x, \tilde{p} + \Delta p) &= G(\tilde{x} + \Delta x_\perp + \Delta x_\parallel, \tilde{p} + \Delta p_\perp + \Delta p_\parallel)\\ &= G(\tilde{x}, \tilde{p}) + G_x(\tilde{x}, \tilde{p}) \Delta x_\perp + G_p(\tilde{x}, \tilde{p}) \Delta p_\perp\\ &\quad + G_x(\tilde{x}, \tilde{p}) \Delta x_\parallel + G_p(\tilde{x}, \tilde{p}) \Delta p_\parallel\\ &= G(\tilde{x}, \tilde{p}) + G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)} + c_0 \left(G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)} \right)\\ &\quad + \sum_{j=1}^{l} a_j G_x(\tilde{x}, \tilde{p}) \phi_j + \sum_{j=1}^{l} a_j c_j \left(G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)} \right)\\ &= G(\tilde{x}, \tilde{p}) + G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)} - G(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)}\\ &\quad + \sum_{j=1}^{l} a_j \left(G_x(\tilde{x}, \tilde{p}) \phi_j - G_x(\tilde{x}, \tilde{p}) \phi_j \right)\\ &= 0. \end{aligned}$$

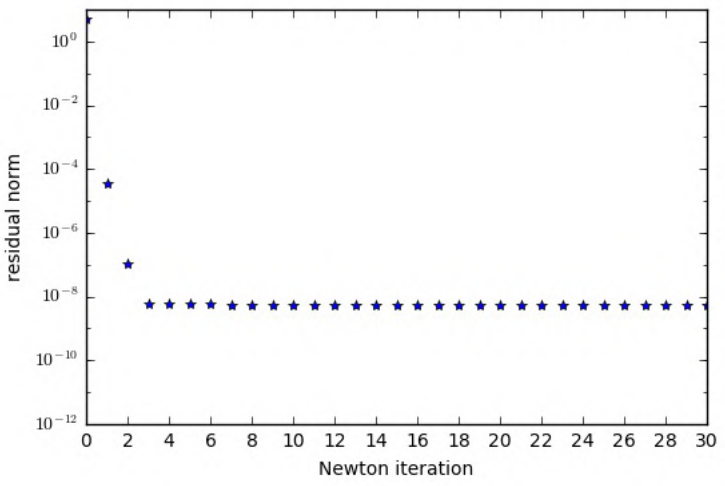


Figure 5.7: Residual plot of example 5.10. The SBN method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm decreases to approximately 10^{-8} after 3 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps.

The *split block Newton method* (SBN) is described by algorithm 5.4 on page 159 in appendix 5.10. Each Newton step starts with the approximation of null vectors of $F_x(\tilde{x}, \tilde{p})$ (by application of algorithm 3.3, see page 43), which are used as deflation vectors for solving the linear systems (5.38) and (5.39) (by application of algorithm 3.4, see page 44). The values $c_0, c_1, \ldots, c_l$ are calculated (see (5.40) and (5.41)), and the values $a_1, \ldots, a_l$ are determined by minimizing (5.60), the preconditioned version of (5.46). The guesses are eventually updated by the vectors Δx and Δp defined by (5.43) and (5.45), before the next Newton iteration is started.

5.5.2 Analysis of convergence for ill-conditioned problems

The method is applied in example 5.10. Slow convergence is still observed.

Example 5.10. Consider the same set-up as in example 5.2. The SBN method is applied, this yields the residual plot of figure 5.7. The residual norm remains at approximately 10^{-8} after 3 Newton iterations. ◇

Lemma 5.11. When applied to a problem with a (near) singular partial Jacobian F_x in the solution, convergence of the SBN method typically slows down after some initial iterations. This is caused by the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim |\lambda_s| \tag{5.47}$$

typically being satisfied by the $a_1, \ldots, a_l$ values in the update vectors (5.43) and (5.45). The index s in (5.47) is chosen such that $|a_s \lambda_s|$ is maximal. $\lambda_1, \ldots, \lambda_l$ are the eigenvalues of approximate null vectors, so we have $\lambda_s \approx 0$, inducing small sized updates.

Proof. The proof is similar to the one of lemma 4.11 in section 4.5.2. The cause of slow convergence will again be clarified by looking at a Taylor expansion. We first split the current residual $F(\tilde{x}, \tilde{p})$ and partial derivative $F_p(\tilde{x}, \tilde{p})$ in

$$F(\tilde{x}, \tilde{p}) \quad = \quad F_\perp + F_\parallel \quad = \quad F_\perp + \sum_{j=1}^{l} b_j^{(1)} \phi_j, \tag{5.48}$$

$$F_p(\tilde{x}, \tilde{p}) \quad = \quad F_{p\perp} + F_{p\parallel} \quad = \quad F_{p\perp} + \sum_{j=1}^{l} b_j^{(2)} \phi_j, \tag{5.49}$$

with $b_j^{(1)} = \langle F(\tilde{x}, \tilde{p}), \phi_j \rangle$, $b_j^{(2)} = \langle F_p(\tilde{x}, \tilde{p}), \phi_j \rangle$, $\langle F_\perp, \phi_j \rangle = 0$ and $\langle F_{p\perp}, \phi_j \rangle = 0$ $(\forall j = 1, \ldots, l)$. A second order Taylor expansion for the new residual eventually yields

$$\begin{aligned}
&F(\tilde{x} + \Delta x, \tilde{p} + \Delta p) \\
&\quad = \quad \sum_{j=1}^{l} \left(b_j^{(1)} + \Delta p(a) b_j^{(2)} + a_j \lambda_j \right) \phi_j + \frac{1}{2} F_{xx}(\tilde{x}, \tilde{p}) y_\perp^{(1)} y_\perp^{(1)} \\
&\qquad + \Delta p(a) \left(F_{xp}(\tilde{x}, \tilde{p}) y_\perp^{(1)} - F_{xx}(\tilde{x}, \tilde{p}) y_\perp^{(1)} y_\perp^{(2)} \right) + \sum_{j=1}^{l} a_j F_{xx}(\tilde{x}, \tilde{p}) y_\perp^{(1)} \phi_j \\
&\qquad + \Delta p(a)^2 \left(\frac{1}{2} F_{xx}(\tilde{x}, \tilde{p}) y_\perp^{(2)} y_\perp^{(2)} - F_{xp}(\tilde{x}, \tilde{p}) y_\perp^{(2)} + \frac{1}{2} F_{pp}(\tilde{x}, \tilde{p}) \right) \\
&\qquad + \Delta p(a) \sum_{j=1}^{l} a_j \left(F_{xp}(\tilde{x}, \tilde{p}) \phi_j - F_{xx}(\tilde{x}, \tilde{p}) y_\perp^{(2)} \phi_j \right) \\
&\qquad + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x}, \tilde{p}) \phi_i \phi_j + \mathcal{O}(\|\Delta x\|^3) + \mathcal{O}(\Delta p^3)
\end{aligned}$$

with $\Delta p(a) = c_0 + \sum_{j=1}^{l} a_j c_j$. To completely remove the approximate null vector part $\sum_{j=1}^{l} \left(b_j^{(1)} + \Delta p(a) b_j^{(2)} + a_j \lambda_j \right) \phi_j$ of the current residual, the choice

$$\forall j = 1, \ldots, l : a_j = -\frac{b_j^{(1)}}{\lambda_j} - \frac{-G(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)} + \sum_{j=1}^{l} \frac{b_j^{(1)}}{\lambda_j} G_x(\tilde{x}, \tilde{p}) \phi_j}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)} - \sum_{j=1}^{l} \frac{b_j^{(2)}}{\lambda_j} G_x(\tilde{x}, \tilde{p}) \phi_j} \frac{b_j^{(2)}}{\lambda_j} \tag{5.50}$$

would need to be made. This choice is typically of order $\mathcal{O}(\lambda_j^{-1})$ for each a_j $(\forall j = 1, \ldots, l)$. Since $|\lambda_j| \ll 1$ $(\forall j = 1, \ldots, l)$, (5.50) would however imply a blow-up of second order derivative terms $\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} a_i a_j F_{xx}(\tilde{x}, \tilde{p}) \phi_i \phi_j$. To prevent dominance of this term in the new residual,

$$\max_{i,j=1,\ldots,l} |a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \lambda_j| \tag{5.51}$$

is required, leading to the bound (5.47). This bound prevents $a_1, \ldots, a_l$ to be chosen near the values (5.50). The approximate null vector part of the residual is only decreased slightly in each Newton step, resulting in slow convergence.

□

Contrary to the method with line search (described in section 5.4.1), the $y_\perp^{(1)}$ term of Δx is not damped when the update vector is constructed. This term is responsible for decreasing the part $F_\perp$ of the residual $F(\tilde{x}, \tilde{p})$ perpendicular to $\phi_1, \ldots, \phi_l$, damping it was one of the causes of slow convergence mentioned in lemma 5.9. The residual part $F_\perp$ decreases quadratically for the SBN method.

As shown in lemma 5.11, the second cause of slow convergence mentioned in lemma 5.9 is, however, not resolved by splitting the update vectors by (5.43) and (5.45). The SBN method still converges slowly, as indicated by example 5.10.

5.5.3 Minimization of the residual norm

In practice the values for $a_1, \ldots, a_l$ are calculated by minimizing (5.46) with the nonlinear conjugate gradients method. The required derivatives are given for the preconditioned case in section 5.5.4 (see (5.61) and (5.62)). To create an initial guess $a^{(0)}$ for the minimization, we first minimize a different function given by

$$\begin{aligned} f : \mathbb{R}^l \to \mathbb{R} : a' \to & \sum_{j=1}^{l} \left((b_j^{(1)} + \Delta p(a') b_j^{(2)} + a'_j \lambda_j \right)^2 + \left(\Delta p(a') \| y_\perp^{(1)} \| \right)^2 \\ & + \sum_{j=1}^{l} \left(a'_j \| y_\perp^{(1)} \| \right)^2 + \Delta p(a')^4 + \sum_{j=1}^{l} \left(\Delta p(a') a'_j \right)^2 \\ & + \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a'_i a'_j \right)^2 \end{aligned} \tag{5.52}$$

with $\Delta p(a') = c_0 + \sum_{j=1}^{l} a'_j c_j$. The values $b_1^{(1)}, \ldots, b_l^{(1)}$ and $b_1^{(2)}, \ldots, b_l^{(2)}$ are defined by respectively (5.48) and (5.49). This extra minimization is executed with the nonlinear conjugate gradients method as well.

As an initial guess $a'^{(0)}$ for the minimization of f, we first calculate the values

$$\forall j = 1, \ldots, l : a_j'^{(r)} = -\frac{b_j^{(1)}}{\lambda_j} \text{ and } a_j'^{(p)} = -\frac{b_j^{(2)}}{\lambda_j}. \tag{5.53}$$

Denoting s the index for which $|b_s^{(1)}|$ is maximal, the values

$$a_s'^{(q)} = \text{sgn}(a_s'^{(r)}) \min\left(|a_s'^{(r)}|, |\lambda_s|, \frac{|b_s^{(1)}|}{\max\left(|c^{(p)}|, \|y_\perp^{(1)}\|\right)}\right),$$
$$\forall j \neq s : a_j'^{(q)} = \text{sgn}(a_j'^{(r)}) \min\left(|a_j'^{(r)}|, |a_s'^{(q)}|\right),$$
$$\text{with } c^{(p)} = \frac{-G(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)} + \sum_{j=1}^{l} a_j'^{(p)} G_x(\tilde{x},\tilde{p})\phi_j}, \tag{5.54}$$

are calculated. Denoting

$$\chi = \min\left(1, \frac{|\lambda_s|}{|c^{(q)}|}, \frac{|\lambda_s||a_s'^{(q)}|}{c^{(q)2}}, \frac{|b_s^{(1)}|}{|c^{(q)}|\max\left(|c^{(p)}|, \|y_\perp^{(1)}\|\right)}\right), \tag{5.55}$$

$$\text{with } c^{(q)} = \frac{-\sum_{j=1}^{l} a_j'^{(q)} G_x(\tilde{x},\tilde{p})\phi_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)} + \sum_{j=1}^{l} a_j'^{(p)} G_x(\tilde{x},\tilde{p})\phi_j}, \tag{5.56}$$

the guess $a'^{(0)}$ is eventually created by

$$\forall j = 1, \ldots, l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)}, \tag{5.57}$$
$$\text{with } \Delta p^{(0)} = \frac{-G(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(1)} - \sum_{j=1}^{l} \chi a_j'^{(q)} G_x(\tilde{x},\tilde{p})\phi_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)} + \sum_{j=1}^{l} a_j'^{(p)} G_x(\tilde{x},\tilde{p})\phi_j}. \tag{5.58}$$

The guess $a'^{(0)}$ is constructed in such a way that none of the separate terms of (5.52) are blown up.

5.5.4 The preconditioned case

Changes required for the preconditioned method are similar as before. The eigenvectors $\phi_1, \ldots, \phi_l$ and corresponding -values $\lambda_1, \ldots, \lambda_l$ of the preconditioned partial Jacobian $P(\tilde{x},\tilde{p})F_x(\tilde{x},\tilde{p})$ are considered, and vectors $\tau_1, \ldots, \tau_l$ are defined such that $P(\tilde{x},\tilde{p})\tau_j = \phi_j$ $(\forall j = 1, \ldots, l)$. The terms $y_\perp^{(1)}$ and $y_\perp^{(2)}$, that appear in the update vector, are calculated by solving the preconditioned deflated linear systems, given by (5.24) and (5.25). The projection operator used in these systems was given by

$$\mathcal{Q}(\tilde{x},\tilde{p}) : \mathbb{C}^n \to \mathbb{C}^n : w \to w - K\langle K, L\rangle^{-1}\langle L, w\rangle, \tag{5.59}$$

with

$$K = \begin{pmatrix}\tau_1 & \tau_2 & \ldots & \tau_l\end{pmatrix},$$
$$L = \begin{pmatrix}\phi_1 & \phi_2 & \ldots & \phi_l\end{pmatrix},$$

such that $P(\tilde{x},\tilde{p})K = L$. The values $c_0, c_1, \ldots, c_l$ are still calculated by (5.40) and (5.41). Update vectors of the form (5.43) and (5.45) are again considered,

where the values $a_1, \ldots, a_l$ are calculated by minimizing

$$g : \mathbb{R}^l \to \mathbb{R} : a \to \left\| F\left(\tilde{x} + y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right), \right. \right. \\ \left. \left. \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j \right) \right\|^2_{P(\tilde{x},\tilde{p})} \tag{5.60}$$

with the nonlinear conjugate gradients method. The first and second partial derivatives of this function are given by $(\forall i, k = 1, \ldots, l)$

$$\frac{\partial g}{\partial a_i} : \mathbb{R}^l \to \mathbb{R} : a \to 2\langle F(x(a), p(a)), F_X(x(a), p(a))\Phi_i \rangle_{P(\tilde{x},\tilde{p})} \tag{5.61}$$

$$\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \to \mathbb{R} : a \to 2\langle F_X(x(a), p(a))\Phi_i, F_X(x(a), p(a))\Phi_k \rangle_{P(\tilde{x},\tilde{p})} \\ + 2\langle F(x(a), p(a)), F_{XX}(x(a), p(a))\Phi_i\Phi_k \rangle_{P(\tilde{x},\tilde{p})}, \tag{5.62}$$

$$\text{with } \forall i = 1, \ldots, l : \Phi_i = \begin{pmatrix} \phi_i - c_i y_\perp^{(2)} \\ c_i \end{pmatrix},$$

$$x(a) = \tilde{x} + y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right)$$

$$\text{and } p(a) = \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j.$$

To create an initial guess for the minimization of g, first the function

$$f : \mathbb{R}^l \to \mathbb{R} : a' \to \sum_{j=1}^{l} \left((b_j^{(1)} + \Delta p(a') b_j^{(2)} + a'_j \lambda_j \right)^2 + \left(\Delta p(a') \| y_\perp^{(1)} \|_{P(\tilde{x},\tilde{p})^{-1}} \right)^2 \\ + \sum_{j=1}^{l} \left(a'_j \| y_\perp^{(1)} \|_{P(\tilde{x},\tilde{p})^{-1}} \right)^2 + \Delta p(a')^4 + \sum_{j=1}^{l} \left(\Delta p(a') a'_j \right)^2 \\ + \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a'_i a'_j \right)^2, \tag{5.63}$$

$$\text{with } \Delta p(a') = c_0 + \sum_{j=1}^{l} a'_j c_j,$$

is minimized, with $b_j^{(1)} = \langle F(\tilde{x}, \tilde{p}), \phi_j \rangle$ and $b_j^{(2)} = \langle F_p(\tilde{x}, \tilde{p}), \phi_j \rangle$ $(\forall j = 1, \ldots, l)$. As an initial guess to minimize f, we use the values

$$\forall j = 1, \ldots, l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)}, \tag{5.64}$$

with $\Delta p^{(0)}$, $c^{(q)}$, $c^{(p)}$, $a_1'^{(r)}, \ldots, a_l'^{(r)}$ and $a_1'^{(p)}, \ldots, a_l'^{(p)}$ defined by (5.58), (5.56), (5.54) and (5.53), $a_1'^{(q)}, \ldots, a_l'^{(q)}$ given by

$$a_s'^{(q)} = \operatorname{sgn}(a_s'^{(r)}) \min\left(|a_s'^{(r)}|, |\lambda_s|, \frac{|b_s^{(1)}|}{\max\left(|c^{(p)}|, \|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}} \right)} \right),$$
$$\forall j \neq s : a_j'^{(q)} = \operatorname{sgn}(a_j'^{(r)}) \min\left(|a_j'^{(r)}|, |a_s'^{(q)}| \right),$$

with s again the index for which $|b_s^{(1)}|$ is maximal, and χ given by

$$\chi = \min\left(1, \frac{|\lambda_s|}{|c^{(q)}|}, \frac{|\lambda_s||a_s'^{(q)}|}{c^{(q)2}}, \frac{|b_s^{(1)}|}{|c^{(q)}| \max\left(|c^{(p)}|, \|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}} \right)} \right).$$

5.6 Addition of extra terms to the update vectors

5.6.1 Description of the method

To improve the convergence of the split block Newton method (SBN), we will again introduce additional parts to the update vectors Δx and Δp. With $\tilde{x}$, $\tilde{p}$ a guess for the wanted zero of F and G, and $\phi_1, \ldots, \phi_l$ approximate null vectors of $F_x(\tilde{x}, \tilde{p})$ (with respective eigenvalues $\lambda_1, \ldots, \lambda_l$), we first split second order partial derivative applications to $\phi_1, \ldots, \phi_l$:

$$\forall i, j = 1, \ldots, l : F_{xx}(\tilde{x}, \tilde{p})\phi_i\phi_j = F_\perp^{(ij)} + F_\parallel^{(ij)} \tag{5.65}$$
$$= F_\perp^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)} \phi_k, \tag{5.66}$$

with $\langle F_\perp^{(ij)}, \phi_k \rangle = 0$ $(\forall i, j = 1, \ldots, l)$ and $c_k^{(ij)} = \langle F_{xx}(\tilde{x}, \tilde{p})\phi_i\phi_j, \phi_k \rangle$ $(\forall i, j, k = 1, \ldots, l)$.

The vectors z_{11}, z_{12}, …, z_{ll} are introduced as the solutions of the linear systems $(\forall i, j = 1, \ldots, l)$

$$\mathcal{Q}(\tilde{x}, \tilde{p}) F_x(\tilde{x}, \tilde{p}) z_{ij} = \mathcal{Q}(\tilde{x}, \tilde{p}) F_{xx}(\tilde{x}, \tilde{p})\phi_i\phi_j, \tag{5.67}$$

where the operator $\mathcal{Q}(\tilde{x}, \tilde{p})$ is given by (5.37).With $c_0, c_1, \ldots, c_l$ defined as before (see (5.40) and (5.41)) and $c_{ij}^{(z)}$ given by

$$\forall i, j = 1, \ldots, l : c_{ij}^{(z)} = \frac{-G_x(\tilde{x}, \tilde{p}) z_{ij}}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}, \tag{5.68}$$

the update vectors Δx and Δp used in the new method are given by

$$\Delta x = \Delta x_\perp + \Delta x_\parallel + \Delta x_z \tag{5.69}$$

$$\begin{aligned} &= y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)}\right) \\ &\quad + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} \left(z_{ij} - c_{ij}^{(z)} y_\perp^{(2)}\right), \end{aligned} \tag{5.70}$$

$$\Delta p = \Delta p_\perp + \Delta p_\parallel + \Delta p_z \tag{5.71}$$

$$= c_0 + \sum_{j=1}^{l} a_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} c_{ij}^{(z)}. \tag{5.72}$$

The vectors $y_\perp^{(1)}$ and $y_\perp^{(2)}$ are defined as the solutions of respectively (5.38) and (5.39), the values of $a_1, \ldots, a_l$ and $\alpha_{11}, \ldots, \alpha_{ll}$ will be calculated by minimizing the residual norm (see section 5.6.3).

The method described in the current section will be called the *split block Newton method with extra terms* (SBNE), pseudo-code is given by algorithm 5.5 on page 160 in appendix 5.10. Each Newton iteration starts by approximating null vectors of the partial Jacobian $F_x(\tilde{x}, \tilde{p})$ (using algorithm 3.3, see page 43), followed by solving linear systems (5.38), (5.39) and (5.67) (using algorithm 3.4, see page 44). After minimizing the function given by (5.82) (the preconditioned version of the residual norm) with nonlinear conjugate gradients (using algorithm 3.5, see page 45), the update vectors are constructed by (5.70) and (5.72). The guesses are updated and the next Newton iteration is started.

Just as was the case for the method without block elimination (see section 4.6), the adjustments made in the current section require solving an additional $\frac{1}{2}l(l+1)$ linear systems in each Newton iteration. This amount should be reduced for the method to become practical. A technique for this purpose will be discussed in section 5.7.

5.6.2 Analysis of convergence for ill-conditioned problems

The SBNE method is tested in example 5.12.

Example 5.12. Consider the same set-up as in example 5.2. Application of the SBNE method yields the residual plot of figure 5.8. Convergence (up to a tolerance of approximately 10^{-12}) is achieved after 5 Newton iterations. ◇

Compared to previous methods, the SBNE method requires considerably less Newton iterations to reach convergence. The bound discussed in lemma 5.11, which caused slow convergence for the SBN method, does not apply to the SBNE one.

Lemma 5.13. By introducing the additional parts Δx_z and Δp_z, defined in (5.70) and (5.72), the bound (5.47) discussed in lemma 5.11 is eliminated.

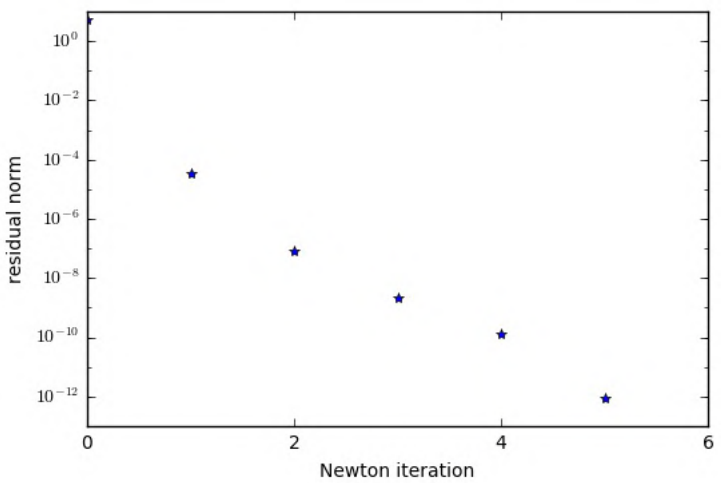


Figure 5.8: Residual plot of example 5.12. The SBNE method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-12} after 5 iterations, which is the attainable accuracy.

Proof. We split the terms $F(\tilde{x},\tilde{p})$ and $F_p(\tilde{x},\tilde{p})$ as in (5.48) and (5.49). Performing a Taylor expansion on the new residual, calculated by updating the guess $\tilde{x}$, $\tilde{p}$ with (5.70) and (5.72), eventually yields

$$
\begin{aligned}
&F(\tilde{x}+\Delta x,\tilde{p}+\Delta p)\\
&\quad= \sum_{j=1}^{l}\left(b_j^{(1)}+\Delta p(a,\alpha)b_j^{(2)}+a_j\lambda_j+\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}c_j^{(ik)}a_ia_j\right)\phi_j\\
&\qquad+\frac{1}{2}F_{xx}(\tilde{x},\tilde{p})y_\perp^{(1)}y_\perp^{(1)}+\Delta p(a,\alpha)\left(F_{xp}(\tilde{x},\tilde{p})y_\perp^{(1)}-F_{xx}(\tilde{x},\tilde{p})y_\perp^{(1)}y_\perp^{(2)}\right)\\
&\qquad+\sum_{j=1}^{l}a_jF_{xx}(\tilde{x},\tilde{p})y_\perp^{(1)}\phi_j+\sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_{ij}+\frac{1}{2}a_ia_j\right)F_\perp^{(ij)}\\
&\qquad+\Delta p(a,\alpha)^2\left(\frac{1}{2}F_{xx}(\tilde{x},\tilde{p})y_\perp^{(2)}y_\perp^{(2)}-F_{xp}(\tilde{x},\tilde{p})y_\perp^{(2)}+\frac{1}{2}F_{pp}(\tilde{x},\tilde{p})\right)\\
&\qquad+\Delta p(a,\alpha)\sum_{j=1}^{l}a_j\left(F_{xp}(\tilde{x},\tilde{p})\phi_j-F_{xx}(\tilde{x},\tilde{p})y_\perp^{(2)}\phi_j\right)+R(\tilde{x},\tilde{p},\Delta x,\Delta p),
\end{aligned}
$$

with the remainder $R(\tilde{x}, \tilde{p}, \Delta x, \Delta p)$ given by

$$\begin{aligned} R(\tilde{x}, \tilde{p}, \Delta x, \Delta p) &= F_{xx}(\tilde{x}, \tilde{p}) y_{\perp}^{(1)} \Delta x_z + F_{xx}(\tilde{x}, \tilde{p}) \Delta x_{\parallel} \Delta x_z \\ &\quad + \Delta p \left(F_{xp}(\tilde{x}, \tilde{p}) \Delta x_z - F_{xx}(\tilde{x}, \tilde{p}) y_{\perp}^{(2)} \Delta x_z \right) \\ &\quad + \frac{1}{2} F_{xx}(\tilde{x}, \tilde{p}) \Delta x_z \Delta x_z + \frac{1}{6} F_{xxx}(\tilde{x}, \tilde{p}) \Delta x \Delta x \Delta x \\ &\quad + \frac{1}{2} F_{xxp}(\tilde{x}, \tilde{p}) \Delta x \Delta x \Delta p + \frac{1}{2} F_{xpp}(\tilde{x}, \tilde{p}) \Delta x \Delta p \Delta p \\ &\quad + \frac{1}{6} F_{ppp}(\tilde{x}, \tilde{p}) \Delta p \Delta p \Delta p + \mathcal{O}(\|\Delta x\|^4) + \mathcal{O}(\Delta p^4), \end{aligned}$$

and $\Delta p(a, \alpha) = c_0 + \sum_{j=1}^{l} a_j c_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_{ij} c_{ij}^{(z)}$.

The influence of the second order partial derivative terms $F_{xx}(\tilde{x}, \tilde{p}) \phi_i \phi_j$ (for $i, j = 1, \ldots, l$) on the new residual, which was the cause of condition (5.51), is strongly reduced by choosing

$$\forall i, j = 1, \ldots, l : \alpha_{ij} \approx -\frac{1}{2} a_i a_j, \tag{5.73}$$

which eliminates terms $\frac{1}{2} a_i a_j F_{\perp}^{(ij)}$ (with $\forall i, j = 1, \ldots, l$) from the Taylor expansion. The null vector part of the $F_{xx}(\tilde{x}, \tilde{p}) \phi_i \phi_j$ terms are reduced as well, by incorporating these in the choice for $a_1, \ldots, a_l$. By using the update vectors (5.70) and (5.72), condition (5.51) is eliminated. □

The bound of lemma 5.11 is eliminated by a specific choice for the values $\alpha_{11}, \ldots, \alpha_{ll}$ in the additional parts Δx_z and Δp_z of the update vectors. Just as in the case without block elimination (see section 4.6.2), a similar bound to the values of $a_1, \ldots, a_l$ however exists.

Lemma 5.14. The values $a_1, \ldots, a_l$, that appear in the update vectors (5.70) and (5.72) of the SBNE method, typically satisfy the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt{|\lambda_s|}. \tag{5.74}$$

The index s is chosen such that $|a_s \lambda_s|$ is maximal.

Proof. Consider the same Taylor expansion as in the proof of lemma 5.13. From the part

$$\frac{1}{6} \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} F_{xxx}(\tilde{x}, \tilde{p}) \phi_k \phi_i \phi_j$$

the condition

$$\max_{k,i,j=1,\ldots,l} |a_k a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \lambda_j|. \tag{5.75}$$

is derived. Denoting s the index for which $|a_s \lambda_s|$ is maximal, (5.74) represents a necessary condition to satisfy this approximate inequality. □

The bound discussed in lemma 5.14 is however not as strict as the one in lemma 5.11: higher values for $a_1, \ldots, a_l$ can be chosen, leading to a bigger decrease of the current residual. Faster convergence is expected.

Remark 5.15. Note that the part

$$\Delta p(a,\alpha)^2 \left(\frac{1}{2}F_{xx}(\tilde{x},\tilde{p})y_\perp^{(2)}y_\perp^{(2)} - F_{xp}(\tilde{x},\tilde{p})y_\perp^{(2)} + \frac{1}{2}F_{pp}(\tilde{x},\tilde{p})\right)$$

of the Taylor expansion in the proof of lemma 5.13 also induces a condition on a and α, given by

$$\Delta p(a,\alpha)^2 \lesssim \max_{j=1,\ldots,l} |a_j\lambda_j|. \tag{5.76}$$

It is possible to relax this condition by using alternative additional parts. Doing so however also leads to a slightly stricter bound on $a_1,\ldots,a_l$ than (5.74). For completeness, this alternative approach is described in section 5.6.5. It will however not be used for any of the numerical experiments performed in chapter 9.

5.6.3 Minimization of the residual norm

The values for $a_1,\ldots,a_l$ and $\alpha_{11},\ldots,\alpha_{ll}$ in (5.70) and (5.72) had not yet been derived. In practice these are calculated by minimizing

$$\begin{aligned} g :&\mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ (a,\alpha) \to &\left\| F\left(\tilde{x} + y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j\left(\phi_j - c_j y_\perp^{(2)}\right)\right.\right. \\ &\qquad + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}\left(z_{ij} - c_{ij}^{(z)}y_\perp^{(2)}\right), \\ &\qquad \left.\left. \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}c_{ij}^{(z)}\right)\right\|^2 \end{aligned} \tag{5.77}$$

with the nonlinear conjugate gradients method. Partial derivatives of g are derived for the preconditioned method in section 5.6.4. Given the initial guesses for $a_1,\ldots,a_l$, the ones for $\alpha_{11},\ldots,\alpha_{ll}$ are calculated by $\alpha_{ij}^{(0)} = -\frac{1}{2}a_i^{(0)}a_j^{(0)}$ $(\forall i,j = 1,\ldots,l)$. The ones for $a_1,\ldots,a_l$ are calculated by minimizing the function

$$\begin{aligned} f : \mathbb{R}^l \to \mathbb{R} : a' \to &\sum_{j=1}^{l}\left(b_j^{(1)} + \Delta p(a')b_j^{(2)} + a'_j\lambda_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}c_j^{(ik)}a'_ia'_j\right)^2 \\ &+ \left(\Delta p(a')\|y_\perp^{(1)}\|\right)^2 + \sum_{j=1}^{l}\left(a'_j\|y_\perp^{(1)}\|\right)^2 + \Delta p(a')^4 \\ &+ \sum_{j=1}^{l}\left(\Delta p(a')a'_j\right)^2 + \sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(a'_ka'_ia'_j\right)^2, \end{aligned} \tag{5.78}$$

with $\Delta p(a') = c_0 + \sum_{j=1}^{l} a'_jc_j - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}a'_ia'_jc_{ij}^{(z)}$.

The values $b_1^{(1)}, \ldots, b_l^{(1)}$, $b_1^{(2)}, \ldots, b_l^{(2)}$ and $c_k^{(ij)}$ (for $i, j, k = 1, \ldots, l$) are defined as in (5.48), (5.49) and (5.66).

In practice f is minimized with nonlinear conjugate gradients as well, requiring an initial guess itself. We again calculate the values

$$\forall j = 1, \ldots, l : a_j'^{(r)} = -\frac{b_j^{(1)}}{\lambda_j} \text{ and } a_j'^{(p)} = -\frac{b_j^{(2)}}{\lambda_j}.$$

With s the index for which $|b_s^{(1)}|$ is maximal, the values

$$\begin{aligned} a_s'^{(q)} &= \operatorname{sgn}(a_s'^{(r)}) \min\left(|a_s'^{(r)}|, \sqrt{|\lambda_s|}, \frac{|b_s^{(1)}|}{\max\left(|c^{(p)}|, \|y_\perp^{(1)}\|\right)} \right), \\ \forall j \neq s : a_j'^{(q)} &= \operatorname{sgn}(a_j'^{(r)}) \min\left(|a_j'^{(r)}|, |a_s'^{(q)}| \right), \end{aligned}$$

are calculated, with $c^{(p)}$ defined as the lowest magnitude solution of the quadratic equation

$$c^{(p)} = c_0 + \sum_{j=1}^{l} c_j c^{(p)} a_j'^{(p)} - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} c_{ij}^{(z)} c^{(p)2} a_i'^{(p)} a_j'^{(p)}.$$

The value

$$\chi = \min\left(1, \frac{|\lambda_s|}{|c^{(q)}|}, \frac{|\lambda_s||a_s'^{(q)}|}{c^{(q)2}}, \frac{|b_s^{(1)}|}{|c^{(q)}| \max\left(|c^{(p)}|, \|y_\perp^{(1)}\|\right)} \right)$$

is calculated, with $c^{(q)}$ defined further in the section. The eventual guess $a'^{(0)}$ is constructed by

$$\forall j = 1, \ldots, l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)}, \tag{5.79}$$

with $\Delta p^{(0)}$ the lowest magnitude solution of the quadratic equation

$$\begin{aligned} \Delta p^{(0)} \;=\; & c_0 + \sum_{j=1}^{l} c_j \left(\chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)} \right) \\ & - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} c_{ij}^{(z)} \left(\chi a_i'^{(q)} + \Delta p^{(0)} a_i'^{(p)} \right) \left(\chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)} \right). \end{aligned} \tag{5.80}$$

To calculate $c^{(q)}$, the lowest magnitude solution of (5.80) needs to be regarded as a function of χ. The maximum of the derivative of this function over an interval $[0; 1]$ yields the required value for $c^{(q)}$.

As before, the guess $a'^{(0)}$ is constructed in such a way that none of the terms of (5.78) yield a blow-up.

5.6.4 The preconditioned case

For the preconditioned method the eigenvectors $\phi_1, \ldots, \phi_l$ and -values $\lambda_1, \ldots, \lambda_l$ of $P(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})$ are considered, and vectors $\tau_1, \ldots, \tau_l$ such that $P(\tilde{x}, \tilde{p})\tau_j = \phi_j$ $(\forall j = 1, \ldots, l)$ are defined. The vectors $y_\perp^{(1)}$ and $y_\perp^{(2)}$ are still calculated as the solutions of (5.24) and (5.25), $z_{11}, \ldots, z_{ll}$ by solving the preconditioned deflated linear systems $(\forall i, j = 1, \ldots, l)$

$$P(\tilde{x}, \tilde{p})\mathcal{Q}(\tilde{x}, \tilde{p})F_x(\tilde{x}, \tilde{p})z_{ij} = P(\tilde{x}, \tilde{p})\mathcal{Q}(\tilde{x}, \tilde{p})F_{xx}(\tilde{x}, \tilde{p})\phi_i\phi_j, \tag{5.81}$$

with $\mathcal{Q}(\tilde{x}, \tilde{p})$ given by (5.59). The values c_0, $c_1, \ldots, c_l$ and $c_{ij}^{(z)}$ $(\forall i, j = 1, \ldots, l)$ are again calculated by application of (5.40), (5.41) and (5.68), the update vectors are defined by (5.70) and (5.72).

To calculate $a_1, \ldots, a_l$ and $\alpha_{11}, \ldots, \alpha_{ll}$, the function

$$\begin{aligned} g :& \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ & (a, \alpha) \to \left\| F\left(\tilde{x} + y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right) \right.\right. \\ & \qquad + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} \left(z_{ij} - c_{ij}^{(z)} y_\perp^{(2)} \right), \\ & \qquad \left.\left. \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} c_{ij}^{(z)} \right) \right\|_{P(\tilde{x},\tilde{p})}^2 \end{aligned} \tag{5.82}$$

is minimized with the nonlinear conjugate gradients method. The first partial derivatives of g are given by $(\forall i, j = 1, \ldots, l)$

$$\begin{aligned} \frac{\partial g}{\partial a_i} :& \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ & (a, \alpha) \to 2\langle F(x(a, \alpha), p(a, \alpha)), F_X(x(a, \alpha), p(a, \alpha))\Phi_i \rangle_{P(\tilde{x},\tilde{p})}, \\ \frac{\partial g}{\partial \alpha_{ij}} :& \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\ & (a, \alpha) \to 2\langle F(x(a, \alpha), p(a, \alpha)), F_X(x(a, \alpha), p(a, \alpha))Z_{ij} \rangle_{P(\tilde{x},\tilde{p})}, \end{aligned} \tag{5.83}$$

with

$$\begin{aligned} \forall i = 1, \ldots, l : \Phi_i &= \begin{pmatrix} \phi_i - c_i y_\perp^{(2)} \\ c_i \end{pmatrix}, \\ \forall i, j = 1, \ldots, l : Z_{ij} &= \begin{pmatrix} z_{ij} - c_{ij}^{(z)} y_\perp^{(2)} \\ c_{ij}^{(z)} \end{pmatrix}, \\ x(a, \alpha) = \tilde{x} &+ y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j \left(\phi_j - c_j y_\perp^{(2)} \right) \\ &+ \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} \left(z_{ij} - c_{ij}^{(z)} y_\perp^{(2)} \right), \end{aligned}$$

$$\text{and } p(a,\alpha) = \tilde{p} + c_0 + \sum_{j=1}^{l} a_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} c_{ij}^{(z)}.$$

The second by $(\forall i,j,k,q = 1,\ldots,l)$

$$\begin{aligned}
&\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\
&\qquad (a,\alpha) \to 2\langle F_X(x(a,\alpha),p(a,\alpha))\Phi_i, F_X(x(a,\alpha),p(a,\alpha))\Phi_k\rangle_{P(\tilde{x},\tilde{p})} \\
&\qquad\qquad + 2\langle F(x(a,\alpha),p(a,\alpha)), F_{XX}(x(a,\alpha),p(a,\alpha))\Phi_i\Phi_k\rangle_{P(\tilde{x},\tilde{p})}, \\
&\frac{\partial^2 g}{\partial a_i \partial \alpha_{jk}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\
&\qquad (a,\alpha) \to 2\langle F_X(x(a,\alpha),p(a,\alpha))\Phi_i, F_X(x(a,\alpha),p(a,\alpha))Z_{jk}\rangle_{P(\tilde{x},\tilde{p})} \\
&\qquad\qquad + 2\langle F(x(a,\alpha),p(a,\alpha)), F_{XX}(x(a,\alpha),p(a,\alpha))\Phi_i Z_{jk}\rangle_{P(\tilde{x},\tilde{p})}, \\
&\frac{\partial^2 g}{\partial \alpha_{ij} \partial \alpha_{kq}} : \mathbb{R}^l \times \mathbb{R}^{l\times l} \to \mathbb{R} : \\
&\qquad (a,\alpha) \to 2\langle F_X(x(a,\alpha),p(a,\alpha))Z_{ij}, F_X(x(a,\alpha),p(a,\alpha))Z_{kq}\rangle_{P(\tilde{x},\tilde{p})} \\
&\qquad\qquad + 2\langle F(x(a,\alpha),p(a,\alpha)), F_{XX}(x(a,\alpha),p(a,\alpha))Z_{ij}Z_{kq}\rangle_{P(\tilde{x},\tilde{p})}.
\end{aligned} \tag{5.84}$$

Given an initial guess for $a_1,\ldots,a_l$, the ones for $\alpha_{11},\ldots,\alpha_{ll}$ are again calculated by $\alpha_{ij}^{(0)} = -\frac{1}{2}a_i a_j$ $(\forall i,j = 1,\ldots,l)$. Defining $b_j^{(1)} = \langle F(\tilde{x},\tilde{p}),\phi_j\rangle$, $b_j^{(2)} = \langle F_p(\tilde{x},\tilde{p}),\phi_j\rangle$ $(\forall j = 1,\ldots,l)$ and $c_k^{(ij)} = \langle F_{xx}(\tilde{x},\tilde{p})\phi_i\phi_j,\phi_k\rangle$ $(\forall i,j,k = 1,\ldots,l)$, initial guesses $a_1^{(0)},\ldots,a_l^{(0)}$ are calculated by minimizing

$$\begin{aligned}
f : \mathbb{R}^l \to \mathbb{R} : a' \to &\sum_{j=1}^{l}\left(b_j^{(1)} + \Delta p(a') b_j^{(2)} + a'_j\lambda_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} c_j^{(ik)} a'_i a'_j\right)^2 \\
&+ \left(\Delta p(a')\|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}}\right)^2 + \sum_{j=1}^{l}\left(a'_j\|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}}\right)^2 \\
&+ \Delta p(a')^4 + \sum_{j=1}^{l}\left(\Delta p(a') a'_j\right)^2 + \sum_{k=1}^{l}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(a'_k a'_i a'_j\right)^2,
\end{aligned} \tag{5.85}$$

with

$$\Delta p(a') = c_0 + \sum_{j=1}^{l} a_j c_j - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} a_i a_j c_{ij}^{(z)},$$

with nonlinear conjugate gradients. As an initial guess for the minimization of f, we use the values

$$\forall j = 1,\ldots,l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(p)}, \tag{5.86}$$

with $\Delta p^{(0)}$, $c^{(q)}$, $c^{(p)}$, $a_1'^{(r)},\ldots,a_l'^{(r)}$ and $a_1'^{(p)},\ldots,a_l'^{(p)}$ defined similar to the unpreconditioned case. With s the index for which $|b_s^{(1)}|$ is maximal, the values

$a_1'^{(q)}, \ldots, a_l'^{(q)}$ are calculated by

$$a_s'^{(q)} = \operatorname{sgn}(a_s'^{(r)}) \min \left(|a_s'^{(r)}|, \sqrt{|\lambda_s|}, \frac{|b_s^{(1)}|}{\max\left(|c^{(p)}|, \|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}}\right)} \right),$$
$$\forall j \neq s : a_j'^{(q)} = \operatorname{sgn}(a_j'^{(r)}) \min\left(|a_j'^{(r)}|, |a_s'^{(q)}|\right).$$

The value χ is given by

$$\chi = \min \left(1, \frac{|\lambda_s|}{|c^{(q)}|}, \frac{|\lambda_s||a_s'^{(q)}|}{c^{(q)2}}, \frac{|b_s^{(1)}|}{|c^{(q)}| \max\left(|c^{(p)}|, \|y_\perp^{(1)}\|_{P(\tilde{x},\tilde{p})^{-1}}\right)} \right).$$

5.6.5 Alternative additional update parts

As mentioned in remark 5.15 (see section 5.6.2), alternative update parts can be considered, that do not lead to condition (5.76). This alternative method is derived in the current section for the unpreconditioned case. We again consider a guess $\tilde{x}$, $\tilde{p}$ and denote $\phi_1, \ldots, \phi_l$ the approximate null vectors of $F_x(\tilde{x}, \tilde{p})$. With $c_1, \ldots, c_l$ defined as before (see (5.41)), the values

$$\forall j = 1, \ldots, l : \beta_j = \sqrt{\langle \phi_j - c_j y_\perp^{(2)}, \phi_j - c_j y_\perp^{(2)} \rangle + c_j^2}^{-1}$$

are calculated. Note that $\beta_1^{-1}, \ldots, \beta_l^{-1}$ respectively equal the norms of the terms $\Phi_1, \ldots, \Phi_l$, defined by

$$\forall j = 1, \ldots, l : \Phi_j = \begin{pmatrix} \phi_j - c_j y_\perp^{(2)} \\ c_j \end{pmatrix}.$$

Instead of splitting (5.66), the second order derivatives $\beta_i \beta_j F_{XX}(\tilde{x}, \tilde{p}) \Phi_i \Phi_j$ $(\forall i, j = 1, \ldots, l)$ are split:

$$\forall i, j = 1, \ldots, l : \beta_i \beta_j F_{XX}(\tilde{x}, \tilde{p}) \Phi_i \Phi_j = F_\perp'^{(ij)} + F_\parallel'^{(ij)} = F_\perp'^{(ij)} + \sum_{k=1}^{l} c_k'^{(ij)} \phi_k. \tag{5.87}$$

This time vectors $z_{11}', \ldots, z_{ll}'$ are introduced as the solutions of $(\forall i, j = 1, \ldots, l)$

$$\mathcal{Q}(\tilde{x}, \tilde{p}) F_x(\tilde{x}, \tilde{p}) z_{ij}' = \mathcal{Q}(\tilde{x}, \tilde{p}) F_{XX}(\tilde{x}, \tilde{p}) \Phi_i \Phi_j. \tag{5.88}$$

With $c_{ij}'^{(z)}$ $(\forall i, j = 1, \ldots, l)$ defined similar to (5.68), the considered update

vectors become

$$\begin{aligned}\Delta x &= \Delta x_\perp + \Delta x_\parallel + \Delta x'_z && (5.89)\\ &= y_\perp^{(1)} - c_0 y_\perp^{(2)} + \sum_{j=1}^{l} a_j\beta_j\left(\phi_j - c_j y_\perp^{(2)}\right) \\ &\quad + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij}\left(z'_{ij} - c_{ij}^{\prime(z)} y_\perp^{(2)}\right), && (5.90)\\ \Delta p &= \Delta p_\perp + \Delta p_\parallel + \Delta p'_z && (5.91)\\ &= c_0 + \sum_{j=1}^{l} a_j\beta_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij} c_{ij}^{\prime(z)}. && (5.92)\end{aligned}$$

Lemma 5.16. By introducing the additional parts $\Delta x'_z$ and $\Delta p'_z$, defined in (5.90) and (5.92), the bound (5.47) discussed in lemma 5.11 is eliminated, without introducing (5.76).

Proof. Performing a Taylor expansion on the new guess, calculated with the alternative update vectors (5.90) and (5.92), yields

$$\begin{aligned}&F(\tilde{x}+\Delta x, \tilde{p}+\Delta p)\\ &= \sum_{j=1}^{l}\left(b_j^{(1)} + \Delta p(a,\alpha) b_j^{(2)} + a_j\beta_j\lambda_j + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} c_j^{\prime(ik)} a_i a_j\right)\phi_j\\ &\quad + \frac{1}{2}F_{XX}(\tilde{x},\tilde{p})\begin{pmatrix} y_\perp^{(1)} - c_0 y_\perp^{(2)} \\ c_0\end{pmatrix}\begin{pmatrix} y_\perp^{(1)} - c_0 y_\perp^{(2)} \\ c_0\end{pmatrix}\\ &\quad + \sum_{j=1}^{l} a_j F_{XX}(\tilde{x},\tilde{p})\begin{pmatrix} y_\perp^{(1)} - c_0 y_\perp^{(2)} \\ c_0\end{pmatrix}\begin{pmatrix} \phi_j - c_j y_\perp^{(2)} \\ c_j\end{pmatrix}\\ &\quad + \sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_{ij} + \frac{1}{2}a_i a_j\right) F_\perp^{\prime(ij)} + R(\tilde{x},\tilde{p},\Delta x,\Delta p),\end{aligned}$$

with $\Delta p(a,\alpha) = c_0 + \sum_{j=1}^{l} a_j\beta_j c_j + \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{ij} c_{ij}^{\prime(z)}$, and the remainder $R(\tilde{x},\tilde{p},\Delta x,\Delta p)$ the sum of terms of order 3 and higher, and of order 2 applied to $z'_{11},\ldots,z'_{ll}$. The condition (5.51) is eliminated by setting $\alpha_{ij} \approx -\frac{1}{2}a_i a_j$ $(\forall i,j = 1,\ldots,l)$. Condition (5.76) is not introduced. □

Lemma 5.17. The values $a_1,\ldots,a_l$, that appear in the update vectors (5.90) and (5.92) of the alternative SBNE method, typically satisfy the bound

$$\forall j = 1,\ldots,l : |a_j| \lesssim \sqrt{|\beta_s\lambda_s|}, \tag{5.93}$$

The index s is chosen such that $|a_s\beta_s\lambda_s|$ is maximal.

Proof. Consider the same Taylor expansion as in the proof of lemma 5.16. The term

$$\frac{1}{6}\sum_{i=1}^{l}\sum_{j=1}^{l}\sum_{k=1}^{l} F_{XXX}(\tilde{x},\tilde{p})\Phi_i\Phi_j\Phi_k,$$

which is included in the remainder $R(\tilde{x}, \tilde{p}, \Delta x, \Delta p)$, induces the condition

$$\max_{k,i,j=1,\ldots,l} |a_k a_i a_j| \lesssim \max_{j=1,\ldots,l} |a_j \beta_j \lambda_j|. \tag{5.94}$$

This condition implies the bound (5.93). □

Compared to the bound on $a_1, \ldots, a_l$ discussed in lemma 5.14 (see section 5.6.2), the one of lemma 5.17 is less predictable: it is possible for $\beta_1, \ldots, \beta_l$ to become very small, this happens when $|c_j| \gg 1$ (for the corresponding index $j \in \{1, \ldots, l\}$). The values of $c_1, \ldots, c_l$ depend on the considered linear function G and cannot be damped.

Though the alternative approach does not yield bound (5.76) (discussed in remark 5.15) on Δp, numerical experiments (e.g. example 5.12) suggest that the required values for Δp in the Newton updates are typically sufficiently small: condition (5.76) does not hamper the convergence.

Since the original SBNE method is preferable when $|c_j| \gg 1$ (for a certain $j \in \{1, \ldots, l\}$), we will not consider the alternative approach for our numerical experiments.

5.7 Reduction of the extra terms in presence of a dominant update direction

The amount of additional linear systems introduced in section 5.6 will be reduced in a similar way as was done for the method without block elimination in section 4.7.

5.7.1 Creation of a new base for the approximate kernel

We consider two consecutive guesses $(\tilde{x}^{(m)}, \tilde{p}^{(m)})$ and $(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$, such that

$$\begin{aligned}
\tilde{x}^{(m+1)} &= \tilde{x}^{(m)} + \Delta x^{(m)} = \tilde{x}^{(m)} + \Delta x_\perp^{(m)} + \Delta x_\parallel^{(m)} + \Delta x_z^{(m)} \\
&= \tilde{x}^{(m)} + y_\perp^{(1)(m)} - c_0^{(m)} y_\perp^{(2)(m)} + \sum_{j=1}^{l} a_j^{(m)} \left(\phi_j^{(m)} - c_j^{(m)} y_\perp^{(2)(m)} \right) \\
&\quad + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_{ij}^{(m)} \left(z_{ij}^{(m)} - c_{ij}^{(z)(m)} y_\perp^{(2)(m)} \right),
\end{aligned} \tag{5.95}$$

$$\begin{aligned}
\tilde{p}^{(m+1)} &= \tilde{p}^{(m)} + \Delta p^{(m)} = \tilde{p}^{(m)} + \Delta p_\perp^{(m)} + \Delta p_\parallel^{(m)} + \Delta p_z^{(m)} \\
&= \tilde{p}^{(m)} + c_0^{(m)} + \sum_{j=1}^{l} a_j^{(m)} c_j^{(m)} + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_{ij}^{(m)} c_{ij}^{(z)(m)},
\end{aligned} \tag{5.96}$$

with the vectors $y_\perp^{(1)(m)}$, $y_\perp^{(2)(m)}$, $z_{11}^{(m)}, \ldots, z_{ll}^{(m)}$ and values $c_0^{(m)}$, $c_1^{(m)}, \ldots, c_l^{(m)}$, $c_{11}^{(z)(m)}, \ldots, c_{ll}^{(z)(m)}$, $a_1^{(m)}, \ldots, a_l^{(m)}$ and $\alpha_{11}^{(m)}, \ldots, \alpha_{ll}^{(m)}$ calculated as described in section 5.6. The vectors $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$ represent the approximate null vectors of

$F_x(\tilde{x}^{(m)}, \tilde{p}^{(m)})$, with respective eigenvalues $\lambda_1^{(m)}, \ldots, \lambda_l^{(m)}$, such that $\|\phi_1^{(m)}\| = \cdots = \|\phi_l^{(m)}\| = 1$.

Denoting $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ the approximate null vectors of the Jacobian $F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$ (such that $\|\phi_1^{(m+1)}\| = \cdots = \|\phi_l^{(m+1)}\| = 1$ and $\phi_j^{(m+1)} \approx \phi_j^{(m)}$ for each $j = 1, \ldots, l$) and $\lambda_1^{(m+1)}, \ldots, \lambda_l^{(m+1)}$ the respective eigenvalues, a new base $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ for the space spanned by $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ is created, entirely similar to section 4.7.1. The vector $\breve{\phi}_1^{(m+1)}$ is defined by

$$\begin{aligned} \mathring{\phi} &= \sum_{j=1}^{l} \langle \phi_j^{(m+1)}, \Delta x_{\|}^{(m)} \rangle \phi_j^{(m+1)}, \\ \breve{\phi}_1^{(m+1)} &= \|\mathring{\phi}\|^{-1} \mathring{\phi}, \end{aligned} \tag{5.97}$$

with $\Delta x_{\|}^{(m)} = \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)}$, vectors $\breve{\phi}_2^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ are chosen as linear combinations of $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ such that $\langle \breve{\phi}_i^{(m+1)}, \breve{\phi}_j^{(m+1)} \rangle = \delta_{ij}$ ($\forall i, j = 1, \ldots, l$). The $l \times l$ matrix D is defined such that

$$\forall i, j = 1, \ldots, l : D_{ij} = \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m+1)} \rangle,$$

it describes the base transformation. Note that D is a unitary matrix, implying $D^T = D^{-1}$.

We split the following vectors into a linear combination of $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ and a part orthogonal to these directions:

$$F(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) = F_\perp + F_\| = F_\perp + \sum_{k=1}^{l} b_k^{(1)} \breve{\phi}_k^{(m+1)}, \tag{5.98}$$

$$F_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) = F_{p\perp} + F_{p\|} = F_{p\perp} + \sum_{k=1}^{l} b_k^{(2)} \breve{\phi}_k^{(m+1)}, \tag{5.99}$$

$\forall i, j = 1, \ldots, l :$

$$F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)} = F_\perp^{(ij)} + F_\|^{(ij)} = F_\perp^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)} \breve{\phi}_k^{(m+1)}. \tag{5.100}$$

Define the vectors $y_\perp^{(1)(m)}$ and $y_\perp^{(2)(m)}$ as the solutions of respectively (5.20) and (5.21), and $z_{11}^{(m+1)}, \ldots, z_{ll}^{(m+1)}$ as the solutions of the linear systems ($\forall i, j = 1, \ldots, l$)

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) z_{ij}^{(m+1)} \\ &\qquad = \quad \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_i^{(m+1)} \breve{\phi}_j^{(m+1)}, \end{aligned} \tag{5.101}$$

with $\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$ defined by

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) : \mathbb{C}^n \to \mathbb{C}^n : y \to y - K \langle K, K \rangle^{-1} \langle K, y \rangle \\ &\text{with } K = \begin{pmatrix} \breve{\phi}_1^{(m+1)} & \breve{\phi}_2^{(m+1)} & \ldots & \breve{\phi}_l^{(m+1)} \end{pmatrix}. \end{aligned}$$

With $c_0^{(m+1)}$ defined similar to (5.40), and $c_1^{(m+1)}, \ldots, c_l^{(m+1)}$, $c_{11}^{(z)(m+1)}, \ldots, c_{ll}^{(z)(m+1)}$ by $(\forall j = 1, \ldots, l)$

$$c_j^{(m+1)} = \frac{-G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\breve{\phi}_j^{(m+1)}}{G_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) - G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})y_\perp^{(2)(m+1)}}, \tag{5.102}$$

$$c_{ij}^{(z)(m+1)} = \frac{-G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})z_{ij}^{(m+1)}}{G_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) - G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})y_\perp^{(2)(m+1)}}, \tag{5.103}$$

the update vectors for a normal step of the split block Newton method with extra terms (SBNE) would be given by

$$\begin{aligned}
\Delta x^{(m+1)} &= \Delta x_\perp^{(m+1)} + \Delta x_\parallel^{(m+1)} + \Delta x_z^{(m+1)} \\
&= y_\perp^{(1)(m+1)} - c_0^{(m+1)} y_\perp^{(2)(m+1)} + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_\perp^{(2)(m+1)} \right) \\
&\quad + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} \left(z_{ij}^{(m+1)} - c_{ij}^{(z)(m+1)} y_\perp^{(2)(m+1)} \right), \\
\Delta p^{(m+1)} &= \Delta p_\perp^{(m+1)} + \Delta p_\parallel^{(m+1)} + \Delta p_z^{(m+1)} \\
&= c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} + \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_{ij} c_{ij}^{(z)(m+1)},
\end{aligned}$$

where $a_1, \ldots, a_l$ and $\alpha_{11}, \ldots, \alpha_{ll}$ would be calculated such that the new residual norm becomes minimal. Note that the vectors $z_{11}^{(m+1)}, \ldots, z_{ll}^{(m+1)}$ will not appear in the eventual update vector (see section 5.7.3).

5.7.2 Elimination of less important terms in presence of a dominant update vector

The same assumption as in section 4.7.2 is made (see assumption 4.15).

Assumption 5.18. The part $\Delta x_\parallel^{(m+1)}$ of the update vector $\Delta x^{(m+1)}$ in the null vector directions is dominated by $\breve{\phi}_1^{(m+1)}$, defined in (5.97). This implies

$$\forall i = 1, \ldots, l : i \neq 1 \Rightarrow |a_i| \ll |a_1| \tag{5.104}$$

in the update vectors.

Assumption 5.18 allows us to ignore terms $z_{ij}^{(m+1)}$ in the part $\Delta x_z^{(m+1)}$ for

which both $i \neq 1$ and $j \neq 1$. A first reduction yields the update vectors

$$\begin{aligned}\Delta x^{(m+1)} &= \Delta x_{\perp}^{(m+1)} + \Delta x_{\parallel}^{(m+1)} + \Delta x_z'^{(m+1)} && (5.105)\\ &= y_{\perp}^{(1)(m+1)} - c_0^{(m+1)} y_{\perp}^{(2)(m+1)} + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_{\perp}^{(2)(m+1)} \right) \\ &\quad + \sum_{j=1}^{l} \alpha_j \left(z_j^{(m+1)} - c_j^{(z)(m+1)} y_{\perp}^{(2)(m+1)} \right), && (5.106)\end{aligned}$$

$$\begin{aligned}\Delta p^{(m+1)} &= \Delta p_{\perp}^{(m+1)} + \Delta p_{\parallel}^{(m+1)} + \Delta p_z'^{(m+1)} && (5.107)\\ &= c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} + \sum_{j=1}^{l} \alpha_j c_j^{(z)(m+1)}, && (5.108)\end{aligned}$$

with $z_{1j}^{(m+1)}$ denoted as $z_j^{(m+1)}$ and $c_{1j}^{(z)(m+1)}$ as $c_j^{(z)(m+1)}$ $(\forall j = 1, \ldots, l)$.

Lemma 5.19. Under assumption 5.18, the bound discussed in lemma 5.14 is eliminated by the additional parts $\Delta x_z'^{(m+1)}$ and $\Delta p_z'^{(m+1)}$ defined in (5.106) and (5.108).

Proof. The proof is entirely similar to the one of lemma 4.16. □

5.7.3 Replacement of the additional update parts based on the previous Newton iteration

The remaining vectors $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ of the part $\Delta x_z'^{(m+1)}$ will be replaced by considering a finite difference approximation. We make two more assumptions.

Assumption 5.20. The part $\begin{pmatrix} \Delta x_{\perp}^{(m)} & \Delta p_{\perp}^{(m)} \end{pmatrix}^T$ of the previous update vector (see (5.96)) is negligible compared to the part $\begin{pmatrix} \Delta x_{\parallel}^{(m)} & \Delta p_{\parallel}^{(m)} \end{pmatrix}^T$. This implies

$$\sqrt{\|y_{\perp}^{(1)(m)} - c_0^{(m)} y_{\perp}^{(2)(m)}\|^2 + c_0^{(m)2}} \ll \max_{j=1,\ldots,l} |a_j^{(m)}|. \tag{5.109}$$

Assumption 5.21. The update $\Delta p^{(m)}$ from $\tilde{p}^{(m)}$ to $\tilde{p}^{(m+1)}$ (see (5.96)) is negligible compared to the part $\Delta x_{\parallel}^{(m)}$ of the update vector $\Delta x^{(m)}$. This implies

$$|\Delta p^{(m)}| \ll \max_{j=1,\ldots,l} |a_j^{(m)}|. \tag{5.110}$$

Both assumptions typically hold in later Newton iterations: the residual in these steps is dominated by approximate null vectors. The term $y_{\perp}^{(1)(m)}$ in the update is very small compared to $\sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)}$. Together with the definition of $c_0^{(m)}$, assumption 5.20 is implied by this. Assumption 5.21 is typically valid due to condition (5.76) (see remark 5.15).

Lemma 5.22. Under assumptions 5.18, 5.20 and 5.21, the terms $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ can approximately be solved from the linear systems ($\forall j = 1, \ldots, l$)

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) z_j^{(m+1)} \\ &\qquad \approx \quad -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m)}, \tilde{p}^{(m)}) \rho_j \end{aligned} \tag{5.111}$$

with $\gamma = \| \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)} \|$ and $\rho_1, \ldots, \rho_l$ defined by

$$\forall j = 1, \ldots, l : \rho_j = \breve{\phi}_j^{(m+1)} - \sum_{i=1}^{l} \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m)} \rangle \phi_i^{(m)}. \tag{5.112}$$

Proof. The proof is similar to that of lemma 4.18 in section 4.7.3. Choose $j \in \{1, \ldots, l\}$. A finite difference approximation gives the approximation

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)} \\ &\quad \approx \quad -\varepsilon^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m+1)} - \varepsilon \breve{\phi}_1^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_j^{(m+1)}, \end{aligned}$$

where $\varepsilon \in \mathbb{R}_0^+$ is yet to be chosen.

Under assumptions 5.20 and 5.21, the full update vector $\begin{pmatrix} \Delta x^{(m)} & \Delta p^{(m)} \end{pmatrix}^T$ is dominated by the part $\sum_{i=1}^{l} a_i^{(m)} \begin{pmatrix} \phi_i^{(m)} & 0 \end{pmatrix}^T$. Note that, since $\alpha_{ik}^{(m)} \approx -\frac{1}{2} a_i^{(m)} a_k^{(m)}$ ($\forall i, k = 1, \ldots, l$), the vectors in the additional part $\Delta x_z'^{(m)}$ of the update vector are dominated by $\sum_{i=1}^{l} a_i^{(m)} \phi_i^{(m)}$. By the definition of $\breve{\phi}_1^{(m+1)}$ (see (5.97)), we have $\sum_{i=1}^{l} a_i^{(m)} \phi_i^{(m)} \approx \gamma \breve{\phi}_1^{(m+1)}$ with $\gamma = \| \sum_{i=1}^{l} a_i^{(m)} \phi_i^{(m)} \|$. Choosing ε as γ, and using

$$\begin{pmatrix} \tilde{x}^{(m+1)} \\ \tilde{p}^{(m+1)} \end{pmatrix} - \begin{pmatrix} \gamma \breve{\phi}_1^{(m+1)} \\ 0 \end{pmatrix} \quad \approx \quad \begin{pmatrix} \tilde{x}^{(m+1)} \\ \tilde{p}^{(m+1)} \end{pmatrix} - \begin{pmatrix} \Delta x^{(m)} \\ \Delta p^{(m)} \end{pmatrix} \quad = \quad \begin{pmatrix} \tilde{x}^{(m)} \\ \tilde{p}^{(m)} \end{pmatrix},$$

the approximation

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)} \\ &\qquad \approx \quad -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m)}, \tilde{p}^{(m)}) \breve{\phi}_j^{(m+1)} \end{aligned}$$

is derived. Splitting the vector $\breve{\phi}_j^{(m+1)}$ as

$$\breve{\phi}_j^{(m+1)} = \sum_{i=1}^{l} \zeta_{ij} \phi_i^{(m)} + \rho_j,$$

with $\zeta_{ij} = \langle \breve{\phi}_j^{(m+1)}, \phi_i^{(m)} \rangle$ ($\forall i = 1, \ldots, l$) and ρ_j defined by (5.112) (note that $\langle \rho_j, \phi_i^{(m)} \rangle = 0$ for each $i = 1, \ldots, l$), similar analysis as in the proof of lemma 4.18 eventually shows

$$\begin{aligned} &\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) \breve{\phi}_1^{(m+1)} \breve{\phi}_j^{(m+1)} \\ &\qquad \approx \quad -\gamma^{-1} \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) F_x(\tilde{x}^{(m)}, \tilde{p}^{(m)}) \rho_j. \end{aligned}$$

The statement now follows by substituting this approximation in the linear system for $z_j^{(m+1)}$ (see (5.67)). □

Lemma 5.22 presents alternative linear systems to solve for the vectors $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$. Instead of solving these systems, we will replace $z_1^{(m+1)}, \ldots, z_l^{(m+1)}$ by $-\gamma^{-1}\rho'_1, \ldots, -\gamma^{-1}\rho'_l$ with $\rho'_j = \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\rho_j$ $(\forall j = 1, \ldots, l)$.

The part $\Delta x_z'^{(m+1)}$ of (5.106) is now replaced by $\Delta x_\rho^{(m+1)}$, consisting of the terms $\rho'_1, \rho'_2, \ldots, \rho'_l$. The part $\Delta p_z'^{(m+1)}$ of (5.108) is replaced as well. The eventual update vectors are given by

$$\begin{aligned}
\Delta x^{(m+1)} &= \Delta x_\perp^{(m+1)} + \Delta x_\parallel^{(m+1)} + \Delta x_\rho^{(m+1)} && (5.113)\\
&= y_\perp^{(1)(m+1)} - c_0^{(m+1)} y_\perp^{(2)(m+1)} + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_\perp^{(2)(m+1)} \right) \\
&\quad - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \left(\rho'_j - c_j^{(\rho)} y_\perp^{(2)(m+1)} \right), && (5.114)\\
\Delta p^{(m+1)} &= \Delta p_\perp^{(m+1)} + \Delta p_\parallel^{(m+1)} + \Delta p_\rho^{(m+1)} && (5.115)\\
&= c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j c_j^{(\rho)}, && (5.116)
\end{aligned}$$

with $c_1^{(\rho)}, \ldots, c_l^{(\rho)}$ defined by

$$\forall j = 1, \ldots, l : c_j^{(\rho)} = \frac{-G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\rho'_j}{G_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) - G_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) y_\perp^{(2)(m+1)}}. \tag{5.117}$$

5.7.4 Minimization of the residual norm

The values for $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ in the update vectors (5.114) and (5.116) are calculated by applying the nonlinear conjugate gradients method to find the arguments that minimize the function

$$\begin{aligned}
g :& \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\
&(a, \alpha) \to \Bigg\| F \Bigg(\tilde{x}^{(m+1)} + y_\perp^{(1)(m+1)} - c_0^{(m+1)} y_\perp^{(2)(m+1)} \\
&\qquad + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_\perp^{(2)(m+1)} \right) \\
&\qquad - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \left(\rho'_j - c_j^{(\rho)} y_\perp^{(2)(m+1)} \right), \\
&\qquad\quad \tilde{p}^{(m+1)} + c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j c_j^{(\rho)} \Bigg) \Bigg\|^2 .
\end{aligned} \tag{5.118}$$

The required derivatives of g are given for the preconditioned case in section 5.7.6. When an initial guess $a^{(0)}$ is derived, the one for $\alpha^{(0)}$ is calculated by $\alpha_j^{(0)} = -\frac{1}{2}a_1^{(0)}a_j^{(0)}$ $(\forall j = 1, \ldots, l)$. The guess for $a^{(0)}$ itself is calculated by first minimizing

$$\begin{aligned} f :& \mathbb{R}^l \to \mathbb{R} : \\ & a' \to \sum_{j=1}^{l} \left((b_j^{(1)} + \Delta p(a') b_j^{(2)} + \sum_{k=1}^{l} a'_k C_{jk} + \frac{1}{2} c_j^{(11)} a_1'^2 + \sum_{k=1}^{l} c_j^{(1k)} a'_1 a'_k \right)^2 \\ & \quad + \left(\Delta p(a') \| y_\perp^{(1)(m+1)} \| \right)^2 + \sum_{j=1}^{l} \left(a'_j \| y_\perp^{(1)(m+1)} \| \right)^2 + \Delta p(a')^4 \\ & \quad + \sum_{j=1}^{l} \left(\Delta p(a') a'_j \right)^2 + \sum_{i=2}^{l} \sum_{j=2}^{l} \left(a'_i a'_j \right)^2 + \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a'_k a'_i a'_j \right)^2, \\ \text{with } & \Delta p(a') = c_0^{(m+1)} + \sum_{j=1}^{l} a'_j c_j^{(m+1)} + \frac{1}{2} \gamma^{-1} \sum_{j=1}^{l} a'_1 a'_j c_j^{(\rho)}, \end{aligned} \tag{5.119}$$

with $C_{jk} = \sum_{i=1}^{l} D_{ik} \lambda_i^{(m+1)} D_{ij}$ $(\forall j, k = 1, \ldots, l)$.

The minimization of f is performed by nonlinear conjugate gradients as well, and requires an initial guess $a'^{(0)}$. Denote $b_j'^{(1)} = \langle F(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}), \phi_j^{(m+1)} \rangle$ and $b_j'^{(2)} = \langle F_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}), \phi_j^{(m+1)} \rangle$ $(\forall j = 1, \ldots, l)$. First the values

$$\begin{aligned} \forall j = 1, \ldots, l : a_j'^{(r)} &= -\frac{|b_j'^{(1)}|}{\lambda_j^{(m+1)}}, \\ \forall j = 1, \ldots, l : a_j'^{(p)} &= -\frac{|b_j'^{(2)}|}{\lambda_j^{(m+1)}}, \\ \forall j = 1, \ldots, l : a_j'^{(t)} &= \sum_{i=1}^{l} D_{ij} a_i'^{(p)} \end{aligned}$$

are calculated. From this, the values

$$\begin{aligned} a_s'^{(e)} &= \operatorname{sgn}(a_s'^{(r)}) \min \left(|a_s'^{(r)}|, \sqrt{|\lambda_s^{(m+1)}|}, \frac{|b_s'^{(1)}|}{\max \left(|c^{(t)}|, \| y_\perp^{(1)(m+1)} \| \right)} \right), \\ \forall j \neq s : a_j'^{(e)} &= \operatorname{sgn}(a_j'^{(r)}) \min \left(|a_j'^{(r)}|, |a_s'^{(e)}| \right), \end{aligned}$$

are derived, with $c^{(t)}$ the lowest magnitude solution of the quadratic equation

$$c^{(t)} = c_0^{(m+1)} + \sum_{j=1}^{l} c_j^{(m+1)} c^{(t)} a_j'^{(t)} + \frac{1}{2} \gamma^{-1} \sum_{j=1}^{l} c_j^{(\rho)} c^{(t)2} a_1'^{(t)} a_j'^{(t)}.$$

A base transformation is executed to gain the values $a_1'^{(b)}, \ldots, a_l'^{(b)}$:

$$\forall j = 1, \ldots, l : a_j'^{(b)} = \sum_{i=1}^{l} D_{ij} a_i'^{(e)}.$$

Defining $\Delta p(\chi)$ as the lowest magnitude solution of the quadratic equation

$$\begin{aligned} \Delta p(\chi) &= c_0^{(m+1)} + \sum_{j=1}^{l} c_j^{(m+1)} \left(\chi a_j'^{(b)} + \Delta p(\chi) a_j'^{(t)} \right) \\ &\quad + \frac{1}{2} \gamma^{-1} \sum_{j=1}^{l} c_j^{(\rho)} \left(\chi a_1'^{(b)} + \Delta p(\chi) a_1'^{(t)} \right) \left(\chi a_j'^{(b)} + \Delta p(\chi) a_j'^{(t)} \right), \end{aligned}$$

the value $c^{(q)}$ is calculated as the maximum of the first derivative of $\Delta p(\chi)$ over an interval $[0; 1]$. The value χ is then calculated by

$$\chi = \min \left(1, \frac{|\lambda_s^{(m+1)}|}{|c^{(q)}|}, \frac{|\lambda_s^{(m+1)}||a_s'^{(e)}|}{c^{(q)2}}, \frac{|b_j'^{(1)}|}{|c^{(q)}| \max \left(|c^{(t)}|, \|y_\perp^{(1)(m+1)}\| \right)} \right).$$

Defining $\tilde{\kappa}$ by

$$\tilde{\kappa} = \min \left(1, \frac{\max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} a_j'^{(b)} C_{kj} \right|}{\chi \max_{j=2,\ldots,l} |a_i'^{(b)} a_j'^{(b)}|} \right), \tag{5.120}$$

with $C_{kj} = \sum_{i=1}^{l} D_{ij} \lambda_i^{(m+1)} D_{ik}$ $(\forall j, k = 1, \ldots, l)$, the values $a_1'^{(q)}, \ldots, a_l'^{(q)}$ are defined by

$$\forall j = 1, \ldots, l : a_j'^{(q)} = \tilde{\kappa} a_j'^{(b)}.$$

The initial guess $a'^{(0)}$ is eventually constructed by

$$\forall j = 1, \ldots, l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(t)}, \tag{5.121}$$

with $\Delta p^{(0)}$ the lowest magnitude solution of the quadratic equation

$$\begin{aligned} \Delta p^{(0)} &= c_0^{(m+1)} + \sum_{j=1}^{l} c_j^{(m+1)} \left(\chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(t)} \right) \\ &\quad + \frac{1}{2} \gamma^{-1} \sum_{j=1}^{l} c_j^{(\rho)} \left(\chi a_1'^{(q)} + \Delta p^{(0)} a_1'^{(t)} \right) \left(\chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(t)} \right). \end{aligned}$$

5.7.5 The method in practice

We will call the method that uses update vectors (5.114) and (5.116) (see section 5.7.3) the *split block Newton method with reduced terms* (SBNR), pseudo-code is given by algorithm 5.6 on page 162 in appendix 5.10. In practice the first iteration that is performed uses update vectors of the form (5.43) and (5.45),

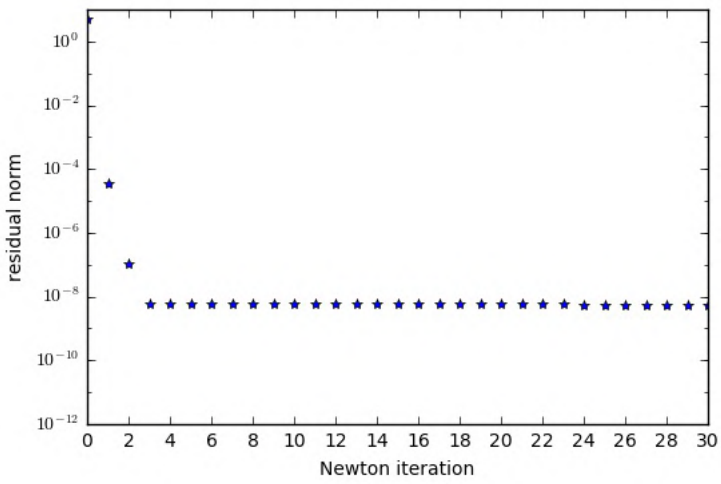


Figure 5.9: Residual plot of example 5.23. The SBNR method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-8} after 3 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps.

afterwards ones of the form (5.114) and (5.116) are used. These steps start by calculating approximate null vectors of the partial Jacobian $F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$ (using algorithm 3.3, see page 43), and performs a base transformation on these by (5.97), yielding $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$. Vectors $y_\perp^{(1)(m+1)}$ and $y_\perp^{(2)(m+1)}$ are calculated by solving linear systems (5.20) and (5.21) as before (using algorithm 3.4, see page 44), vectors $\rho_1', \ldots, \rho_l'$ by orthogonalizing $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ to $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$, the approximate null vectors used in the previous Newton iteration, and applying $\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$. The update vectors are constructed by application of (5.114) and (5.116), where $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ are calculated by minimizing (5.125), the preconditioned version of (5.118), with the nonlinear conjugate gradients method (algorithm 3.5, page 45). After updating the guesses, $\Delta x_\parallel^{(m)}$ and γ are calculated and the approximate null vectors are stored for use in the next Newton iteration.

The algorithm is applied in example 5.23. Similar conclusions as for the method without block elimination are valid: convergence behaviour depends on whether assumption 5.18 is valid (see lemma 5.19). No additional linear systems need to be solved for the method though.

Example 5.23. Consider the same set-up as in example 5.2. Figure 5.9 shows the residual plot after application of the SBNR method. After 3 Newton iterations, the residual norm remains around a value of 10^{-8}. ◇

5.7.6 The preconditioned case

For the preconditioned case $\phi_1^{(m)}, \ldots, \phi_l^{(m)}$ and $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ represent the approximate null vectors of respectively $P(\tilde{x}^{(m)}, \tilde{p}^{(m)})F_x(\tilde{x}^{(m)}, \tilde{p}^{(m)})$ and $P(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$. The vectors $\tau_1^{(m)}, \ldots, \tau_l^{(m)}$ are defined such that $P(\tilde{x}^{(m)}, \tilde{p}^{(m)})\tau_j^{(m)} = \phi_j^{(m)}$ $(\forall j = 1, \ldots, l)$, $\tau_1^{(m+1)}, \ldots, \tau_l^{(m+1)}$ such that $P(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\tau_j^{(m+1)} = \phi_j^{(m+1)}$ $(\forall j = 1, \ldots, l)$. The respective eigenvalues of $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ are denoted by $\lambda_1^{(m+1)}, \ldots, \lambda_l^{(m+1)}$. To create a new base for the space spanned by $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$, we define

$$\mathring{\phi} = \sum_{j=1}^{l} \langle \tau_j^{(m+1)}, \Delta x_{\parallel}^{(m)} \rangle \phi_j^{(m+1)}, \quad \mathring{\tau} = \sum_{j=1}^{l} \langle \tau_j^{(m+1)}, \Delta x_{\parallel}^{(m)} \rangle \tau_j^{(m+1)},$$

$$\breve{\phi}_1^{(m+1)} = \left(\sqrt{|\langle \mathring{\phi}, \mathring{\tau} \rangle|} \right)^{-1} \mathring{\phi}, \qquad \breve{\tau}_1^{(m+1)} = \left(\sqrt{|\langle \mathring{\phi}, \mathring{\tau} \rangle|} \right)^{-1} \mathring{\tau},$$

with $\Delta x_{\parallel}^{(m)} = \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)}$. Vectors $\breve{\tau}_2^{(m+1)}, \ldots, \breve{\tau}_l^{(m+1)}$ and $\breve{\phi}_2^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ are again chosen as linear combinations such that $\langle \breve{\tau}_i^{(m+1)}, \breve{\phi}_j^{(m+1)} \rangle = \delta_{ij}$ $(\forall i, j = 1, \ldots, l)$ and $P(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\breve{\tau}_i^{(m+1)} = \breve{\phi}_i^{(m+1)}$ $(\forall i = 1, \ldots, l)$. The elements of the matrix D that describes the base transformation are this time given by

$$\forall i, j = 1, \ldots, l : D_{ij} = \langle \breve{\phi}_j^{(m+1)}, \tau_i^{(m+1)} \rangle.$$

The following vectors are split:

$$F(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) = F_{\perp} + F_{\parallel} = F_{\perp} + \sum_{k=1}^{l} b_k^{(1)} \breve{\tau}_k^{(m+1)}, \tag{5.122}$$

$$F_p(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}) = F_{p\perp} + F_{p\parallel} = F_{p\perp} + \sum_{k=1}^{l} b_k^{(2)} \breve{\tau}_k^{(m+1)}, \tag{5.123}$$

$\forall i, j = 1, \ldots, l :$

$$F_{xx}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\breve{\phi}_i^{(m+1)}\breve{\phi}_j^{(m+1)} = F_{\perp}^{(ij)} + F_{\parallel}^{(ij)} = F_{\perp}^{(ij)} + \sum_{k=1}^{l} c_k^{(ij)} \breve{\tau}_k^{(m+1)}, \tag{5.124}$$

and the vectors $y_{\perp}^{(1)(m+1)}$ and $y_{\perp}^{(2)(m+1)}$ are solved from the linear systems (5.24) and (5.25). The approximate null vectors $\breve{\phi}_1^{(m+1)}, \ldots, \breve{\phi}_l^{(m+1)}$ are split as well, yielding

$$\forall j = 1, \ldots, l : \breve{\phi}_j^{(m+1)} = \sum_{i=1}^{l} \zeta_{ij} \phi_i^{(m)} + \rho_j,$$

with$\langle \rho_j, \tau_i^{(m)} \rangle = 0$ $(\forall i, j = 1, \ldots, l)$. Terms of the form $-\gamma^{-1}\rho_1', \ldots, -\gamma^{-1}\rho_l'$ are used in the additional part $\Delta x_{\rho}^{(m+1)}$ of the update vector, with $\gamma = \| \sum_{j=1}^{l} a_j^{(m)} \phi_j^{(m)} \|_{\tilde{P}^{-1}}$, $\rho_j' = \mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})\rho_j$ $(\forall j = 1, \ldots, l)$ and the deflation operator $\mathcal{Q}(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$ defined by (5.59). Note that we use $\tilde{P}$ to

denote the preconditioner $\tilde{P} = P(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$. With $c_1^{(m+1)}, \ldots, c_l^{(m+1)}$ and $c_1^{(\rho)}, \ldots, c_l^{(\rho)}$ defined by (5.102) and (5.117), the update vectors are again constructed by (5.114) and (5.116).

The values $a_1, \ldots, a_l$ and $\alpha_1, \ldots, \alpha_l$ that appear in the formulas are calculated by application of the nonlinear conjugate gradients method to the function

$$\begin{aligned} g :&\mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &(a, \alpha) \to \left\| F\Bigg(\tilde{x}^{(m+1)} + y_\perp^{(1)(m+1)} - c_0^{(m+1)} y_\perp^{(2)(m+1)} \right. \\ &\qquad + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_\perp^{(2)(m+1)} \right) \\ &\qquad - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \left(\rho_j' - c_j^{(\rho)} y_\perp^{(2)(m+1)} \right), \\ &\qquad\qquad \left. \tilde{p}^{(m+1)} + c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j c_j^{(\rho)} \Bigg) \right\|_{\tilde{P}}^2 . \end{aligned} \tag{5.125}$$

The first partial derivatives of g are given by $(\forall i = 1, \ldots, l)$

$$\begin{aligned} &\frac{\partial g}{\partial a_i} :\mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to 2 \langle F(x(a,\alpha), p(a,\alpha)), F_X(x(a,\alpha), p(a,\alpha)) \Phi_i \rangle_{\tilde{P}}, \\ &\frac{\partial g}{\partial \alpha_i} :\mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\ &\qquad (a, \alpha) \to -2\gamma^{-1} \langle F(x(a,\alpha), p(a,\alpha)), F_X(x(a,\alpha), p(a,\alpha)) Z_i \rangle_{\tilde{P}}, \\ &\text{with } \forall i = 1, \ldots, l : \Phi_i = \begin{pmatrix} \breve{\phi}_i^{(m+1)} - c_i^{(m+1)} y_\perp^{(2)(m+1)} \\ c_i^{(m+1)} \end{pmatrix}, \\ &\forall i, j = 1, \ldots, l : Z_j = \begin{pmatrix} \rho_j' - c_j^{(\rho)} y_\perp^{(2)(m+1)} \\ c_j^{(\rho)} \end{pmatrix}, \\ &x(a,\alpha) = \tilde{x}^{(m+1)} + y_\perp^{(1)(m+1)} - c_0^{(m+1)} y_\perp^{(2)(m+1)} \\ &\qquad + \sum_{j=1}^{l} a_j \left(\breve{\phi}_j^{(m+1)} - c_j^{(m+1)} y_\perp^{(2)(m+1)} \right) \\ &\qquad - \gamma^{-1} \sum_{j=1}^{l} \alpha_j \left(\rho_j' - c_j^{(\rho)} y_\perp^{(2)(m+1)} \right), \\ &\text{and } p(a,\alpha) = \tilde{p}^{(m+1)} + c_0^{(m+1)} + \sum_{j=1}^{l} a_j c_j^{(m+1)} - \gamma^{-1} \sum_{j=1}^{l} \alpha_j c_j^{(\rho)}. \end{aligned} \tag{5.126}$$

The second ones by $(\forall i, k = 1, \ldots, l)$

$$\begin{aligned}
&\frac{\partial^2 g}{\partial a_i \partial a_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\
&\quad (a, \alpha) \to 2\langle F_X(x(a,\alpha), p(a,\alpha))\Phi_i, F_X(x(a,\alpha), p(a,\alpha))\Phi_k \rangle_{\tilde{P}} \\
&\qquad + 2\langle F(x(a,\alpha), p(a,\alpha)), F_{XX}(x(a,\alpha), p(a,\alpha))\Phi_i\Phi_k \rangle_{\tilde{P}}, \\
&\frac{\partial^2 g}{\partial a_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\
&\quad (a, \alpha) \to -2\gamma^{-1}\langle F_X(x(a,\alpha), p(a,\alpha))\Phi_i, F_X(x(a,\alpha), p(a,\alpha))Z_k \rangle_{\tilde{P}} \\
&\qquad - 2\gamma^{-1}\langle F(x(a,\alpha), p(a,\alpha)), F_{XX}(x(a,\alpha), p(a,\alpha))\Phi_i Z_k \rangle_{\tilde{P}}, \\
&\frac{\partial^2 g}{\partial \alpha_i \partial \alpha_k} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : \\
&\quad (a, \alpha) \to 2\gamma^{-2}\langle F_X(x(a,\alpha), p(a,\alpha))Z_i, F_X(x(a,\alpha), p(a,\alpha))Z_k \rangle_{\tilde{P}} \\
&\qquad + 2\gamma^{-2}\langle F(x(a,\alpha), p(a,\alpha)), F_{XX}(x(a,\alpha), p(a,\alpha))Z_i Z_k \rangle_{\tilde{P}}.
\end{aligned} \tag{5.127}$$

Given an initial guess $a^{(0)}$, $\alpha^{(0)}$ is still calculated by $\alpha_j^{(0)} = -\frac{1}{2}a_1^{(0)}a_j^{(0)}$ $(\forall j = 1, \ldots, l)$. The guess for a is derived by first minimizing

$$\begin{aligned}
&f : \mathbb{R}^l \to \mathbb{R} : \\
&\quad a' \to \sum_{j=1}^{l} \left((b_j^{(1)} + \Delta p(a') b_j^{(2)} + \sum_{k=1}^{l} a'_k C_{jk} + \frac{1}{2} c_j^{(11)} a_1'^2 + \sum_{k=1}^{l} c_j^{(1k)} a'_1 a'_k \right)^2 \\
&\qquad + \left(\Delta p(a') \| y_\perp^{(1)(m+1)} \|_{\tilde{P}^{-1}} \right)^2 + \sum_{j=1}^{l} \left(a'_j \| y_\perp^{(1)(m+1)} \|_{\tilde{P}^{-1}} \right)^2 + \Delta p(a')^4 \\
&\qquad + \sum_{j=1}^{l} \left(\Delta p(a') a'_j \right)^2 + \sum_{i=2}^{l} \sum_{j=2}^{l} \left(a'_i a'_j \right)^2 + \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a'_k a'_i a'_j \right)^2,
\end{aligned} \tag{5.128}$$

$$\text{with } \Delta p(a') = c_0^{(m+1)} + \sum_{j=1}^{l} a'_j c_j^{(m+1)} + \frac{1}{2}\gamma^{-1} \sum_{j=1}^{l} a'_1 a'_j c_j^{(\rho)},$$

with nonlinear conjugate gradients.

The creation of an initial guess $a'^{(0)}$ for the minimization of f is entirely similar to the unpreconditioned case. Denoting $b_j'^{(1)} = \langle F(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)}), \phi_j^{(m+1)} \rangle$ $(\forall j = 1, \ldots, l)$, values

$$\begin{aligned}
&a_s'^{(e)} = \operatorname{sgn}(a_s'^{(r)}) \min \left(|a_s'^{(r)}|, \sqrt{|\lambda_s^{(m+1)}|}, \frac{|b_s'^{(1)}|}{\max\left(|c^{(t)}|, \| y_\perp^{(1)(m+1)} \|_{\tilde{P}^{-1}} \right)} \right), \\
&\forall j \neq s : a_j'^{(e)} = \operatorname{sgn}(a_j'^{(r)}) \min \left(|a_j'^{(r)}|, |a_s'^{(e)}| \right),
\end{aligned}$$

are calculated, with $a_j'^{(r)}$, $a_1'^{(p)}, \ldots, a_l'^{(p)}$, $a_1'^{(t)}, \ldots, a_l'^{(t)}$ and $c^{(t)}$ calculated as before (using the approximate null vectors $\phi_1^{(m+1)}, \ldots, \phi_l^{(m+1)}$ of the precondi-

tioned Jacobian $P(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$). A base transformation is again executed:

$$\forall j = 1, \ldots, l : a_j'^{(b)} = \sum_{i=1}^{l} D_{ij} a_i'^{(e)}.$$

With $c^{(q)}$ defined as for the unpreconditioned case, χ is calculated by

$$\chi = \min \left(1, \frac{|\lambda_s^{(m+1)}|}{|c^{(q)}|}, \frac{|\lambda_s^{(m+1)}||a_s'^{(e)}|}{c^{(q)2}}, \frac{|b_s'^{(1)}|}{|c^{(q)}| \max\left(|c^{(t)}|, \|y_\perp^{(1)(m+1)}\|_{\tilde{P}^{-1}}\right)} \right).$$

Defining $\tilde{\kappa}$ by (5.120), with $C_{kj} = \sum_{i=1}^{l} D_{ij} \lambda_i^{(m+1)} D_{ik}$ $(\forall j, k = 1, \ldots, l)$, values $a_1'^{(q)}, \ldots, a_l'^{(q)}$ are again calculated by

$$\forall j = 1, \ldots, l : a_j'^{(q)} = \tilde{\kappa} a_j'^{(b)}.$$

With $\Delta p^{(0)}$ defined as in the unpreconditioned case, the guess $a'^{(0)}$ is eventually constructed by application of

$$\forall j = 1, \ldots, l : a_j'^{(0)} = \chi a_j'^{(q)} + \Delta p^{(0)} a_j'^{(t)}. \tag{5.129}$$

5.8 When is reduction justified?

In section 5.6 slow convergence of the split block Newton method (SBN) is countered by extending the update vectors by additional parts Δx_z and Δp_z (see (5.70) and (5.72)). This lead to the split block Newton method with extra terms (SBNE), in which additional linear systems need to be solved.

Alternative parts Δx_ρ and Δp_ρ (see (5.114) and (5.116)) were derived in section 5.7, these did not require any solutions of additional linear systems. The method that uses these alternative parts was called the split block Newton method with reduced terms (SBNR). If assumption 5.18 is valid, an update with this alternative method should perform as well as an update with the one derived in section 5.6. If this is not the case, convergence should be similar to the normal split block Newton method (see section 5.5).

5.8.1 Derivation of a reduction criterion

To validate assumption 5.18, we will check whether $a_1, \ldots, a_l$ satisfy the relation

$$\forall i, j = 2, \ldots, l : |a_i a_j| \lesssim \max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} a_j C_{ij} \right|. \tag{5.130}$$

For this purpose the value

$$\kappa = \min \left(1, \frac{\max_{i=1,\ldots,l} \left| \sum_{j=1}^{l} \hat{a}_j' C_{kj} \right|}{\max_{i,j=2,\ldots,l} |\hat{a}_i' \hat{a}_j'|} \right), \tag{5.131}$$

with $\hat{a}'_1, \ldots, \hat{a}'_l$ defined as the arguments that minimize

$$
\begin{aligned}
f :& \mathbb{R}^l \to \mathbb{R} : \\
& a' \to \sum_{j=1}^{l} \left(b_j^{(1)} + \Delta p(a') b_j^{(2)} + \sum_{k=1}^{l} a'_k C_{jk} + \frac{1}{2} c_j^{(11)} a_1'^2 + \sum_{j=1}^{l} c_j^{(1k)} a'_1 a'_j \right)^2 \\
& \quad + \left(\Delta p(a') \| y_\perp^{(1)} \| \right)^2 + \sum_{j=1}^{l} \left(a'_j \| y_\perp^{(1)} \| \right)^2 + \Delta p(a')^4 + \sum_{j=1}^{l} \left(\Delta p(a') a'_j \right)^2 \\
& \quad + \sum_{k=1}^{l} \sum_{i=1}^{l} \sum_{j=1}^{l} \left(a'_k a'_i a'_j \right)^2, \\
\text{with } & \Delta p(a') = c_0 + \sum_{j=1}^{l} a'_j c_j + \frac{1}{2} \gamma^{-1} \sum_{j=1}^{l} a'_1 a'_j c_j^{(\rho)},
\end{aligned}
$$

is approximated. The values $\|y_\perp^{(1)}\|$, c_0, $c_1, \ldots, c_l$, $c_1^{(\rho)}, \ldots, c_l^{(\rho)}$, $b_1^{(1)}, \ldots, b_l^{(1)}$, $b_1^{(2)}, \ldots, b_l^{(2)}$, $c_1^{(1k)}, \ldots, c_l^{(1k)}$ $(\forall k = 1, \ldots, l)$ and γ that appear in the function f are defined as in section 5.7 (dropping the index $(m+1)$).

If $\kappa \approx 1$, relation (5.130) is satisfied. An update with the SBNR method is expected to perform as well as the SBNE one. For $\kappa \ll 1$, assumption 5.18 should not be made. In this case update vectors of the form (5.70) and (5.72) should be used instead of (5.114) and (5.116).

5.8.2 The method in practice

To incorporate a criterion that checks whether the reduction of the terms is valid in practice, we start each Newton iteration as if executing the SBNR method, up to the calculation of $a_1'^{(b)}, \ldots, a_l'^{(b)}$. The value κ is then approximated by $\tilde{\kappa}$, calculated from (5.120). If $\tilde{\kappa} \approx 1$, update vectors of the form (5.114) and (5.116) are considered. If $\tilde{\kappa} \ll 1$, an update with (5.70) and (5.72) will be performed.

Pseudo-code of the resulting method, which we call the *split block Newton method with mixed terms* (SBNM), is given by algorithm 5.7 on page 164 in appendix 5.10. Note that an update without additional terms is executed in the first iteration.

Algorithm 5.7 is applied in example 5.24. Convergence is achieved in 6 Newton steps, 4 of these were executed with reduced terms. Compared to the SBNE method, more Newton iterations are required to reach convergence. The computational work is however strongly reduced since the total amount of linear systems that need to be solved, is decreased.

Example 5.24. Consider the same set-up as in example 5.2. The SBNM method is applied, yielding the residual plot in figure 5.10. After 6 Newton iterations the residual norm converges (up to a tolerance of 10^{-12}). 2 of the 6 iterations were executed with extra terms, in 4 iterations reduced ones were used instead. ◇

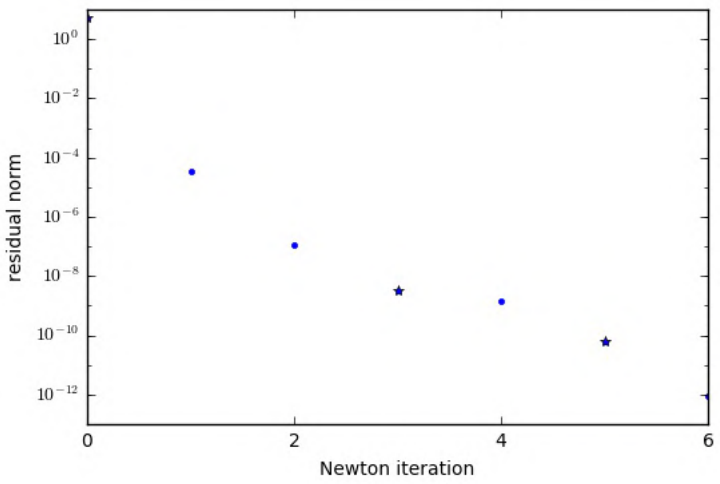


Figure 5.10: Residual plot of example 5.24. The SBNM method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. 4 Newton steps were executed with reduced terms (illustrated by circles), 3 with extra terms (illustrated by stars). The residual norm converges to approximately 10^{-12} after 6 iterations, which is the attainable accuracy. Compared to figure 4.8 more iterations are required to reach this value, but the total computational work is reduced.

Similar to the method without block elimination (see section 4.8), in the current section the robustness of the SBNE method was combined with the low computational cost of the SBNR one. The resulting algorithm is computationally efficient and still prevents slow convergence caused by possible ill-conditionedness of the partial Jacobian $F_x(\tilde{x}^{(m+1)}, \tilde{p}^{(m+1)})$.

5.8.3 The preconditioned case

The preconditioned version of the SBNM method is derived similarly to the unpreconditioned case. The value $\tilde{\kappa}$ used to check assumption 5.18 is still calculated from (5.120), this time using the preconditioned versions for the values $C_{11}, \ldots, C_{ll}$.

5.9 Discussion on convergence

Though convergence of the split block Newton method with extra (SBNE) or mixed (SBNM) terms generally behaves well in our applications, this is not always the case. This is the case for example 5.25, where the SBNE method converges slowly.

Example 5.25. A similar setting as for example 5.2 is considered: the same equation $\mathcal{H}$, inner product and preconditioner are used. The point $(\psi^{(0)}, \mu^{(0)})$

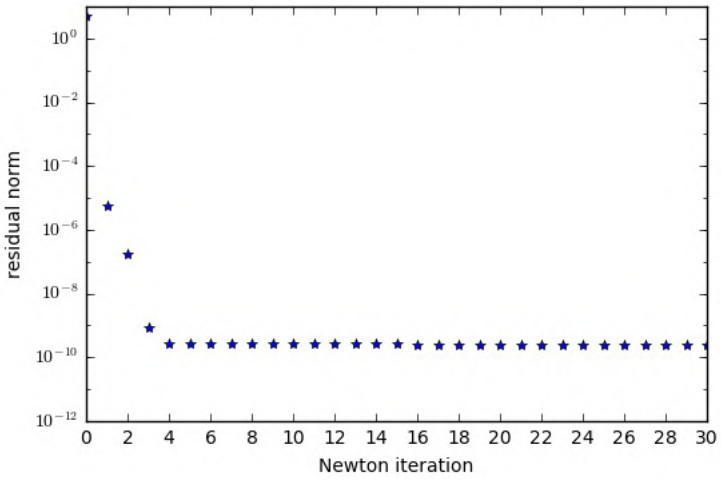


Figure 5.11: Residual plot of example 5.25. The SBNM method is applied to an ill-conditioned nonlinear problem, where the partial Jacobian contains a three-dimensional kernel in the searched solution. The residual norm converges to approximately 10^{-10} after 4 iterations. Further iterations show a very slow decrease in residual norm, the attainable accuracy (10^{-12}) is not reached in an acceptable amount of Newton steps. Note that the same method is applied for the residual plot of figure 5.8, where convergence up to the value 10^{-12} was observed.

is chosen on the line between $(\psi'^{(1)}, \mu'^{(1)})$, a solution of (2.20) for $\mu'^{(1)} = 1.06$, and $(\psi'^{(2)}, \mu'^{(2)})$, a solution for $\mu'^{(2)} = 1.059$. Both solutions lie on branch B in figure 9.23 (see section 9.5.3). We choose the point $(\psi^{(0)}, \mu^{(0)})$ such that $\mu^{(0)} = 1.0583$.

The SBNE method is applied, as an initial guess the point $(\tilde{\psi}^{(0)}, \mu^{(0)})$ is used, with $\tilde{\psi}^{(0)}$ a small perturbation of $\psi^{(0)}$. The residual plot of this problem is given by figure 5.11. The residual norms stagnate at a value of approximately 10^{-10} after 4 Newton iterations. ◇

Considering update vectors of the form (5.70) and (5.72), slow convergence can be clarified by the bound

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt{|\lambda_s|} \tag{5.132}$$

on the values $a_1, \ldots, a_l$ in the update vectors (see lemma 5.14). Since λ_s is the eigenvalue of an approximate null vector, the values that can be chosen for $a_1, \ldots, a_l$ are limited. Slow convergence occurs if decreasing the residual requires choices of $a_1, \ldots, a_l$ with $|a_j| \gg \sqrt{|\lambda_s|}$ for a certain $j \in \{1, \ldots, l\}$.

Similar to the methods without block elimination (see section 4.9), the analysis of introducing additional parts to the update vectors can be extended

to replace (5.132) with a less strict bound of the form

$$\forall j = 1, \ldots, l : |a_j| \lesssim \sqrt[3]{|\lambda_s|}. \tag{5.133}$$

This extension is however not practical and requires an additional amount of $\frac{1}{6}l(l+1)(4l+2)$ linear systems to be solved (see section 4.9). The analysis done in sections 5.7 and 5.8 would become invalid as well.

Instead of further adjusting the update vectors, we will use the SBNM method for our applications. Convergence problems are handled within the applications themselves. For example, the Newton step length adaptation algorithm (see section 6.6.2) will adjust its guesses when the Newton method does not reach convergence within a specified amount of iterations.

5.10 Appendix

The standard block Newton method

Algorithm 5.1 NewtonBlockStandard

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

1: Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: $\tilde{x} = x^{(0)}$
5: $\tilde{p} = p^{(0)}$
6: $r_F = F(\tilde{x}, \tilde{p})$
7: $r_G = G(\tilde{x}, \tilde{p})$
8: $i = 0$
9: **while** $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
10: $\quad i \leftarrow i + 1$
11: $\quad$ Calculate $y^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$
12: $\quad$ Calculate $y^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = F_p(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$
13: $\quad \Delta p = \frac{-r_G - G_x(\tilde{x},\tilde{p})y^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(2)}}$
14: $\quad \Delta x = y^{(1)} - \Delta p y^{(2)}$
15: $\quad \tilde{x} \leftarrow \tilde{x} + \Delta x$
16: $\quad \tilde{p} \leftarrow \tilde{p} + \Delta p$
17: $\quad r_F = F(\tilde{x}, \tilde{p})$
18: $\quad r_G = G(\tilde{x}, \tilde{p})$
19: **end while**
20: Return $\tilde{x}, \tilde{p}$

The deflated block Newton method

Algorithm 5.2 NewtonBlockDeflated

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

1: Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified

4: $\tilde{x} = x^{(0)}$
5: $\tilde{p} = p^{(0)}$
6: $r_F = F(\tilde{x}, \tilde{p})$
7: $r_G = G(\tilde{x}, \tilde{p})$
8: $i = 0$
9: **while** $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
10: $i \leftarrow i + 1$
11: Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
12: Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$ and K
13: $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
14: Calculate $y^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
15: Calculate $y^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = F_p(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
16: $\Delta p = \frac{-r_G - G_x(\tilde{x},\tilde{p}) y^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y^{(2)}}$
17: $\Delta x = y^{(1)} - \Delta p y^{(2)}$
18: $\tilde{x} \leftarrow \tilde{x} + \Delta x$
19: $\tilde{p} \leftarrow \tilde{p} + \Delta p$
20: $r_F = F(\tilde{x}, \tilde{p})$
21: $r_G = G(\tilde{x}, \tilde{p})$
22: **end while**
23: Return $\tilde{x}, \tilde{p}$

The block Newton method with line search (BNLS)

Algorithm 5.3 NewtonBlockLS

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, F_{xx}, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

1: Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
2: Define F_{xx} by (5.6) if not specified
3: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
4: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
5: $\tilde{x} = x^{(0)}$
6: $\tilde{p} = p^{(0)}$
7: $r_F = F(\tilde{x}, \tilde{p})$
8: $r_G = G(\tilde{x}, \tilde{p})$
9: $i = 0$
10: **while** $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
11: $i \leftarrow i + 1$
12: Calculate $y^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$

13: Calculate $y^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = F_p(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$
14: $\Delta x = y^{(1)} - \frac{-r_G - G_x(\tilde{x},\tilde{p})y^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y^{(2)}} y^{(2)}$
15: $c_0 = \frac{-r_G}{G_p(\tilde{x},\tilde{p})}$
16: $c_1 = \frac{-G_x(\tilde{x},\tilde{p})\Delta x}{G_p(\tilde{x},\tilde{p})}$
17: Define g, g' and g'' as in (5.33), (5.34) and (5.35)
18: $\gamma = \min(2\|\Delta x\|^{-2}_{P(\tilde{x},\tilde{p})^{-1}}, |c_1|^{-1}\|\Delta x\|^{-1}_{P(\tilde{x},\tilde{p})^{-1}}, 2|c_1|^{-2}, c_0|^{-1}\|\Delta x\|^{-1}_{P(\tilde{x},\tilde{p})^{-1}}, |c_0|^{-1}|c_1|^{-1})$
19: $\xi^{(0)} = \min\left(1, \|r_F\|_{P(\tilde{x},\tilde{p})}\gamma\right)$
20: Calculate $\tilde{\xi}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \xi^{(0)}$ and given g
21: $\tilde{x} \leftarrow \tilde{x} + \xi\Delta x$
22: $\tilde{p} \leftarrow \tilde{p} + c_0 + \xi c_1$
23: $r_F = F(\tilde{x}, \tilde{p})$
24: $r_G = G(\tilde{x}, \tilde{p})$
25: **end while**
26: Return $\tilde{x}, \tilde{p}$

The split block Newton method (SBN)

Algorithm 5.4 NewtonBlockSplit

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, F_{xx}, F_{xp}, F_{pp}, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

1: Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
2: Define F_{xx}, F_{xp}, F_{pp} by (5.6), (5.7) and (5.8) if not specified
3: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
4: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
5: $\tilde{x} = x^{(0)}$
6: $\tilde{p} = p^{(0)}$
7: $r_F = F(\tilde{x}, \tilde{p})$
8: $r_G = G(\tilde{x}, \tilde{p})$
9: $i = 0$
10: **while** $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
11: $i \leftarrow i + 1$
12: Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
13: Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$ and K
14: $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
15: $W \leftarrow \begin{pmatrix} P(\tilde{x}, \tilde{p})K & W \end{pmatrix}$

16: Calculate $y_\perp^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
17: $r_p = F_p(\tilde{x}, \tilde{p})$
18: Calculate $y_\perp^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = r_p$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
19: $c_0 = \frac{-r_G - G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}$
20: **for** $j = 1, \ldots, l$ **do**
21: $b_j^{(1)} = \langle r_F, W_j \rangle$
22: $b_j^{(2)} = \langle r_p, W_j \rangle$
23: $c_j = \frac{-G_x(\tilde{x}, \tilde{p}) W_j}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}$
24: **end for**
25: Calculate $a'^{(0)}$ by formula (5.64) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
26: Define f by (5.63) (with $\lambda_j = L_{jj}$)
27: Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
28: Define g, g' and g'' by (5.60), (5.61) and (5.62) (with $\phi_j = W_j$)
29: Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
30: $\Delta p = c_0 + \sum_{j=1}^{l} \tilde{a}_j c_j$
31: $\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^{l} \tilde{a}_j W_j$
32: $\tilde{x} \leftarrow \tilde{x} + \Delta x$
33: $\tilde{p} \leftarrow \tilde{p} + \Delta p$
34: $r_F = F(\tilde{x}, \tilde{p})$
35: $r_G = G(\tilde{x}, \tilde{p})$
36: **end while**
37: Return $\tilde{x}, \tilde{p}$

The split block Newton method with extra terms (SBNE)

Algorithm 5.5 NewtonBlockExtra

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, F_{xx}, F_{xp}, F_{pp}, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

1: Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
2: Define F_{xx}, F_{xp}, F_{pp} by (5.6), (5.7) and (5.8) if not specified
3: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
4: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
5: $\tilde{x} = x^{(0)}$
6: $\tilde{p} = p^{(0)}$
7: $r_F = F(\tilde{x}, \tilde{p})$
8: $r_G = G(\tilde{x}, \tilde{p})$
9: $i = 0$

while $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$
  Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
  Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$ and K
  $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
  $W \leftarrow \begin{pmatrix} P(\tilde{x}, \tilde{p})K & W \end{pmatrix}$
  Calculate $y_\perp^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  $r_p = F_p(\tilde{x}, \tilde{p})$
  Calculate $y_\perp^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = r_p$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  for $j = 1, \ldots, l$ **do**
    for $k = 1, \ldots, j$ **do**
      $v_{jk} = F_{xx}(\tilde{x}, \tilde{p}) W_j W_k$
      Calculate z_{jk} by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = v_{jk}$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
      $z_{kj} = z_{jk}$
      for $q = 1, \ldots, l$ **do**
        $c_q^{(jk)} = \langle v_{jk}, W_q \rangle$
        $c_q^{(kj)} = c_q^{(jk)}$
      end for
    end for
  end for
  $c_0 = \frac{-r_G - G_x(\tilde{x}, \tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}$
  for $j = 1, \ldots, l$ **do**
    $b_j^{(1)} = \langle r_F, W_j \rangle$
    $b_j^{(2)} = \langle r_p, W_j \rangle$
    $c_j = \frac{-G_x(\tilde{x}, \tilde{p}) W_j}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}$
    for $k = 1, \ldots, j$ **do**
      $c_{jk}^{(z)} = \frac{-G_x(\tilde{x}, \tilde{p}) z_{ij}}{G_p(\tilde{x}, \tilde{p}) - G_x(\tilde{x}, \tilde{p}) y_\perp^{(2)}}$
      $c_{kj}^{(z)} = c_{jk}^{(z)}$
    end for
  end for
  Calculate $a'^{(0)}$ by formula (5.86) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
  Define f by (5.85) (with $\lambda_j = L_{jj}$)
  Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
  for $j = 1, \ldots, l$ **do**
    for $k = 1, \ldots, j$ **do**
      $\alpha_{jk}^{(0)} = -\frac{1}{2} \tilde{a}_j^{(0)} \tilde{a}_k^{(0)}$
      $\alpha_{kj}^{(0)} = \alpha_{jk}^{(0)}$
    end for
  end for

Define g, g' and g'' by (5.82), (5.83) and (5.84) (with $\phi_j = W_j$)
Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
$\Delta p = c_0 + \sum_{j=1}^{l} \tilde{a}_j c_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \tilde{\alpha}_{ij} c_{ij}^{(z)}$
$\Delta x = y_{\perp}^{(1)} - \Delta p y_{\perp}^{(2)} + \sum_{j=1}^{l} \tilde{a}_j W_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \tilde{\alpha}_{ij} z_{ij}$
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$\tilde{p} \leftarrow \tilde{p} + \Delta p$
$r_F = F(\tilde{x}, \tilde{p})$
$r_G = G(\tilde{x}, \tilde{p})$
end while
Return $\tilde{x}, \tilde{p}$

The split block Newton method with reduced terms (SBNR)

Algorithm 5.6 NewtonBlockReduced

Input $m_{New} \in \mathbb{N}$, tolerance $\epsilon_{New} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, F_{xx}, F_{xp}, F_{pp}, inner product $\langle \cdot, \cdot \rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x, p) = 0$, $G(x, p) = 0$

Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
Define F_{xx}, F_{xp}, F_{pp} by (5.6), (5.7) and (5.8) if not specified
Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
$\tilde{x} = x^{(0)}$
$\tilde{p} = p^{(0)}$
$r_F = F(\tilde{x}, \tilde{p})$
$r_G = G(\tilde{x}, \tilde{p})$
$i = 0$
while $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
$i \leftarrow i + 1$
Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$ and K
$U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
$W \leftarrow \begin{pmatrix} P(\tilde{x}, \tilde{p})K & W \end{pmatrix}$
Calculate $y_{\perp}^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
$r_p = F_p(\tilde{x}, \tilde{p})$
Calculate $y_{\perp}^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = r_p$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
if $i = 1$ **then**
$\breve{W} = W$
$\breve{U} = U$

$c_0 = \frac{-r_G - G_x(\tilde{x},\tilde{p})y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)}}$
for $j = 1, \dots, l$ **do**
$b_j^{(1)} = \langle r_F, W_j \rangle$
$b_j^{(2)} = \langle r_p, W_j \rangle$
$c_j = \frac{-G_x(\tilde{x},\tilde{p})W_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)}}$
end for
Calculate $a'^{(0)}$ by formula (5.64) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
Define f by (5.63) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
Define g, g' and g'' by (5.60), (5.61) and (5.62) (with $\phi_j = W_j$)
Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
$\Delta p = c_0 + \sum_{j=1}^{l} \tilde{a}_j c_j$
$\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^{l} \tilde{a}_j W_j$
else
$\mathring{W} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle W_j$
$\mathring{U} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle U_j$
$\beta = \sqrt{|\langle \mathring{W}, \mathring{U} \rangle|}$
$\breve{W}_1 = \beta^{-1}\mathring{W}$
$\breve{U}_1 = \beta^{-1}\mathring{U}$
Create $\breve{W}_2, \dots, \breve{W}_l$ and $\breve{U}_2, \dots, \breve{U}_l$ as linear combinations of respectively $W_1, \dots, W_l$ and $U_1, \dots, U_l$ such that $\langle \breve{U}_j, \breve{W}_k \rangle = \delta_{jk}$ and $P(\tilde{x},\tilde{p})\breve{U}_j = \breve{W}_j$ for each $j, k = 1, \dots, l$
for $j = 1, \dots, l$ **do**
for $k = 1, \dots, l$ **do**
$D_{jk} = \langle \breve{W}_k, U_j \rangle$
end for
end for
$C = D^T L D$
for $j = 1, \dots, l$ **do**
$\rho'_j = \breve{W}_j$
for $k = 1, \dots, l$ **do**
$\rho'_j \leftarrow \rho'_j - \hat{W}_k \langle \rho'_j, \hat{U}_k \rangle$
end for
for $k = 1, \dots, l$ **do**
$\rho'_j \leftarrow \rho'_j - W_k \langle \rho'_j, U_k \rangle$
end for
end for
$c_0 = \frac{-r_G - G_x(\tilde{x},\tilde{p})y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p})y_\perp^{(2)}}$
for $j = 1, \dots, l$ **do**
$b_j^{(1)} = \langle r_F, \breve{W}_j \rangle$
$b_j^{(2)} = \langle r_p, \breve{W}_j \rangle$

$c_j = \frac{-G_x(\tilde{x},\tilde{p})\breve{W}_j}{G_p(\tilde{x},\tilde{p})-G_x(\tilde{x},\tilde{p})y_\perp^{(2)}}$
$c_j^{(\rho)} = \frac{-G_x(\tilde{x},\tilde{p})\rho_j'}{G_p(\tilde{x},\tilde{p})-G_x(\tilde{x},\tilde{p})y_\perp^{(2)}}$
$v_j = F_{xx}(\tilde{x},\tilde{p})\breve{W}_1\breve{W}_j$
for $k = 1,\dots,l$ **do**
$c_k^{(1j)} = \langle v_j, \breve{W}_k\rangle$
end for
end for
Calculate $a'^{(0)}$ by formula (5.129) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
Define f by (5.128) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
for $j = 1,\dots,l$ **do**
$\alpha_j^{(0)} = -\frac{1}{2}\tilde{a}_1^{(0)}\tilde{a}_j^{(0)}$
end for
Define g, g' and g'' by (5.125), (5.126) and (5.127) (with $\phi_j = W_j$)
Calculate $(\tilde{a},\tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)},\alpha^{(0)})$ and given g
$\Delta p = c_0 + \sum_{j=1}^l \tilde{a}_j c_j - \gamma^{-1}\sum_{j=1}^l \tilde{\alpha}_j c_j^{(\rho)}$
$\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^l \tilde{a}_j\breve{W}_j - \gamma^{-1}\sum_{j=1}^l \tilde{\alpha}_j\rho_j'$
end if
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$\tilde{p} \leftarrow \tilde{p} + \Delta p$
$r_F = F(\tilde{x},\tilde{p})$
$r_G = G(\tilde{x},\tilde{p})$
$\hat{U} \leftarrow U$
$\hat{W} \leftarrow W$
$\Delta x_\parallel = \sum_{j=1}^l \tilde{a}_j\breve{W}_j$
$\gamma \leftarrow \sqrt{\langle\sum_{j=1}^l \tilde{a}_j\breve{W}_j, \sum_{j=1}^l \tilde{a}_j\breve{U}_j\rangle}$
end while
Return $\tilde{x},\tilde{p}$

The split block Newton method with mixed terms (SBNM)

Algorithm 5.7 NewtonBlockMixed

Input $m_{New} \in \mathbb{N}$, tolerances $\epsilon_{New}, \epsilon_{\tilde{\kappa}} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $G : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R}$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives F_x, F_p, G_x, G_p, F_{xx}, F_{xp}, F_{pp}, inner product $\langle\cdot,\cdot\rangle$, initial guesses $x^{(0)} \in \mathbb{C}^n$, $p^{(0)} \in \mathbb{R}$

Output Approximations $\tilde{x}$, $\tilde{p}$ for $F(x,p) = 0$, $G(x,p) = 0$

Define F_x, G_x, F_p, G_p by (5.2), (5.3), (5.4) and (5.5) if not specified
Define F_{xx}, F_{xp}, F_{pp} by (5.6), (5.7) and (5.8) if not specified
Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x,p) \to \mathcal{I}$ if not specified
Set $\langle\cdot,\cdot\rangle = \langle\cdot,\cdot\rangle_2$ if not specified

$\tilde{x} = x^{(0)}$
$\tilde{p} = p^{(0)}$
$r_F = F(\tilde{x}, \tilde{p})$
$r_G = G(\tilde{x}, \tilde{p})$
$i = 0$
while $i < m_{New}$ and $\sqrt{\|r_F\|^2_{P(\tilde{x},\tilde{p})} + r_G^2} > \epsilon_{New}$ **do**
  $i \leftarrow i + 1$
  Initialize deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 known null vectors
  Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$ and given $\langle \cdot, \cdot \rangle$ and K
  $U \leftarrow \begin{pmatrix} K & U \end{pmatrix}$
  $W \leftarrow \begin{pmatrix} P(\tilde{x}, \tilde{p})K & W \end{pmatrix}$
  Calculate $y_\perp^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = -r_F$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  $r_p = F_p(\tilde{x}, \tilde{p})$
  Calculate $y_\perp^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = r_p$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  if $i = 1$ **then**
    $\breve{W} = W$
    $\breve{U} = U$
    $c_0 = \frac{-r_G - G_x(\tilde{x},\tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
    for $j = 1, \ldots, l$ **do**
      $b_j^{(1)} = \langle r_F, W_j \rangle$
      $b_j^{(2)} = \langle r_p, W_j \rangle$
      $c_j = \frac{-G_x(\tilde{x},\tilde{p}) W_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
    end for
    Calculate $a'^{(0)}$ by formula (5.64) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
    Define f by (5.63) (with $\lambda_j = L_{jj}$)
    Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
    Define g, g' and g'' by (5.60), (5.61) and (5.62) (with $\phi_j = W_j$)
    Calculate $\tilde{a}$ by executing NCG (algorithm 3.5) with $a^{(0)} = \tilde{a}^{(0)}$ and given g
    $\Delta p = c_0 + \sum_{j=1}^{l} \tilde{a}_j c_j$
    $\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^{l} \tilde{a}_j W_j$
  else
    $\mathring{W} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle W_j$
    $\mathring{U} = \sum_{j=1}^{l} \langle U_j, \Delta x_\parallel \rangle U_j$
    $\beta = \sqrt{|\langle \mathring{W}, \mathring{U} \rangle|}$
    $\breve{W}_1 = \beta^{-1} \mathring{W}$
    $\breve{U}_1 = \beta^{-1} \mathring{U}$
    Create $\breve{W}_2, \ldots, \breve{W}_l$ and $\breve{U}_2, \ldots, \breve{U}_l$ as linear combinations of respectively $W_1, \ldots, W_l$ and $U_1, \ldots, U_l$ such that $\langle \breve{U}_j, \breve{W}_k \rangle = \delta_{jk}$ and $P(\tilde{x}, \tilde{p}) \breve{U}_j = \breve{W}_j$ for each $j, k = 1, \ldots, l$

for $j = 1, \ldots, l$ **do**
  for $k = 1, \ldots, l$ **do**
    $D_{jk} = \langle \breve{W}_k, U_j \rangle$
  end for
end for
$C = D^T L D$
Calculate $\tilde{\kappa}$ by formula (5.120) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
if $\tilde{\kappa} > \epsilon_{\tilde{\kappa}}$ **then**
  for $j = 1, \ldots, l$ **do**
    $\rho'_j = \breve{W}_j$
    for $k = 1, \ldots, l$ **do**
      $\rho'_j \leftarrow \rho'_j - \hat{W}_k \langle \rho'_j, \hat{U}_k \rangle$
    end for
    for $k = 1, \ldots, l$ **do**
      $\rho'_j \leftarrow \rho'_j - W_k \langle \rho'_j, U_k \rangle$
    end for
  end for
  $c_0 = \frac{-r_G - G_x(\tilde{x},\tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
  for $j = 1, \ldots, l$ **do**
    $b_j^{(1)} = \langle r_F, \breve{W}_j \rangle$
    $b_j^{(2)} = \langle r_p, \breve{W}_j \rangle$
    $c_j = \frac{-G_x(\tilde{x},\tilde{p}) \breve{W}_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
    $c_j^{(\rho)} = \frac{-G_x(\tilde{x},\tilde{p}) \rho'_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
    $v_j = F_{xx}(\tilde{x},\tilde{p}) \breve{W}_1 \breve{W}_j$
    for $k = 1, \ldots, l$ **do**
      $c_k^{(1j)} = \langle v_j, \breve{W}_k \rangle$
    end for
  end for
  Calculate $a'^{(0)}$ by formula (5.129) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
  Define f by (5.128) (with $\lambda_j = L_{jj}$)
  Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
  for $j = 1, \ldots, l$ **do**
    $\alpha_j^{(0)} = -\frac{1}{2} \tilde{a}_1^{(0)} \tilde{a}_j^{(0)}$
  end for
  Define g, g' and g'' by (5.125), (5.126) and (5.127) (with $\phi_j = W_j$)
  Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
  $\Delta p = c_0 + \sum_{j=1}^l \tilde{a}_j c_j - \gamma^{-1} \sum_{j=1}^l \tilde{\alpha}_j c_j^{(\rho)}$
  $\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^l \tilde{a}_j \breve{W}_j - \gamma^{-1} \sum_{j=1}^l \tilde{\alpha}_j \rho'_j$
else
  for $j = 1, \ldots, l$ **do**
    for $k = 1, \ldots, j$ **do**
      $v_{jk} = F_{xx}(\tilde{x},\tilde{p}) W_j W_k$

Calculate z_{jk} by executing GMRES (algorithm 3.4) with $\mathcal{A} = F_x(\tilde{x}, \tilde{p})$, $b = v_{jk}$, $\mathcal{P} = P(\tilde{x}, \tilde{p})$, $K = U$ and given $\langle \cdot, \cdot \rangle$
  $z_{kj} = z_{jk}$
  for $q = 1, \ldots, l$ **do**
    $c_q^{(jk)} = \langle v_{jk}, W_q \rangle$
    $c_q^{(kj)} = c_q^{(jk)}$
  end for
 end for
end for
$c_0 = \frac{-r_G - G_x(\tilde{x},\tilde{p}) y_\perp^{(1)}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
for $j = 1, \ldots, l$ **do**
  $b_j^{(1)} = \langle r_F, W_j \rangle$
  $b_j^{(2)} = \langle r_p, W_j \rangle$
  $c_j = \frac{-G_x(\tilde{x},\tilde{p}) W_j}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
  for $k = 1, \ldots, j$ **do**
    $c_{jk}^{(z)} = \frac{-G_x(\tilde{x},\tilde{p}) z_{ij}}{G_p(\tilde{x},\tilde{p}) - G_x(\tilde{x},\tilde{p}) y_\perp^{(2)}}$
    $c_{kj}^{(z)} = c_{jk}^{(z)}$
  end for
end for
Calculate $a'^{(0)}$ by formula (5.86) (with $\lambda_j = L_{jj}$ and $\phi_j = W_j$)
Define f by (5.85) (with $\lambda_j = L_{jj}$)
Calculate $\tilde{a}^{(0)}$ by executing NCG (algorithm 3.5) with $g = f$ and $a^{(0)} = a'^{(0)}$
for $j = 1, \ldots, l$ **do**
  for $k = 1, \ldots, j$ **do**
    $\alpha_{jk}^{(0)} = -\frac{1}{2} \tilde{a}_j^{(0)} \tilde{a}_k^{(0)}$
    $\alpha_{kj}^{(0)} = \alpha_{jk}^{(0)}$
  end for
end for
Define g, g' and g'' by (5.82), (5.83) and (5.84) (with $\phi_j = W_j$)
Calculate $(\tilde{a}, \tilde{\alpha})$ by executing NCG (algorithm 3.5) with $a^{(0)} = (\tilde{a}^{(0)}, \alpha^{(0)})$ and given g
$\Delta p = c_0 + \sum_{j=1}^{l} \tilde{a}_j c_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \tilde{\alpha}_{ij} c_{ij}^{(z)}$
$\Delta x = y_\perp^{(1)} - \Delta p y_\perp^{(2)} + \sum_{j=1}^{l} \tilde{a}_j W_j + \sum_{i=1}^{l} \sum_{j=1}^{l} \tilde{\alpha}_{ij} z_{ij}$
end if
end if
$\tilde{x} \leftarrow \tilde{x} + \Delta x$
$\tilde{p} \leftarrow \tilde{p} + \Delta p$
$r_F = F(\tilde{x}, \tilde{p})$
$r_G = G(\tilde{x}, \tilde{p})$
$\hat{U} \leftarrow U$
$\hat{W} \leftarrow W$
$\Delta x_\parallel = \sum_{j=1}^{l} \tilde{a}_j \breve{W}_j$

124: $\gamma \leftarrow \sqrt{\langle \sum_{j=1}^{l} \tilde{a}_j \breve{W}_j, \sum_{j=1}^{l} \tilde{a}_j \breve{U}_j \rangle}$
125: **end while**
126: Return $\tilde{x}, \tilde{p}$

CHAPTER 6

Calculation of solution curves

"I'm not crazy. My mother had me tested."
*– **Sheldon Cooper** –*
The Big Bang Theory

Chapter highlights:

- We review the theoretical existence of solution curves, and extend the theory for dynamical systems with (continuous) symmetry.
- We explain how a solution curve is approximated by the pseudo-arclength continuation algorithm, and provide a technique to counter the false reversion that might occur for systems with continuous symmetries.
- We show how bifurcation points are detected along the curve, and provide two strategies to approximate these points.
- We discuss how bifurcation points are used to simplify the problem of determining the physical stability of equilibria.
- The results in this chapter are mainly based on the following references: [3, 10, 48, 62, 32, 72].

6.1 Introduction

With the Newton methods described in chapter 4, it is possible to calculate steady states of physical systems. For given, fixed values of the physical parameters, the methods approximate the solutions of a steady state equation $\mathcal{F}(\psi, \mu)=0$ (for a given $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $n \in \mathbb{N}$).

We are however not necessarily interested in the solutions of these equations themselves, but rather how these steady states change when certain physical parameters are perturbed. Based on the implicit function theorem, these dependencies are described by continuous solution curves.

Definition 6.1 (Solution curve)**.** Given an interval $I \subset \mathbb{R}$ and functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $\psi : I \to \mathbb{C}^n$ and $\mu : I \to \mathbb{R}$. We call $(\psi(s), \mu(s))$ a solution curve of $\mathcal{F}$ if

$$\forall s \in I : \mathcal{F}(\psi(s), \mu(s)) = 0.$$

The solution curve is continuous if both ψ and μ are.

The effect on the steady state ψ of perturbing the physical parameter μ, is examined by varying the value s in above definition. In practice solution curves will be approximated by the pseudo-arclength continuation algorithm, which provides a finite set of points $(\psi^{(0)}, \mu^{(0)}), (\psi^{(1)}, \mu^{(1)}), (\psi^{(2)}, \mu^{(2)}), \ldots$ that belong to the curve. The block Newton methods derived in chapter 5 will form an essential part of this algorithm.

This chapter will focus on the construction of solution curves, as well as the calculation of bifurcation points.

Definition 6.2 (Bifurcation point)**.** Given a function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$. A point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$ is called a bifurcation point of $\mathcal{F}$ if $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ contains an eigenvalue with zero real part:

$$\exists \phi \in \mathbb{C}^n, \lambda \in \mathbb{R} : \mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\phi = \lambda\phi, \mathfrak{Re}(\lambda) = 0.$$

Bifurcation points have multiple uses: they are for example essential when considering automatic exploration techniques (see chapter 7), and also mark transitions in physical stability of steady states.

For the applications described in chapter 2, the partial derivatives $\mathcal{F}_\psi$ are self-adjoint with respect to a provided inner product. As a consequence, definition 6.2 simplifies: a point $(\psi^{(b)}, \mu^{(b)})$ is a bifurcation point if $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ is singular. As discussed in chapter 5, this property complicates the application of Newton methods.

In this chapter we will consider a general (nonlinear) function

$$\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu),$$

with a Hermitian partial Jacobian $\mathcal{F}_\psi$. The pseudo-arclength continuation algorithm is derived, and methods to calculate bifurcation points of $\mathcal{F}$ are provided. The chapter ends with one of the applications of bifurcation points: the determination of physical stability of steady states along a solution curve.

6.2 The implicit function theorem

The implicit function theorem (IFT) guarantees the existence of solution curves (see definition 6.1) under certain conditions.

Theorem 6.3 (Implicit function theorem [59, 62, 32])**.** Let $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$ and $(\psi^{(0)}, \mu^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$ satisfy:

- $\mathcal{F}(\psi^{(0)}, \mu^{(0)}) = 0$.
- $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ is nonsingular with bounded inverse.

- $\mathcal{F}$ and $\mathcal{F}_\psi$ are locally Lipschitz continuous in $S_\rho((\psi^{(0)}, \mu^{(0)}))$ for a certain $\rho \in \mathbb{R}_0^+$.

Then there exists $\delta \in \mathbb{R}$ with $0 < \delta \leq \rho$ and an unique function $\psi^{(i)} : S_\delta(\mu^{(0)}) \to \mathbb{C}^n$ that is continuous on $S_\delta(\mu^{(0)})$ with $\psi^{(i)}(\mu^{(0)}) = \psi^{(0)}$ such that

$$\forall \mu \in S_\delta(\mu^{(0)}) : \mathcal{F}(\psi^{(i)}(\mu), \mu) = 0.$$

Proof. See Doedel [32]. □

The applications considered in chapter 2 satisfy the third condition of theorem 6.3 for any point $(\psi^{(0)}, \mu^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$. The implicit function theorem is applicable for each $(\psi^{(0)}, \mu^{(0)})$ that satisfies $\mathcal{F}(\psi^{(0)}, \mu^{(0)}) = 0$ and is not a bifurcation of the equation. Given such a point, it states that an implicit function of the form

$$\psi^{(i)} : S_\delta(\mu^{(0)}) \to \mathbb{C}^n : \mu \to \psi^{(i)}(\mu) \tag{6.1}$$

exists such that $(\psi^{(i)}(\mu), \mu)$ is a zero of $\mathcal{F}$ for each $\mu \in S_\delta(\mu^{(0)})$. The value δ is chosen sufficiently small such that no bifurcation points appear in $(\psi^{(i)}(\mu), \mu)$. Note that the IFT can be reapplied to each point $(\psi^{(i)}(\mu), \mu)$ for $\mu \in S_\delta(\mu^{(0)})$. By applying the theorem multiple times, functions (6.1) are combined to create an implicit function of the form

$$\psi^{(i)} :]\delta_1, \delta_2[\to \mathbb{C}^n : \mu \to \psi^{(i)}(\mu), \tag{6.2}$$

with $\delta_1, \delta_2 \in \mathbb{R}_0^+$ and $\psi^{(i)}$ continuous. If the domain is finite, it is contained between two values that yield bifurcation points. The interval $]\delta_1, \delta_2[$ consists of an union of intervals of the form $S_\delta(\mu^{(0)})$, for certain values of $\delta \in \mathbb{R}_0^+$ and $\mu^{(0)} \in \mathbb{R}$. If $|\delta_1| \neq \infty$, $|\delta_2| \neq \infty$, the points $\lim_{\mu \to \delta_1} (\psi^{(i)}(\mu), \mu)$ and $\lim_{\mu \to \delta_2} (\psi^{(i)}(\mu), \mu)$ are bifurcations of the equation. Note that multiple functions of the form (6.2) can exist, depending on the point $(\psi^{(0)}, \mu^{(0)})$ used for the construction.

Segments of solution curves $(\psi(s), \mu(s))$ are created from these functions by defining

$$\begin{aligned} &\psi :]0, 1[\to \mathbb{C}^n : s \to \psi^{(i)}(\mu(s)), \\ &\mu :]0, 1[\to \mathbb{R} : s \to s\delta_2 + (1 - s)\delta_1. \end{aligned} \tag{6.3}$$

Theorem 6.4 analyses the continuous differentiability of these segments.

Theorem 6.4. Let $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$ and $(\psi^{(0)}, \mu^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$ satisfy the conditions of theorem 6.4. If the partial derivative $\mathcal{F}_\mu$ is continuous in $S_\rho((\psi^{(0)}, \mu^{(0)}))$, then the function $\psi^{(i)}$, defined by (6.1), has a continuous derivative on $S_\delta(\mu^{(0)})$.

Proof. See Doedel [32]. □

The extra condition given in theorem 6.4 is satisfied by the considered applications of chapter 2 as well. The implicit functions implied by the IFT will always be continuously differentiable. As a consequence, each point on a solution curve segment (defined by (6.3)) has got an unique tangent direction.

Definition 6.5. Consider a curve segment of the form (6.3). The tangent direction in a point $(\psi(s), \mu(s))$ (for a certain $s \in]0,1[$) is defined as $(\dot{\psi}(s), \dot{\mu}(s))$, with

$$\begin{aligned}\dot{\psi} &:]0,1[\to \mathbb{C}^n : s \to \frac{d\psi^{(i)}(\mu(s))}{d\mu},\\ \dot{\mu} &:]0,1[\to \mathbb{R} : s \to 1.\end{aligned}$$

Every steady state of $\mathcal{F}$ that is not a bifurcation, belongs to a single curve segment of the form (6.3). Bifurcation points are still excluded.

Definition 6.6 (Isolated bifurcation point)**.** Given a function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$ and a bifurcation point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$ of $\mathcal{F}$. The point $(\psi^{(b)}, \mu^{(b)})$ is called isolated if there exists a $\rho \in \mathbb{R}_0^+$ such that $(\psi^{(b)}, \mu^{(b)})$ is the unique bifurcation point in $S_\rho((\psi^{(b)}, \mu^{(b)}))$.

To include isolated bifurcation points, we will construct solution curves by combining several segments. Two segments, denoted by $(\psi^{(1)}(s), \mu^{(1)}(s))$ and $(\psi^{(2)}(s), \mu^{(2)}(s))$ with

$$\begin{aligned}\psi^{(1)} &:]0,1[\to \mathbb{C}^n : s \to \psi^{(i)}(\mu^{(1)}(s)), \quad \mu^{(1)} :]0,1[\to \mathbb{R} : s \to s\delta_2^{(1)} + (1-s)\delta_1^{(1)},\\ \psi^{(2)} &:]0,1[\to \mathbb{C}^n : s \to \psi^{(i)}(\mu^{(2)}(s)), \quad \mu^{(2)} :]0,1[\to \mathbb{R} : s \to s\delta_2^{(2)} + (1-s)\delta_1^{(2)},\end{aligned}$$

are combined when $\exists j, k \in \{0;1\}$ such that

$$\begin{aligned}\lim_{s\to j}(\psi^{(1)}(s), \mu^{(1)}(s)) &= \lim_{s\to k}(\psi^{(2)}(s), \mu^{(2)}(s)),\\ \lim_{s\to j}(\dot{\psi}^{(1)}(s), \dot{\mu}^{(1)}(s)) &= \lim_{s\to k}(\dot{\psi}^{(2)}(s), \dot{\mu}^{(2)}(s)).\end{aligned}$$

In this case the combined segment becomes $(\psi(s), \mu(s))$, defined by

$$\begin{aligned}\psi &:]0,1[\to \mathbb{C}^n : s \to \lim_{s'\to s} \psi^{(i)}(\mu(s')),\\ \mu &:]0,1[\to \mathbb{R} : s \to \begin{cases} 2s\delta_2 + (1-2s)\delta_1 & \text{for } s \le \frac{1}{2},\\ (2s-1)\delta_3 + (2-2s)\delta_2 & \text{for } s > \frac{1}{2},\end{cases}\end{aligned}$$

with $\delta_2 \in \{\delta_1^{(1)}, \delta_2^{(1)}, \delta_1^{(2)}, \delta_2^{(2)}\}$ the common value of the intervals $[\delta_1^{(1)}, \delta_2^{(1)}]$ and $[\delta_1^{(2)}, \delta_2^{(2)}]$, $\delta_1 \in [\delta_1^{(1)}, \delta_2^{(1)}]$ and $\delta_3 \in [\delta_1^{(2)}, \delta_2^{(2)}]$ such that $\delta_1 \neq \delta_2$ and $\delta_3 \neq \delta_2$. Note that a bifurcation point is included for the value $s = \frac{1}{2}$. By repeating the same argument, solution curves of the form

$$\begin{aligned}\psi &:]0,1[\to \mathbb{C}^n : s \to \psi(s),\\ \mu &:]0,1[\to \mathbb{R} : s \to \mu(s),\end{aligned} \tag{6.4}$$

are eventually constructed. Isolated bifurcation points of $\mathcal{F}$ are not ignored, but possibly belong to multiple curves.

The pseudo-arclength continuation algorithm (see section 6.4) will allow us to approximate a single curve of the form (6.4) by a finite set of points

$(\psi^{(0)}, \mu^{(0)}), (\psi^{(1)}, \mu^{(1)}), (\psi^{(2)}, \mu^{(2)}), \ldots$. The curve being approximated depends on the starting point of the algorithm.

Note that, when required, tangent directions along a solution curve (see definition 6.5) will be normalized before use in further algorithms. In bifurcation points these directions were not yet defined, it is possible for such points to contain multiple. This will be one of the topics of chapter 7.

6.3 Complications due to symmetry

In section 6.2 we described the theoretical existence and construction of solution curves, based on the implicit function theorem. In the current section we will analyse the effect of underlying symmetries on the conditions of the IFT.

Definition 6.7 (Group action [48])**.** Consider a group G. A group action φ of G on $\mathbb{C}^n$ is a function

$$\varphi : G \times \mathbb{C}^n \to \mathbb{C}^n : (g, \psi) \to \varphi(g, \psi)$$

that satisfies

- $\forall \psi \in \mathbb{C}^n : \varphi(e, \psi) = \psi$, with e the identity element of G,
- $\forall g, h \in G : \varphi(gh, \psi) = \varphi(g, \varphi(h, \psi))$.

We will use the notation $g(\psi) = \varphi(g, \psi)$.

Definition 6.8 (Group invariance of a function [47, 48])**.** A function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$ is considered invariant under the actions of a symmetry group G if

$$\forall g \in G, \psi \in \mathbb{C}^n, \mu \in \mathbb{R} : g\left(\mathcal{F}(\psi, \mu)\right) = \mathcal{F}(g(\psi), \mu).$$

Example 6.9. The Ginzburg-Landau equation (described in section 2.5) applied to a square sample, subject to a perpendicular homogeneous magnetic field, is invariant under the actions of the group $S^1 \times D_4$ (see proposition 2.3). ◇

In functions derived from a dynamical system, group invariances are typically implied by symmetries of the underlying sample. Note that symmetry is broken when an asymmetric set of discretization points is chosen. The discretization of the sample should always preserve its symmetry.
As a consequence of definition 6.8, each solution (ψ_s, μ_s) of $\mathcal{F}(\psi, \mu) = 0$ induces other solutions $(g(\psi_s), \mu_s)$ [47, 48]:

$$\forall g \in G : \mathcal{F}(g(\psi_s), \mu_s) = g\left(\mathcal{F}(\psi_s, \mu_s)\right) = 0.$$

We distinguish two different kinds of symmetry: continuous and discrete.

6.3.1 Continuous symmetry

When $\mathcal{F}$ is invariant under the actions of a finite-dimensional Lie group G, we have

$$\forall \psi, \in \mathbb{C}^n, \rho \in \mathbb{R}_0^+ : \exists g \in G, g \neq e : g(\psi) \in S_\rho(\psi),$$

with e the identity element of G. Each solution (ψ_s, μ_s) of $\mathcal{F}(\psi, \mu) = 0$ belongs to a continuous family of solutions

$$\{(g(\psi_s), \mu_s) | g \in G\}$$

in this case. The continuous nature of such families induces a null vector for the partial Jacobian $\mathcal{F}_\psi(\psi_s, \mu_s)$ [15, 92].

Example 6.10. The Ginzburg-Landau equation of example 6.9 is invariant under the actions of the S^1 symmetry group. Each solution (ψ_s, μ_s) belongs to a family of solutions of the form $\{(\theta_\eta \psi_s, \mu_s) | \theta_\eta \in S^1\}$. The vector $\mathrm{i}\psi_s$ is a null vector of $\mathcal{F}_\psi(\psi_s, \mu_s)$, induced by this continuous symmetry (see section 2.5.5). ◇

As a consequence, when $\mathcal{F}$ contains continuous symmetry, each solution (ψ_s, μ_s) of $\mathcal{F}(\psi, \mu) = 0$ is in fact a (non-isolated) bifurcation point. The standard implicit function theorem (theorem 6.3) is not applicable in this case, due to its second condition not being satisfied [26, 91]. We can however apply an alternative version of this theorem.

Theorem 6.11 (G-invariant implicit function theorem [26, 80])**.** Let $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$ be invariant under the actions of a finite-dimensional Lie group G. Let $\mathcal{F}$ and $(\psi^{(0)}, \mu^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$ satisfy:

- $\mathcal{F}(\psi^{(0)}, \mu^{(0)}) = 0$.
- The kernel of $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ consists of only the null vector induced by G.
- $\mathcal{F}$ and $\mathcal{F}_\psi$ are locally Lipschitz continuous in $S_\rho((\psi^{(0)}, \mu^{(0)}))$ for a certain $\rho \in \mathbb{R}_0^+$.

Then there exists $\delta \in \mathbb{R}$ with $0 < \delta \leq \rho$ and a function $\psi^{(i)} : S_\delta(\mu^{(0)}) \to \mathbb{C}^n$ that is continuous on $S_\delta(\mu^{(0)})$ with $\psi^{(i)}(\mu^{(0)}) = \psi^{(0)}$ such that

$$\forall \mu \in S_\delta(\mu^{(0)}) : \mathcal{F}(\psi^{(i)}(\mu), \mu) = 0.$$

The function $\psi^{(i)}$ has a continuous derivative on $S_\delta(\mu^{(0)})$ if $\mathcal{F}_\mu$ is continuous in $S_\rho((\psi^{(0)}, \mu^{(0)}))$.

Proof. See [80] □

Note that $\psi^{(i)}$ is not unique due to the continuous symmetry. We will choose this function such that

$$\forall s \in]0, 1[: \left\langle \frac{d\psi^{(i)}(\mu(s))}{d\mu}, \phi(s) \right\rangle = 0, \tag{6.5}$$

with $\phi(s)$ the null vector of $\mathcal{F}_\psi(\psi^{(i)}(s), \mu(s))$ induced by the continuous symmetry.

Given a continuous family of points

$$\{(g(\psi^{(0)}), \mu^{(0)}) | g \in G\}$$

with $\mathcal{F}(\psi^{(0)}, \mu^{(0)}) = 0$, theorem 6.11 allows the construction of solution curve families of the form $\{(g(\psi(s)), \mu(s)) \mid g \in G\}$ for certain $\psi :]0,1[\to \mathbb{C}^n : s \to \psi(s)$ and $\mu :]0,1[\to \mathbb{R} : s \to \mu(s)$. These curves are constructed analogously to section 6.2.

In theory every solution of $\mathcal{F}(\psi, \mu) = 0$ is a bifurcation point when $\mathcal{F}$ is invariant under the actions of a finite-dimensional Lie group G. The bifurcations solely induced by the continuous symmetry are however not useful when analysing physical stability or applying automatic exploration techniques (these techniques would lead to a different representative of the same solution family). For only a limited amount of the bifurcations, the null vectors of the Jacobian $\mathcal{F}_\psi(\psi, \mu)$ are not solely induced by the continuous symmetry. These points will be considered as the actual bifurcation points of $\mathcal{F}$ in the remainder of this chapter. For these bifurcations $(\psi^{(b)}, \mu^{(b)})$ definition 6.2 is satisfied, with ϕ different from the null vector induced by G. Note that these points do not satisfy the second condition of 6.11. They are added to the solution curves in an analogue manner as in section 6.2.

The tangent direction in a point $(\psi(s), \mu(s))$ of a solution curve family is again defined as $(\dot{\psi}(s), \dot{\mu}(s))$, with

$$\begin{aligned} \dot{\psi} &:]0,1[\to \mathbb{C}^n : s \to \frac{d\psi^{(i)}(\mu(s))}{d\mu}, \\ \dot{\mu} &:]0,1[\to \mathbb{R} : s \to 1, \end{aligned} \tag{6.6}$$

where $\psi^{(i)}$ was defined such that (6.5) holds. Note that these tangent directions are perpendicular to ones that lead to a representative of the same solution family.

When solution curves are approximated in practice, it is sufficient to consider a single representative for each family of solutions. Instead of approximating an entire solution curve family $\{(g(\psi(s)), \mu(s)) \mid g \in G\}$, we will ignore perturbations in the solutions caused by the symmetry group G. A finite set of points $(\psi^{(0)}, \mu^{(0)}), (\psi^{(1)}, \mu^{(1)}), (\psi^{(2)}, \mu^{(2)}), \ldots$ will be calculated, that approximate solutions of the form $(g^{(0)}(\psi(s^{(0)})), \mu(s^{(0)}))$, $(g^{(1)}(\psi(s^{(1)})), \mu(s^{(1)}))$, $(g^{(2)}(\psi(s^{(2)})), \mu(s^{(2)}))$, $\ldots$ for $s^{(0)}, s^{(1)}, s^{(2)}, \cdots \in]0,1[$ and arbitrary $g^{(0)}$, $g^{(1)}$, $g^{(2)}$, $\cdots \in G$. When bifurcation points are approximated, it will be sufficient to calculate a single representative as well.

6.3.2 Discrete symmetry

No adjustments need to be made to the implicit function theorem when $\mathcal{F}$ is invariant under the actions of a finite group G: the discrete symmetry of

the equation does not induce a null vector for the partial Jacobian $\mathcal{F}_\psi$, the conditions of the IFT hold for general solutions of $\mathcal{F}(\psi, \mu) = 0$. An example of invariance under a finite group is given by the dihedral group D_4 in example 6.9.

Discrete symmetries are important when automatic exploration techniques are considered. Given the invariances of $\mathcal{F}$, the equivariant branching lemma predicts the symmetry of solution curves that emerge at bifurcation points. The lemma and its applications are discussed in detail in section 7.5.

Similar to invariance under continuous symmetry, we will omit calculation of solution curves that belong to the same group orbit.

Definition 6.12 (Group orbit [47, 48])**.** Given a function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$, invariant under the actions of a finite group G. The group orbit of a solution curve $(\psi(s), \mu(s))$ of $\mathcal{F}$ is defined as

$$\{(g(\psi(s)), \mu) \mid g \in G\}.$$

For a given solution curve $(\psi(s), \mu(s))$, each element in its group orbit is also a solution curve of $\mathcal{F}$. In practice the calculation of curves in a same orbit is prevented by comparing outgoing tangent directions in bifurcation points. If multiple directions lead to curves in the same orbit, only a single one is used for further continuation.

To check whether two (approximate) directions $(\dot{\psi}^{(1)}, \dot{\mu}^{(1)})$ and $(\dot{\psi}^{(2)}, \dot{\mu}^{(2)})$ lead to curves in the same orbit, we check if the value

$$\min_{g \in G} \left(\left\| g(\dot{\psi}^{(1)}) - \dot{\psi}^{(2)} \right\|^2 \right) + \left(\dot{\mu}^{(1)} - \dot{\mu}^{(2)} \right)^2$$

approximates zero. Note that similar comparisons, where the group symmetry G needs to be accounted for, are used when comparing approximated (bifurcation) points of $\mathcal{F}$. These comparisons are used when creating a bifurcation diagram in practice (see section 8.2).

6.4 Numerical continuation

6.4.1 Pseudo-arclength continuation

In this section the pseudo-arclength continuation algorithm [10, 61, 62] is described. It allows the approximation of a single solution curve $(\psi(s), \mu(s))$ of $\mathcal{F}$ by a finite set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ...

Starting from an initial solution $(\psi^{(0)}, \mu^{(0)})$ of $\mathcal{F}(\psi, \mu) = 0$ and a (normalized) tangent direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$ to the wanted solution curve, the next point $(\psi^{(1)}, \mu^{(1)})$ is constructed with a predictor-corrector strategy. A prediction $(\tilde{\psi}^{(1)}, \tilde{\mu}^{(1)})$ for the next point is constructed by perturbing the given point in the direction of the tangent:

$$\begin{aligned} \tilde{\psi}^{(1)} &= \psi^{(0)} + \Delta s \dot{\psi}^{(0)}, \\ \tilde{\mu}^{(1)} &= \mu^{(0)} + \Delta s \dot{\mu}^{(0)}, \end{aligned} \tag{6.7}$$

with $\Delta s \in \mathbb{R}_0^+$ a sufficiently small step size. We will use a variable step size in practice, more details on how to choose this value are given in section 6.7. The predicted point $(\tilde{\psi}^{(1)}, \tilde{\mu}^{(1)})$ is corrected to a solution of $\mathcal{F}(\psi, \mu) = 0$ by application of the (standard or deflated) block Newton-Krylov method (see sections 5.2 and 5.3) to the nonlinear system

$$\mathcal{H}\left(\psi^{(1)}, \mu^{(1)}\right) = \begin{cases} \mathcal{F}\left(\psi^{(1)}, \mu^{(1)}\right) = 0, \\ \mathcal{G}\left(\psi^{(1)}, \mu^{(1)}\right) = 0, \end{cases} \tag{6.8}$$

with $\mathcal{G}$ the pseudo-arclength condition:

$$\mathcal{G} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R} : (\psi, \mu) \to \langle \dot{\psi}^{(0)}, \psi - \tilde{\psi}^{(1)} \rangle + \dot{\mu}^{(0)*} \left(\mu - \tilde{\mu}^{(1)}\right) = 0. \tag{6.9}$$

The partial derivatives of the linear function $\mathcal{G}$ are given by $(\forall \psi \in \mathbb{C}^n, \mu \in \mathbb{R})$

$$\mathcal{G}_\psi(\psi, \mu) : \mathbb{C}^n \to \mathbb{R} : \xi \to \langle \dot{\psi}^{(0)}, \xi \rangle, \tag{6.10}$$

$$\mathcal{G}_\mu(\psi, \mu) = \dot{\mu}^{(0)*}. \tag{6.11}$$

The solution of (6.8) yields a second point $(\psi^{(1)}, \mu^{(1)})$ of the solution curve. The steps are repeated with $(\psi^{(1)}, \mu^{(1)})$ as the given point to construct a solution $(\psi^{(2)}, \mu^{(2)})$. Further repetition eventually yields a finite set of solutions $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ...

In practice we do not calculate further tangent directions $(\dot{\psi}^{(j)}, \dot{\mu}^{(j)})$ $(j > 0)$ exactly. Instead they are approximated by

$$\forall j = 1, 2, 3, \ldots : \dot{\psi}^{(j)} = \gamma^{-1} \breve{\psi}^{(j)}, \qquad \dot{\mu}^{(j)} = \gamma^{-1} \breve{\mu}^{(j)}, \tag{6.12}$$

$$\text{with } \breve{\psi}^{(j)} = \psi^{(j)} - \psi^{(j-1)}, \quad \breve{\mu}^{(j)} = \mu^{(j)} - \mu^{(j-1)}, \quad \gamma = \sqrt{\left\| \breve{\psi}^{(j)} \right\|^2 + \breve{\mu}^{(j)2}}.$$

The initial direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$ to start the continuation should be provided. When the method is used to construct full bifurcation diagrams, this direction will either be calculated by one of the algorithms discussed in chapter 7, or approximated by $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)}) = (0, 1)$. Note that this approximation corresponds to a parameter continuation step [10, 32].

Pseudo-code for a single step of the pseudo-arclength continuation algorithm is provided by algorithm 6.2 on page 195 in appendix 6.9. A prediction is made from the given points by application of (6.7). The solution of the nonlinear system (6.8) is then approximated by the deflated block Newton method (see section 5.3). If the function $\mathcal{F}$ is invariant under a finite-dimensional Lie group (see section 6.3.1), null vectors of the partial Jacobian $\mathcal{F}_\psi$ (induced by the continuous symmetry) are deflated, preventing diverging convergence behaviour. If $\mathcal{F}_\psi$ does not contain any approximate null vectors, the deflated method is equivalent to the standard one (described in section 5.2).

Algorithm 6.2 forms an important part of algorithm 6.1 (page 193), used to construct an approximation to an entire solution curve through an initial point $(\psi^{(0)}, \mu^{(0)})$. Points of the curve are constructed until a specified amount is reached, or until certain specified conditions are satisfied. These conditions are

often problem-dependent, examples include the last calculated point approximating $(\psi^{(0)}, \mu^{(0)})$, or the value of the physical parameter leaving a certain range. In practice these are provided in algorithm 6.1 as functions $C_1, \ldots, C_l$ ($l \in \mathbb{N}$) of the form $C_i : \mathbb{C}^n \times \mathbb{R} \to \{True, False\}$ ($\forall i = 1, \ldots, l$). Except for points on the solution curve, algorithm 6.1 also approximates bifurcation points and analyses the stability along the curve. Further details of the algorithm are provided in subsequent sections.

6.4.2 False reversion

If the function $\mathcal{F}$ is invariant under the actions of a finite-dimensional Lie group G (see section 6.3.1), a further adjustment is required when the pseudo-arclength continuation algorithm is applied.

Consider points $(\psi^{(j)}, \mu^{(j)}) \in \mathbb{C}^n \times \mathbb{R}$ ($j = 0, 1, 2, \ldots$) calculated by the algorithm and denote $\phi^{(j)}$ the null vector of $\mathcal{F}_\psi(\psi^{(j)}, \mu^{(j)})$ induced by G. If approximations (6.12) are used to calculate tangent directions $(\dot{\psi}^{(j)}, \dot{\mu}^{(j)})$, the vectors $\dot{\psi}^{(j)}$ are not necessarily perpendicular to $\phi^{(j)}$.

Though a solution curve is still approximated, it is possible for false reversion to occur. This happens when the physical parameter of the problem shifts its direction (the sign of $\dot{\mu}^{(j)}$ changes) after a certain step j, without an actual bifurcation point being near. This shift of the physical parameter is entirely caused by the absence of orthogonality between $\dot{\psi}^{(j)}$ and $\phi^{(j)}$. Though this might not happen in the first steps of the pseudo-arclength continuation algorithm, flaws in the orthogonality between $\dot{\psi}^{(j)}$ and $\phi^{(j)}$ are typically transferred to further steps, eventually causing the false reversion problem. As a consequence of this problem, the approximated solution curves are possibly incomplete.

To prevent false reversion, (approximated) vectors $\dot{\psi}^{(j)}$ ($j = 0, 1, 2, \ldots$) should always be orthogonalized to any null vectors of $\mathcal{F}(\psi^{(j)}, \mu^{(j)})$ induced by G before being used to construct a prediction by (6.7).

6.5 Detection of bifurcation points

For automatic exploration techniques to be applied (see chapter 7) or physical stability to be analysed (see section 6.8), approximations to the bifurcation points of $\mathcal{F}$ are required. In the current section, we first derive a condition that indicates the proximity of bifurcations.

6.5.1 Detection by Ritz values

We use a method based on the analysis done in Mei [72] for indicating bifurcation proximity. After each pseudo-arclength continuation step j (yielding a point $(\psi^{(j)}, \mu^{(j)})$), the lowest magnitude Ritz values of the Jacobian $\mathcal{F}_\psi(\psi^{(j)}, \mu^{(j)})$ are calculated. Remember that we only consider self-adjoint Jacobians in the thesis. If the point $(\psi^{(j)}, \mu^{(j)})$ is close to a bifurcation, one of the Ritz values will approximate zero. Furthermore, if the eigenvalue associated with the bifurcation changes its sign after passing this point, the same happens

with the corresponding Ritz value. These properties give rise to the following definition:

Definition 6.13 (Near bifurcation condition)**.** Consider two consecutive points $(\psi^{(0)}, \mu^{(0)})$ and $(\psi^{(1)}, \mu^{(1)})$ of a solution curve, calculated by pseudo-arclength continuation. Let $\tilde{\lambda}_1^{(0)}, \tilde{\lambda}_2^{(0)}, \ldots, \tilde{\lambda}_k^{(0)}$ and $\tilde{\lambda}_1^{(1)}, \tilde{\lambda}_2^{(1)}, \ldots, \tilde{\lambda}_k^{(1)}$ be the k lowest magnitude Ritz values of respectively $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ and $\mathcal{F}_\psi(\psi^{(1)}, \mu^{(1)})$, ignoring values possibly induced by continuous symmetry. Assume the Ritz values are ordered according to the same eigenvectors. A bifurcation point is close to $(\psi^{(1)}, \mu^{(1)})$ if either

$$\exists j \in \{1, \ldots, k\} : |\tilde{\lambda}_j^{(1)}| < \epsilon_1 \text{ for a certain threshold } \epsilon_1 \in \mathbb{R}_0^+, \epsilon_1 \ll 1,$$

$$\exists j \in \{1, \ldots, k\} : \begin{cases} |\tilde{\lambda}_j^{(1)}| < \epsilon_2 \text{ for a certain threshold } \epsilon_2 \in \mathbb{R}_0^+, \epsilon_1 \ll \epsilon_2 \ll 1, \\ \tilde{\lambda}_j^{(0)} \tilde{\lambda}_j^{(1)} < 0. \end{cases}$$

In this case we say that the point $(\psi^{(1)}, \mu^{(1)})$ satisfies the near bifurcation condition.

In practice Ritz values are calculated by algorithm 3.3 (page 43). Any Ritz values induced by continuous symmetry invariances of $\mathcal{F}$ are removed by providing a deflation matrix with the corresponding approximate null vectors. It is typically sufficient to consider a small amount of Ritz values, for example $k = 5$, in definition 6.13. Pseudo-code for the near bifurcation condition is provided by algorithm 6.3 on page 195 in appendix 6.9.

6.5.2 Detection by test functions

A different technique for the detection of bifurcation points consists of applying a certain test function $T : \mathcal{C}^n \times \mathbb{R} \to \mathbb{R}$ after each pseudo-arclength continuation step [10, 95]. This function is constructed such that it has the property

$$(\psi, \mu) \text{ is a bifurcation point of } \mathcal{F} \iff T(\psi, \mu) = 0.$$

A bifurcation point is nearby when the test function changes its sign or approximates zero. A typical choice is the function $T = det(\mathcal{F}_\psi(\cdot, \cdot))$ [10, 95]. For our applications the determinant of the partial Jacobian $\mathcal{F}_\psi$ is however not easily calculated or approximated. We will not use test functions for the detection of bifurcation points. The near bifurcation condition defined by 6.13 will be used instead.

6.6 Approximation of bifurcation points

Points that satisfy the near bifurcation condition will be used as initial guesses for the algorithms that calculate approximations of bifurcation points. In the current section we will describe multiple methods to use for problems with a Hermitian partial Jacobian.

The choice of method should be provided as input in algorithm 6.1. It is executed after constructing the finite set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$,

$(\psi^{(2)}, \mu^{(2)}), \ldots$ that belong to the solution curve. Points that satisfied the near bifurcation condition (checked during the pseudo-arclength continuation process) are used as initial guesses.

6.6.1 Construction of an extended nonlinear system

Absence of continuous symmetry

A first method for the approximation of bifurcation points is described by [72, 95]. An extended nonlinear system of equations is constructed, such that only a bifurcation point solves the system. The bifurcation is then approximated by application of the standard Newton method (see section 4.2), combined with a block elimination technique, to this system. In absence of continuous symmetries of $\mathcal{F}$, the nonlinear system is given by [95]

$$\mathcal{H}(\psi, \phi, \mu) = \begin{cases} \mathcal{F}(\psi, \mu) = 0, \\ \mathcal{F}_\psi(\psi, \mu)\phi = 0, \\ \langle \hat{\phi}, \phi \rangle - 1 = 0. \end{cases} \tag{6.13}$$

These equations are solved for $\psi, \phi \in \mathbb{C}^n$ and $\mu \in \mathbb{R}$. The part (ψ, μ) of the solution represents the bifurcation point, ϕ a null vector of the partial Jacobian $\mathcal{F}_\psi(\psi, \mu)$. The vector $\hat{\phi} \in \mathbb{C}^n$ is a reference solution, used to prevent the trivial choice $\phi = 0$.

The bifurcation point is approximated by solving (6.13) by a Newton-Krylov algorithm. For given guesses $\tilde{\psi}$, $\tilde{\phi}$ and $\tilde{\mu}$, the linear system

$$\begin{pmatrix} \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu}) & 0 & \mathcal{F}_\mu(\tilde{\psi}, \tilde{\mu}) \\ \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi} & \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu}) & \mathcal{F}_{\psi\mu}(\tilde{\psi}, \tilde{\mu})\tilde{\phi} \\ 0 & \langle \hat{\phi}, \cdot \rangle & 0 \end{pmatrix} \begin{pmatrix} \Delta\psi \\ \Delta\phi \\ \Delta\mu \end{pmatrix} = - \begin{pmatrix} \mathcal{F}(\tilde{\psi}, \tilde{\mu}) \\ \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi} \\ \langle \hat{\phi}, \tilde{\phi} \rangle - 1 \end{pmatrix} \tag{6.14}$$

is solved for update vectors $\Delta\psi$, $\Delta\phi$ and $\Delta\mu$. We do not solve (6.14) by a direct application of a Krylov method, but use a block elimination technique. First two smaller linear systems are solved for the vectors $y^{(1)}$ and $y^{(2)}$:

$$\mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})y^{(1)} = -\mathcal{F}(\tilde{\psi}, \tilde{\mu}), \tag{6.15}$$

$$\mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})y^{(2)} = \mathcal{F}_\mu(\tilde{\psi}, \tilde{\mu}). \tag{6.16}$$

Given these solutions, $y^{(3)}$ and $y^{(4)}$ are calculated by solving two more linear systems

$$\mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})y^{(3)} = -\mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi} - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(1)}, \tag{6.17}$$

$$\mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})y^{(4)} = \mathcal{F}_{\psi\mu}(\tilde{\psi}, \tilde{\mu})\tilde{\phi} - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(2)}. \tag{6.18}$$

In practice these linear systems are solved by application of the (preconditioned) GMRES algorithm (algorithm 3.4, page 44). The update vectors are

now constructed by

$$\Delta\mu = \frac{-(\langle\hat{\phi},\tilde{\phi}\rangle - 1) - \langle\hat{\phi}, y^{(3)}\rangle}{-\langle\hat{\phi}, y^{(4)}\rangle}, \tag{6.19}$$

$$\Delta\psi = y^{(1)} - \Delta\mu y^{(2)}, \tag{6.20}$$

$$\Delta\phi = y^{(3)} - \Delta\mu y^{(4)}. \tag{6.21}$$

Algorithm 6.4 (page 196) contains pseudo-code for the bifurcation approximation method described in the current section. As an initial guess for the Newton method, a point $(\psi^{(0)}, \mu^{(0)})$ that satisfies the near bifurcation condition is used. An approximate null vector of $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ is calculated by algorithm 3.3 (page 43) as an initial guess $\phi^{(0)}$. This vector is also used for the reference solution $\hat{\phi}$. If a (Hermitian, positive-definite) preconditioner P for the linear systems is provided, an approximate null vector of $P(\psi^{(0)}, \mu^{(0)})\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ should be calculated. Each Newton iteration solves the linear system (6.14) for update vectors, used to construct the next guess.

Instead of a single GMRES application to (6.14), solutions $y^{(1)}$, $y^{(2)}$, $y^{(3)}$ and $y^{(4)}$ to four smaller sized linear systems are approximated. These vectors are then used to construct the solution of (6.14). Newton iterations are performed until the residual norm of (6.13) is sufficiently small (below a specified tolerance), or after a maximum amount of steps have been executed.

Algorithm 6.4 is applied to approximate a bifurcation point of the Liouville-Bratu-Gelfand equation (see section 2.4) in example 6.14. The point is successfully approximated.

Example 6.14. Consider the Liouville-Bratu-Gelfand equation (described in section 2.4), applied to a square domain with 900 discretization points. The point $(\psi^{(0)}, \mu^{(0)})$, with

$$\psi^{(0)} = \begin{pmatrix} 2.022 \\ 2.022 \\ \vdots \\ 2.022 \end{pmatrix}$$

and $\mu^{(0)} = 0.268$ satisfies the near bifurcation condition. We apply algorithm 6.4 to search a bifurcation point of the equation $\mathcal{F}$, defined by (2.7).The point $(\psi^{(0)}, \mu^{(0)})$, together with an approximate null vector $\phi^{(0)}$ of $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$ (calculated with algorithm 3.3, see page 43), is used as an initial guess. The inner product (2.9) is used, preconditioning is not applied.

The algorithm yields an approximate bifurcation point at $\mu \approx 0.2635$ after 3 Newton steps. A plot with the residual norms of the extended nonlinear system for each iteration is given by figure 6.1. ◇

Note that the method described in the current section is also applied when the partial Jacobian $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ contains multiple null vectors. Bifurcation points with this property still satisfy (6.13) [109, 27]. These points typically exist when the problem contains a discrete symmetry.

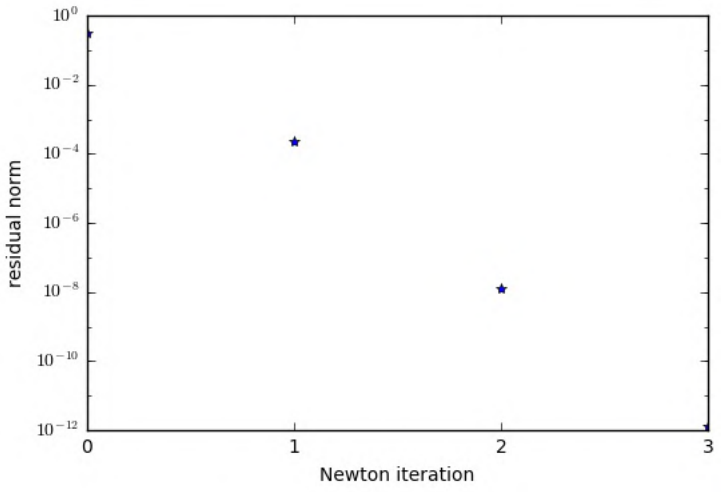


Figure 6.1: Residual plot of example 6.14. The extended nonlinear system (6.13) is solved by the Newton method to find a bifurcation point. The residual norm converges to approximately 10^{-12} after 3 iterations, which is the attainable accuracy.

Review of other extended nonlinear systems

Different nonlinear systems than (6.13) with similar properties can be used to approximate bifurcation points as well. One class of systems demands the point (ψ, μ) to be a zero of a test function T (see section 6.5) [72, 95]. The nonlinear system is given by

$$\mathcal{H}(\psi, \mu) = \begin{cases} \mathcal{F}(\psi, \mu) = 0, \\ T(\psi, \mu) = 0. \end{cases} \tag{6.22}$$

As stated in 6.5, the typical choice $T = \det(\mathcal{F}_\psi(\cdot, \cdot))$ is not practical for large-scale systems. An alternative test function is given by [95]

$$T(\psi, \mu) = e_l^T \mathcal{F}_\psi(\psi, \mu) \mathcal{F}_\psi^{(lk)}(\psi, \mu)^{-1} e_l. \tag{6.23}$$

The operator $\mathcal{F}_\psi^{(lk)}(\psi, \mu) = (I - e_l e_l^T)\mathcal{F}_\psi(\psi, \mu) + e_l e_k^T$ is the partial Jacobian $\mathcal{F}_\psi(\psi, \mu)$ where the lth row is replaced by the kth unit vector (e_k^T) for certain $k, l \in \mathbb{N}$. To find bifurcations from (6.22) with test function (6.23), linear systems with $\mathcal{F}_\psi^{(lk)}(\psi, \mu)$ need to be solved.

The extended system (6.13) is easily generalized to [95]

$$\mathcal{H}(\psi, \phi, \mu) = \begin{cases} \mathcal{F}(\psi, \mu) = 0, \\ \mathcal{F}_\psi(\psi, \mu)\phi = 0, \\ h(\phi) - 1 = 0, \end{cases} \tag{6.24}$$

where the functional h is chosen such that $\phi = 0$ is excluded from the solution space. In (6.13) the choice

$$h : \mathbb{C}^n \to \mathbb{R} : \phi \to \langle \hat{\phi}, \phi \rangle$$

was made, for a certain reference solution $\hat{\phi} \in \mathbb{C}^n$. Other functionals, that exclude $\phi = 0$ from the solution space, can however be chosen.

An extended system similar to (6.13) is given by [95]

$$\mathcal{H}(\psi, \mu, \phi, \eta) = \begin{cases} \mathcal{F}(\psi, \mu) = 0, \\ \left(I - e_l^T e_l\right) \mathcal{F}_\psi(\psi, \mu)\phi = 0, \\ h(\phi) - 1 = 0, \\ \langle e_l, \mathcal{F}_\psi(\psi, \mu)\phi \rangle - \eta = 0, \end{cases} \tag{6.25}$$

with $h(\phi)$ and e_l (for certain $l \in \mathbb{N}$) defined as before and $\eta \in \mathbb{R}$ an additional unknown. This system is larger than (6.13), requiring more computational work to solve. A similar block elimination technique as discussed for (6.13) can be applied, requiring 2 more linear system solves in each Newton iteration.

Another system that can be solved for bifurcation points is [75]

$$\mathcal{H}(\psi, \mu, \phi, \eta) = \begin{cases} \mathcal{F}(\psi, \mu) + \eta\phi = 0, \\ \mathcal{F}_\psi(\psi, \mu)\phi = 0, \\ \langle \phi, \mathcal{F}_\mu(\psi, \mu) \rangle = 0, \\ \frac{1}{2}\left(\langle \phi, \phi \rangle - 1\right) = 0, \end{cases} \tag{6.26}$$

with again $\eta \in \mathbb{R}$ an additional unknown. Just like (6.25), solving this system with the Newton method requires more computational work than solving (6.13).

A final extended system, discussed in Mei [72], is given by

$$\mathcal{H}(\psi, \mu, \phi, v) = \begin{cases} \mathcal{F}(\psi, \mu) + \langle \phi, \mathcal{F}_\psi(\psi, \mu)\phi \rangle \phi = 0, \\ \mathcal{F}_\psi(\psi, \mu)\phi + \frac{1}{2}\left(\langle \phi, \phi \rangle - 1\right)\phi = 0, \\ \mathcal{F}_\psi(\psi, \mu)v + \mathcal{F}_\mu(\psi, \mu) + \langle \phi, v \rangle \phi = 0, \\ \langle \phi, \mathcal{F}_\psi(\psi, \mu)v + \mathcal{F}_\mu(\psi, \mu) \rangle = 0, \end{cases} \tag{6.27}$$

with $v \in \mathbb{C}^n$ a vector of additional unknowns. This system can however not be solved with a block elimination technique since both the unknowns ψ and ϕ appear in the first and second equation.

Presence of continuous symmetry

When the function $\mathcal{F}$ is invariant under a finite-dimensional Lie group G, each solution of $\mathcal{F}(\psi, \mu) = 0$ is in fact a bifurcation point (see section 6.3.1) and solves (6.13). The vector ϕ used in the solution is the null vector of $\mathcal{F}_\psi(\psi, \mu)$ induced by the continuous symmetry, we denote this vector by $\breve{\phi}(\psi, \mu)$ (highlighting its dependency on ψ and μ).

In order to calculate actual bifurcation points of $\mathcal{F}$ (not induced by the continuous symmetry), an alternative nonlinear system of equations is constructed:

$$\mathcal{H}(\psi,\phi,\mu) = \begin{cases} \mathcal{F}(\psi,\mu) = 0, \\ \mathcal{F}_\psi(\psi,\mu)\phi = 0, \\ \langle \breve{\phi}(\psi,\mu), \phi\rangle = 0. \end{cases} \tag{6.28}$$

The last condition asserts that the calculated null vector ϕ will differ from $\breve{\phi}(\psi,\mu)$: the operator $\mathcal{F}_\psi(\psi,\mu)$ contains different null vectors than the one induced by continuous symmetry.

A Newton-Krylov method is again applied to solve the nonlinear system. Given guesses $\tilde{\psi}$, $\tilde{\phi}$ and $\tilde{\mu}$, the linear system

$$\begin{pmatrix} \mathcal{F}_\psi(\tilde{\psi},\tilde{\mu}) & 0 & \mathcal{F}_\mu(\tilde{\psi},\tilde{\mu}) \\ \mathcal{F}_{\psi\psi}(\tilde{\psi},\tilde{\mu}) & \mathcal{F}_\psi(\tilde{\psi},\tilde{\mu}) & \mathcal{F}_{\psi\mu}(\tilde{\psi},\tilde{\mu})\tilde{\phi} \\ \langle \breve{\phi}_\psi(\tilde{\psi},\tilde{\mu})\cdot,\tilde{\phi}\rangle & \langle \breve{\phi}(\tilde{\psi},\tilde{\mu}),\cdot\rangle & \langle \breve{\phi}_\mu(\tilde{\psi},\tilde{\mu}),\tilde{\phi}\rangle \end{pmatrix} \begin{pmatrix} \Delta\psi \\ \Delta\phi \\ \Delta\mu \end{pmatrix} = - \begin{pmatrix} \mathcal{F}(\tilde{\psi},\tilde{\mu}) \\ \mathcal{F}_\psi(\tilde{\psi},\tilde{\mu})\tilde{\phi} \\ \langle \breve{\phi}(\tilde{\psi},\tilde{\mu}),\tilde{\phi}\rangle \end{pmatrix} \tag{6.29}$$

is solved for update vectors. A similar block elimination technique as before is used to solve this system: first vectors $y^{(1)}$, $y^{(2)}$, $y^{(3)}$ and $y^{(4)}$ are calculated by solving the linear systems (6.15), (6.16), (6.17) and (6.18) with (preconditioned) GMRES. Note that we will not use deflation when solving these systems, but the right-hand sides are orthogonalized to any symmetry-induced approximate null vectors before GMRES is applied. The update vector $\Delta\mu$ is constructed from $y^{(1)}$, $y^{(2)}$, $y^{(3)}$ and $y^{(4)}$ by

$$\Delta\mu = \frac{-\langle \breve{\phi}(\tilde{\psi},\tilde{\mu}),\tilde{\phi}\rangle - \langle \breve{\phi}_\psi(\tilde{\psi},\tilde{\mu})y^{(1)},\tilde{\phi}\rangle - \langle \breve{\phi}(\tilde{\psi},\tilde{\mu}),y^{(3)}\rangle}{\langle \breve{\phi}_\mu(\tilde{\psi},\tilde{\mu}),\tilde{\phi}\rangle - \langle \breve{\phi}_\psi(\tilde{\psi},\tilde{\mu})y^{(2)},\tilde{\phi}\rangle - \langle \breve{\phi}(\tilde{\psi},\tilde{\mu}),y^{(4)}\rangle}, \tag{6.30}$$

update vectors $\Delta\psi$ and $\Delta\phi$ by application of (6.20) and (6.21). To prevent the trivial solution $\phi = 0$ from being calculated, the guess for ϕ should be normalized after each Newton iteration.

Pseudo-code for the method is given by algorithm 6.5 (page 197). We again use a point $(\psi^{(0)},\mu^{(0)})$ that satisfies the near bifurcation condition, together with an approximate null vector $\phi^{(0)}$ of $\mathcal{F}_\psi(\psi^{(0)},\mu^{(0)})$ (or, in the preconditioned case, $P(\psi^{(0)},\mu^{(0)})\mathcal{F}_\psi(\psi^{(0)},\mu^{(0)})$), as an initial guess. The vector $\phi^{(0)}$ should be chosen perpendicular to any null vectors induced by continuous symmetry.

The algorithm is applied to approximate bifurcation points for two instances of the Ginzburg-Landau equation (see section 2.5) in examples 6.15 and 6.16. For these examples, the function $\breve{\phi}$ is explicitly given by (see (2.23))

$$\breve{\phi} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi,\mu) \to \mathrm{i}\psi,$$

with derivatives

$$\begin{aligned} &\breve{\phi}_\psi(\psi,\mu) : \mathbb{C}^n \to \mathbb{C}^n : \phi \to \mathrm{i}\phi, \\ &\breve{\phi}_\mu(\psi,\mu) = 0. \end{aligned}$$

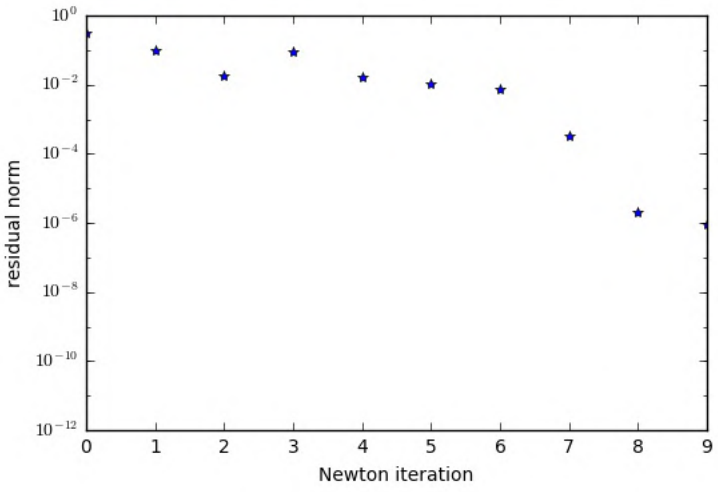


Figure 6.2: Residual plot of example 6.15. The extended nonlinear system (6.28) is solved by the Newton method to find a bifurcation point. The residual norm converges to approximately 10^{-12} after 9 iterations, which is the attainable accuracy.

Example 6.15. The Ginzburg-Landau equation (see section 2.5) is considered, applied to a square-shaped material with $n = 20737$ discretization points. We use a solution $\psi^{(0)}$ at $\mu^{(0)} = 1.573$ (lying on branch A in figure 9.9 (see section 9.5.1)) as an initial guess for a bifurcation point of (2.20). This point satisfies the near bifurcation condition.

Algorithm 6.5 is applied, again using algorithm 3.3 (page 43) to create an initial guess $\phi^{(0)}$ for the null vector. The inner product (2.21) is used, and the preconditioner (2.25) is applied. After 9 iterations an approximate bifurcation at $\mu \approx 1.6468$ is found (up to a tolerance of 10^{-6}). The residual plot of the problem is given by figure 6.2. ◇

Example 6.16. We again consider the Ginzburg-Landau equation, this time applied to a pentagon-shaped material with $n = 10401$ discretization points. A solution $\psi^{(0)}$ at $\mu^{(0)} = 1.0588$ (lying on branch B in figure 9.23 (see section 9.5.3)) is used as an initial guess for a bifurcation. The guess $\phi^{(0)}$ is constructed by application of algorithm 3.3, and the same inner product and preconditioner as for example 6.15 are used.

Algorithm 6.5 fails to find a decent approximation of the nearby bifurcation point within 10 Newton steps. Most of the iterations in figure 6.3, containing a residual plot of the problem, show an increase in residual norm. ◇

Though for example 6.15 the residual norm converges (up to the desired tolerance), this is not the case for example 6.16. This is caused by the approximate singularity of $\mathcal{F}_\psi(\psi, \mu)$ near the bifurcation, in a similar way as described in section 4.2 for the standard Newton method.

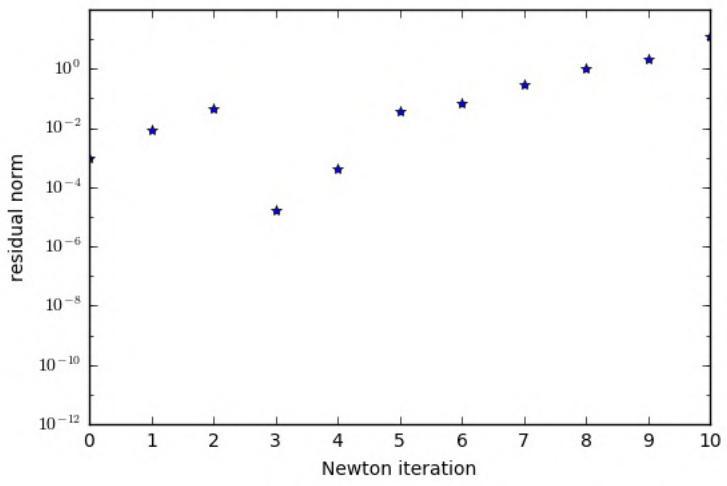


Figure 6.3: Residual plot of example 6.16. The extended nonlinear system (6.28) is solved by the Newton method to find a bifurcation point. Most iterations show an increase in residual norm, there is no convergence to a solution.

Note that the alternative extended nonlinear systems (6.25) and (6.26) can be adapted in a similar way as (6.13) to (6.28) by changing one of the equations to $\langle \breve{\phi}(\psi, \mu), \phi \rangle = 0$.

6.6.2 Newton step length adaptation

An alternative method is not based on constructing extended nonlinear systems, but on normal pseudo-arclength continuation. Consider a solution curve $(\psi(s), \mu(s))$ of $\mathcal{F}$, with $(\psi^{(b)}, \mu^{(b)}) = (\psi(s^{(b)}), \mu(s^{(b)}))$ a bifurcation point of the equation (with $s^{(b)} \in]0, 1[$). If $\mathcal{F}$ is invariant under a continuous symmetry, the considered bifurcation should not be induced by this.

We denote $\lambda(s)$ the lowest magnitude eigenvalue of $\mathcal{F}_\psi(\psi(s), \mu(s))$ (again ignoring the one induced by any continuous symmetries). We have $\lambda(s^{(b)}) = 0$, and $\lambda(s) \neq 0$ for $s \in S_\rho(s^{(b)})$, for a certain $\rho \in \mathbb{R}_0^+$ sufficiently small. Given an initial approximation $(\psi(s^{(0)}), \mu(s^{(0)}))$, with $s^{(0)} \approx s^{(b)}$, of the bifurcation, the Newton method predicts a better approximation $(\psi(s^{(1)}), \mu(s^{(1)}))$ by calculating [3]

$$s^{(1)} = s^{(0)} - \frac{\lambda(s^{(0)})}{\lambda'(s^{(0)})}. \tag{6.31}$$

In practice we do not have access to the full solution curve $(\psi(s), \mu(s))$, but only a finite set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ... Consider two consecutive points $(\psi^{(-1)}, \mu^{(-1)})$ and $(\psi^{(0)}, \mu^{(0)})$ such that $(\psi^{(0)}, \mu^{(0)})$ satisfies the near bifurcation condition, and denote $\tilde{\lambda}^{(-1)}, \tilde{\lambda}^{(0)} \in \mathbb{R}$ the lowest magnitude Ritz values of respectively $\mathcal{F}_\psi(\psi^{(-1)}, \mu^{(-1)})$ and $\mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$, calculated

by algorithm 3.3 (again ignoring the values induced by possible continuous symmetries). We have $\tilde{\lambda}^{(0)} \approx 0$.

Denoting $s^{(0)}$ such that $(\psi(s^{(0)}), \mu(s^{(0)})) = (\psi^{(0)}, \mu^{(0)})$, a first-order Taylor expansion of $(\psi(s^{(1)}), \mu(s^{(1)}))$, with $s^{(1)}$ defined by (6.31), yields an approximation [3]

$$\begin{aligned} \tilde{\psi}^{(1)} &= \psi^{(0)} + \Delta s \dot{\psi}^{(0)}, \\ \tilde{\mu}^{(1)} &= \mu^{(0)} + \Delta s \dot{\mu}^{(0)}, \end{aligned} \tag{6.32}$$

for $(\psi(s^{(1)}), \mu(s^{(1)}))$, with $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$ the (normalized) tangent direction in $(\psi^{(0)}, \mu^{(0)})$ and Δs defined by

$$\Delta s = -\frac{\lambda(s^{(0)})}{\lambda'(s^{(0)})}.$$

By the definition of $s^{(0)}$, we have $\tilde{\lambda}^{(0)} \approx \lambda(s^{(0)})$. We will not calculate the derivative $\lambda'(s^{(0)})$ explicitly, but approximate it by the finite difference formula

$$\lambda'(s^{(0)}) \approx \frac{\tilde{\lambda}^{(0)} - \tilde{\lambda}^{(-1)}}{\gamma},$$

with γ the distance between points $(\psi^{(-1)}, \mu^{(-1)})$ and $(\psi^{(0)}, \mu^{(0)})$, explicitly given by

$$\gamma = \sqrt{\left\|\psi^{(0)} - \psi^{(-1)}\right\|^2 + \left(\mu^{(0)} - \mu^{(-1)}\right)^2}.$$

The approximations of $\lambda(s^{(0)})$ and $\lambda'(s^{(0)})$ yield the choice [3]

$$\Delta s = -\frac{\tilde{\lambda}^{(0)}\gamma}{\tilde{\lambda}^{(0)} - \tilde{\lambda}^{(-1)}} \tag{6.33}$$

in (6.32). Note the similarity between (6.32) and (6.7). Since $(\tilde{\psi}^{(1)}, \tilde{\mu}^{(1)})$ does not necessarily solve $\mathcal{F}(\psi, \mu) = 0$, it is corrected by solving the nonlinear system (6.8) with a block Newton-Krylov method (see chapter 5). The solution $(\psi^{(1)}, \mu^{(1)})$ of this system approximates $(\psi(s^{(1)}), \mu(s^{(1)}))$, it should be a better guess for the bifurcation point than the initial guess $(\psi^{(0)}, \mu^{(0)})$.

The process of constructing a new guess by above method is repeated, and gives rise to the Newton step length adaptation (NSA) algorithm [3]. Eventually an approximation $(\psi^{(j)}, \mu^{(j)})$ (for certain $j \in \mathbb{N}$) that lies sufficiently close to the bifurcation point should be obtained. To check this proximity, the value $\lambda^{(j)}$ should be monitored throughout the algorithm: if $|\lambda^{(j)}|$ becomes sufficiently small (compared to a specified tolerance), the algorithm is stopped. Pseudocode for the Newton step length adaptation method is given by algorithm 6.6 on page 198.

Note that the nonlinear systems (6.8), used to assert the next guess to be a solution of $\mathcal{F}(\psi, \mu)$, are typically ill-posed. If a guess $(\psi^{(j)}, \mu^{(j)})$ lies too close to the bifurcation point (which is the goal of the algorithm), the partial Jacobian $\mathcal{F}_\psi(\psi^{(j)}, \mu^{(j)})$ becomes ill-conditioned. This linear operator is part

of the Jacobian of (6.8), required when applying the Newton method to solve this nonlinear system of equations. The closer the guess lies to the bifurcation, the worse this problem becomes. To prevent diverging convergence behaviour when solving (6.8), the split block Newton-Krylov method with mixed terms (SBNM) (described in section 5.8, see algorithm 5.7 on page 164) is applied.

If no convergence is achieved by this method within the amount of permitted iterations, or if the constructed guess lies further away from the bifurcation point (indicated by a higher lowest magnitude Ritz value), the NSA step is reset. The value for Δs is halved, a new guess is constructed by (6.32) and again corrected by algorithm 5.7. This backtracking procedure is repeated until the NSA step yields a better guess for the bifurcation, that lies on the solution curve (the Newton method converged).

The method is executed to the three examples of section 6.6.1 in examples 6.17, 6.18 and 6.19. The bifurcation point is successfully approximated in each example.

Example 6.17. Consider the same set-up as for example 6.14. The points $(\psi^{(-1)}, \mu^{(-1)})$ and $(\psi^{(0)}, \mu^{(0)})$, given by

$$\psi^{(-1)} = \begin{pmatrix} 2.022 \\ 2.022 \\ \vdots \\ 2.022 \end{pmatrix}, \qquad \psi^{(0)} = \begin{pmatrix} 2.056 \\ 2.056 \\ \vdots \\ 2.056 \end{pmatrix},$$

$\mu^{(-1)} = 0.268$ and $\mu^{(0)} = 0.263$ are used as initial guesses for the bifurcation point. Algorithm 6.6 is applied, the lowest magnitude of the Ritz values is shown for each NSA step in figure 6.4. After 3 steps, an approximation of the bifurcation is found with a Ritz value of magnitude lower than 10^{-14}. ◇

Example 6.18. The same set-up as for example 6.15 is considered. As initial guesses two solutions $(\psi^{(-1)}, \mu^{(-1)})$ and $(\psi^{(0)}, \mu^{(0)})$, with $\mu^{(-1)} = 1.573$ and $\mu^{(0)} = 1.564$ are used (both lying on branch A in figure 9.9 (see section 9.5.1)).

We apply the NSA method to search a nearby bifurcation point, this yields figure 6.5. An approximation with a Ritz value of magnitude lower than 10^{-13} is found after 4 steps. ◇

Example 6.19. We finally apply the NSA method to the set-up of example 6.16. For this example, the construction and solving of an extended nonlinear system did not yield a decent approximation to the bifurcation point. The NSA method is applied to two points $(\psi^{(-1)}, \mu^{(-1)})$ and $(\psi^{(0)}, \mu^{(0)})$, with $\mu^{(-1)} = 1.0588$ and $\mu^{(0)} = 1.0590$ (both lying on branch B in figure 9.23 (see section 9.5.3)).

Figure 6.6 shows the lowest magnitude of the Ritz values for each NSA step. After 8 iterations this value approximates 10^{-13}. Contrary to algorithm 6.5, the NSA method finds a decent approximation to the bifurcation point. ◇

Compared to the methods described in section 6.6.1, the Newton step length adaptation method appears to be more robust: the bifurcation point of example

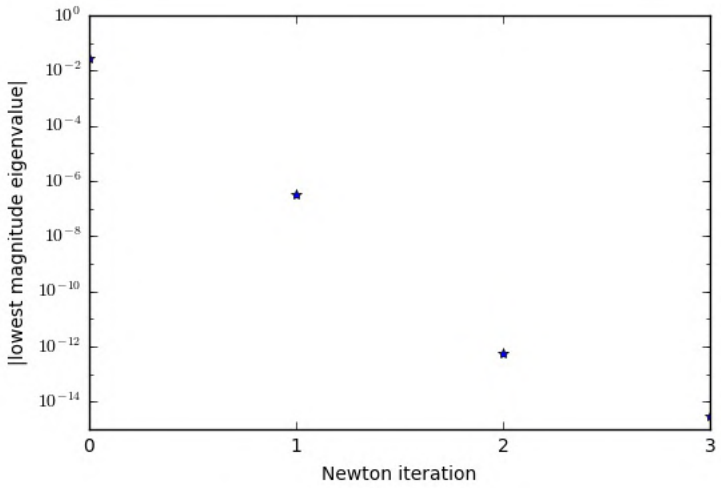


Figure 6.4: Lowest magnitude of the Ritz values for each NSA step of example 6.17, in which the NSA method is applied to find a bifurcation point. The magnitude converges to approximately 10^{-14} after 3 iterations.

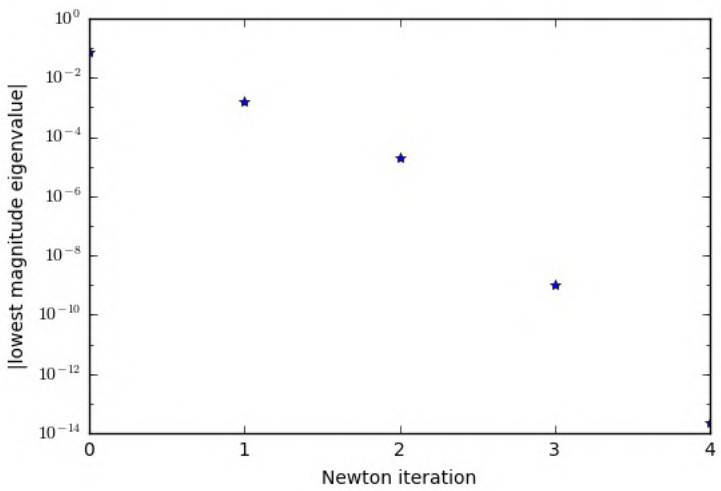


Figure 6.5: Lowest magnitude of the Ritz values for each NSA step of example 6.18, in which the NSA method is applied to find a bifurcation point. The magnitude converges to approximately 10^{-13} after 4 iterations.

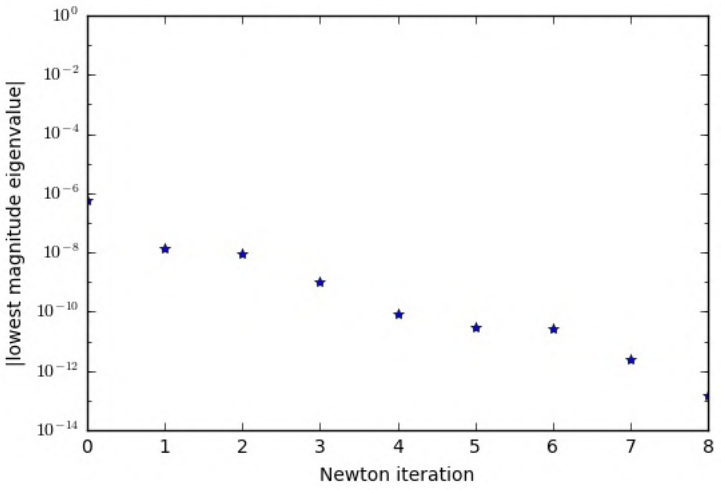


Figure 6.6: Lowest magnitude of the Ritz values for each NSA step of example 6.19, in which the NSA method is applied to find a bifurcation point. The magnitude converges to approximately 10^{-13} after 8 iterations. Contrary to application of the extended nonlinear system (6.28) (see figure 6.3), the bifurcation point is successfully approximated.

6.19 could not be approximated by solving an extended nonlinear system (see example 6.16). Typically more computational time is required though, since multiple systems of the form (6.8) have to be solved. If more than 2 Newton step length adaptation iterations are required, solving a single extended nonlinear system should be computationally more efficient.

6.7 Choice of the step size in pseudo-arclength continuation

In the predictor step of the pseudo-arclength continuation algorithm (see (6.7)), a sufficiently small step size $\Delta s \in \mathbb{R}_0^+$ needs to be chosen. We had not yet established this choice. For the first iteration of the algorithm, the step size value should be provided as input. For subsequent iterations Δs will be calculated from information gathered from the construction of the previous point.

Let $(\psi^{(j)}, \mu^{(j)})$ $(j \geq 1)$ be a point calculated by pseudo-arclength continuation. We want the step size Δs, used to construct a prediction $(\tilde{\psi}^{(j+1)}, \tilde{\mu}^{(j+1)})$ for the next point, to be sufficiently small such that two conditions are satisfied.

As a first condition we do not want the block Newton method to require a high amount of iterations in order to converge (up to a given tolerance). This amount m is approximated by the number $\tilde{m}$ of Newton steps that was required to calculate $(\psi^{(j)}, \mu^{(j)})$. If $\tilde{m}$ is sufficiently low, the condition should be satisfied by choosing Δs equal to the previously used step size. In practice, to speed up the algorithm, it will be chosen slightly higher [36]. If $\tilde{m}$ surpasses a certain

threshold, the step size is decreased [3]. In the extreme case where convergence up to the desired tolerance was not achieved, the step size is halved. We will also recalculate $(\psi^{(j)}, \mu^{(j)})$, using half of the step size, when this happens [36].

The second condition asserts that no bifurcation points are skipped during the continuation. We want to find at least one point that satisfies the near bifurcation condition for each bifurcation of $\mathcal{F}$. For this purpose the lowest magnitude Ritz value $\tilde{\lambda}^{(j)}$ of $\mathcal{F}_\psi(\psi^{(j)}, \mu^{(j)})$ (ignoring any values induced by possible continuous symmetries) is monitored. If $|\tilde{\lambda}^{(j)}|$ is lower than a specified threshold, the step size Δs should be decreased accordingly.

Algorithm 6.7 (page 199) checks the two conditions established in the current section, and uses them to update a given step size. It is used in algorithm 6.1 (page 193), before the construction of a predictor for the next point.

6.8 Physical stability

We end the chapter with an application of bifurcation points: the determination of physical stability along a solution curve.

Definition 6.20 (Linear stability [48, 95])**.** We call a zero $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$ of a function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$ linearly stable if all of the eigenvalues λ of $\mathcal{F}_\psi(\psi, \mu)$, ignoring the ones induced by possible continuous symmetries, have a positive real part ($\mathfrak{Re}(\lambda) > 0$).

Linearly unstable solutions of $\mathcal{F}(\psi, \mu) = 0$ correspond to steady states that do not persist under small perturbations. These states are usually not physically realizable [48].

Note that, since we only consider problems with a Hermitian partial Jacobian $\mathcal{F}_\psi$ in the thesis, definition 6.20 simplifies: a point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$ is linearly stable if all of the eigenvalues of $\mathcal{F}_\psi(\psi, \mu)$ are positive (ignoring the ones induced by possible continuous symmetries).

To determine the stability of a single solution (ψ_s, μ_s) of $\mathcal{F}(\psi, \mu) = 0$ in practice, we check the k lowest magnitude Ritz values $\tilde{\lambda}_1$, $\tilde{\lambda}_2$, …, $\tilde{\lambda}_k$ of $\mathcal{F}_\psi(\psi_s, \mu_s)$ (ignoring any induced by continuous symmetries). Note that these values are also calculated to check the near bifurcation condition. If one of the Ritz values is negative, (ψ_s, μ_s) is said to be linearly unstable.

When an approximation $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, … of a solution curve $(\psi(s), \mu(s))$ is considered, we will not determine the stability of the points $(\psi^{(j)}, \mu^{(j)})$ $(j = 0, 1, 2, \dots)$ individually. Individual calculation is prone to mistakes due to e.g. negative Ritz values leaving the set of the k lowest magnitude ones.

Instead, we note that points that mark transitions in linear stability are always bifurcations of $\mathcal{F}$, this is a consequence of the definition. Given the bifurcation points of $(\psi(s), \mu(s))$, we first divide the set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$,$(\psi^{(2)}, \mu^{(2)})$, … in subsets of the form $(\psi^{(j)}, \mu^{(j)})$,…,$(\psi^{(j+l)}, \mu^{(j+l)})$ (for certain $j, l \in \mathbb{N}$), chosen such that a bifurcation point is present between each last and first point of two consecutive subsets. Linear stability remains constant for points belonging to the same set.

To determine the stability of points in a single subset, the percentage of solutions that yield a negative Ritz value for $\mathcal{F}_\psi$ is calculated. If this percentage surpasses a certain threshold (e.g. 10%), all of the points of the subset are said to be linearly unstable.

Algorithm 6.8 (page 200) contains pseudo-code that determines the stability of a set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ..., calculated by the pseudo-arclength continuation algorithm. It uses approximate bifurcation points to split the set in subsets, and determines the stability of each such subset by the rule described in the current section. It is used in algorithm 6.1 (page 193), where it is executed after the bifurcation points of the solution curve have been approximated.

6.9 Appendix

Main algorithm

Algorithm 6.1 CurveCalculation

Input $m_P \in \mathbb{N}$, initial step size $\Delta s \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, choice $FindBif$ of bifurcation search algorithm, list $C = [C_1, \ldots, C_l]$ of condition functions, inner product $\langle \cdot, \cdot \rangle$, initial solution $\psi^{(0)} \in \mathbb{C}^n$, $\mu^{(0)} \in \mathbb{R}$, initial direction $\dot{\psi}^{(0)} \in \mathbb{C}^n$, $\dot{\mu}^{(0)} \in \mathbb{R}$

Output Matrices P, B and S, respectively containing points of the solution curve, bifurcations, and stability results of $\mathcal{F}$ as column vectors.

1: Set $\mathcal{P} = \mathcal{I}$ if not specified
2: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
3: Set $C = [\,]$ an empty list if not specified
4: $n_P = 0$
5: $n_D = 0$
6: Create deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 null vectors, induced by continuous symmetry
7: Calculate $(L^{(0)}, W^{(0)}, U^{(0)})$ by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(0)}, \mu^{(0)})$, $\mathcal{P} = P(\psi^{(0)}, \mu^{(0)})$ and given $\langle \cdot, \cdot \rangle$ and K
8: $L^{(-1)} = L^{(0)}$
9: $W^{(-1)} = W^{(0)}$
10: Calculate *detected* by executing CheckDetection (algorithm 6.3) with $L = L^{(-1)}$, $L' = L^{(0)}$, $W = W^{(-1)}$ and $W' = W^{(0)}$
11: **if** *detected* **then**
12: $n_D \leftarrow n_D + 1$
13: $D = \begin{pmatrix} 0 \end{pmatrix}$
14: **else**
15: Initialize D as an empty 1×0 matrix
16: **end if**
17: **for** $j = 1, \ldots, m_P$ **do**
18: Calculate $(\psi^{(j)}, \mu^{(j)}, \dot{\psi}^{(j)}, \dot{\mu}^{(j)}, \tilde{m})$ by executing PseudoArc (algorithm 6.2) with $(\psi, \mu) = (\psi^{(j-1)}, \mu^{(j-1)})$, $(\dot{\psi}, \dot{\mu}) = (\dot{\psi}^{(j-1)}, \dot{\mu}^{(j-1)})$ and given Δs, $\mathcal{F}$, P, $\langle \cdot, \cdot \rangle$ and K
19: **if** The Newton method in PseudoArc did not converge **then**
20: $\Delta s \leftarrow \frac{1}{2} \Delta s$
21: Go back to line 18
22: **end if**
23: $n_P \leftarrow n_P + 1$
24: Create deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 null vectors, induced by continuous symmetry
25: Calculate $(L^{(j)}, W^{(j)}, U^{(j)})$ by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(j)}, \mu^{(j)})$, $\mathcal{P} = P(\psi^{(j)}, \mu^{(j)})$ and given $\langle \cdot, \cdot \rangle$ and K
26: Calculate *detected* by executing CheckDetection (algorithm 6.3) with $L = L^{(j-1)}$, $L' = L^{(j)}$, $W = W^{(j-1)}$ and $W' = W^{(j)}$

if *detected* **then**
  $n_D \leftarrow n_D + 1$
  $D = \begin{pmatrix} D & j \end{pmatrix}$
end if
Calculate Δs by executing AdaptStep (algorithm 6.7) with $m = \tilde{m}$, $\lambda = L_{11}^{(j)}$ and given Δs
for C_i in C **do**
  if $C_i(\psi^{(j)}, \mu^{(j)})$ **then**
    Break
  end if
end for
end for
$P = \begin{pmatrix} \psi^{(0)} & \dots & \psi^{(n_P)} \\ \mu^{(0)} & \dots & \mu^{(n_P)} \end{pmatrix}$
$n_B = 0$
for $j = 1, \dots, n_D$ **do**
  $p = D_j$
  $new = True$
Check if any already found bifurcation points approximate $(\psi^{(p)}, \mu^{(p)})$. Set *new* to $False$ if this is the case.
  if *new* **then**
Set $p' = p - 1$ if $p > 0$, $p' = p + 1$ otherwise.
    if $FindBif$ is the algorithm FindBifNSA **then**
Calculate $(\hat{\psi}^{(n_B)}, \hat{\mu}^{(n_B)})$ by executing FindBifNSA (algorithm 6.6) with $(\psi, \mu) = (\psi^{(p')}, \mu^{(p')})$, $(\psi', \mu') = (\psi^{(p)}, \mu^{(p)})$ and given $\mathcal{F}$, $\mathcal{P}$ and $\langle \cdot, \cdot \rangle$
    else
      if The system does not contain a continuous symmetry **then**
Calculate $(\hat{\psi}^{(n_B)}, \hat{\mu}^{(n_B)})$ by executing FindBifExt1 (algorithm 6.4) with $\tilde{\psi} = \psi^{(p)}$, $\tilde{\phi} = W_1^{(p)}$, $\tilde{\mu} = \mu^{(p)}$ and given $\mathcal{F}$, $\mathcal{P}$ and $\langle \cdot, \cdot \rangle$
      else
Calculate $(\hat{\psi}^{(n_B)}, \hat{\mu}^{(n_B)})$ by executing FindBifExt2 (algorithm 6.5) with $\tilde{\psi} = \psi^{(p)}$, $\tilde{\phi} = W_1^{(p)}$, $\tilde{\mu} = \mu^{(p)}$ and given $\mathcal{F}$, $\mathcal{P}$ and $\langle \cdot, \cdot \rangle$
      end if
    end if
    if FindBif did not converge **then**
      Delete $(\hat{\psi}^{(n_B)}, \hat{\mu}^{(n_B)})$
    else
      $n_B < -n_B + 1$
    end if
  end if
end for
$B = \begin{pmatrix} \hat{\psi}^{(0)} & \dots & \hat{\psi}^{(n_B)} \\ \hat{\mu}^{(0)} & \dots & \hat{\mu}^{(n_B)} \end{pmatrix}$
Initialize L as an empty $k_{eig} \times 0$ matrix
for $j = 1, \dots, n_P$ **do**
  $L_{vec}^{(j)} = \begin{pmatrix} L_{11}^{(j)} & L_{22}^{(j)} & \dots & L_{k_{eig}k_{eig}}^{(j)} \end{pmatrix}^T$

66: $L = \begin{pmatrix} L & L_{vec}^{(j)} \end{pmatrix}$
67: **end for**
68: Calculate S by executing StabAnalysis (algorithm 6.8) with given P, B and L
69: Return P, B, S

Single pseudo-arclength continuation step

Algorithm 6.2 PseudoArc

Input step size $\Delta s \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, deflation matrix $K \in \mathbb{C}^{n \times l_1}$, solution $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$, direction $(\dot{\psi}, \dot{\mu}) \in \mathbb{C}^n \times \mathbb{R}$

Output Approximations (ψ', μ') of $\mathcal{F}(\psi, \mu) = 0$, and $(\dot{\psi}', \dot{\mu}')$ of tangent direction, number of used Newton iterations m.

1: $\tilde{\psi}' = \psi + \Delta s \dot{\psi}$
2: $\tilde{\mu}' = \mu + \Delta s \dot{\mu}$
3: Define $\mathcal{G}$ by (6.9)
4: Define $\mathcal{G}_\psi$ by (6.10)
5: Define $\mathcal{G}_\mu$ by (6.11)
6: Calculate (ψ', μ') by executing NewtonBlockDeflated (algorithm 5.2) with $F = \mathcal{F}$, $G = \mathcal{G}$, $x^{(0)} = \tilde{\psi}'$, $p^{(0)} = \tilde{\mu}'$ and given P and $\langle \cdot, \cdot \rangle$
7: Denote m the number of used Newton iterations in NewtonBlockDeflated
8: $\breve{\psi} = \psi' - \psi$
9: $\breve{\mu} = \mu' - \mu$
10: Orthogonalize $\breve{\psi}$ to the column vectors of K
11: $\gamma = \sqrt{\|\breve{\psi}\|^2 + \breve{\mu}^2}$
12: $\dot{\psi} = \gamma^{-1} \breve{\psi}$
13: $\dot{\mu} = \gamma^{-1} \breve{\mu}$
14: Return ψ', μ', $\dot{\psi}$, $\dot{\mu}$, m

Detection of bifurcation points

Algorithm 6.3 CheckDetection

Input Diagonal matrices with Ritz values $L, L' \in \mathbb{R}^{k \times k}$, matrices with eigenvectors $W, W' \in \mathbb{C}^{n \times k}$ tolerances $\epsilon_{det1}, \epsilon_{det2} \in \mathbb{R}_0^+$

Output Boolean *detected* that determines if the provided matrices indicate the near bifurcation condition.

1: $detected = False$

2: **for** $j = 1, \dots, k$ **do**
3: **if** $|L'_{jj}| < \epsilon_{det1}$ **then**
4: $detected = True$
5: Break
6: **end if**
7: **end for**
8: **if** not *detected* and $L \neq L'$ **then**
9: Orden values in L and L' according to similar eigenvectors (using W and W')
10: Denote k_{sim} the number of similar eigenvectors
11: **for** $j = 1, \dots, k_{sim}$ **do**
12: **if** $|L_{jj}L'_{jj}| < \epsilon_{det2}$ **then**
13: $detected = True$
14: Break
15: **end if**
16: **end for**
17: **end if**
18: Return *detected*

Approximation of bifurcation points

By solving an extended system (no continuous symmetry)

Algorithm 6.4 FindBifExt1

Input $m_{ext} \in \mathbb{N}$, tolerances $\epsilon_{ext1}, \epsilon_{ext2} \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives $\mathcal{F}_\psi$, $\mathcal{F}_\mu$, $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, inner product $\langle \cdot, \cdot \rangle$, initial guesses $\tilde{\psi}, \tilde{\phi} \in \mathbb{C}^n$, $\tilde{\mu} \in \mathbb{R}$

Output Approximations $\tilde{\psi}$, $\tilde{\mu}$ for a bifurcation ponit of $\mathcal{F}$.

1: Define $\mathcal{F}_\psi$, $\mathcal{F}_\mu$, $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$ by (5.2), (5.4), (5.6) and (5.7) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: $\hat{\phi} = \tilde{\phi}$
5: $r_{\mathcal{F}} = \mathcal{F}(\tilde{\psi}, \tilde{\mu})$
6: $r_\phi = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi}$
7: $r_{ref} = \langle \hat{\phi}, \tilde{\phi} \rangle - 1$
8: $i = 0$
9: **while** $i < m_{ext}$ and $\sqrt{\|r_{\mathcal{F}}\|^2_{P(\tilde{x},\tilde{p})} + r^2_{ref}} > \epsilon_{ext1}$ and $\|r_\phi\|_{P(\tilde{x},\tilde{p})} > \epsilon_{ext2}$ **do**
10: $i \leftarrow i + 1$
11: Calculate $y^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = -r_{\mathcal{F}}$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$
12: Calculate $y^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = \mathcal{F}_\mu$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$

13: $b_1 = -r_\phi - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(1)}$
14: Calculate $y^{(3)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = b_1$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$
15: $b_2 = \mathcal{F}_{\psi\mu}(\tilde{\psi}, \tilde{\mu})\tilde{\phi} - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(2)}$
16: Calculate $y^{(4)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = b_2$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$
17: Calculate $\Delta\mu$ by (6.19)
18: $\Delta\psi = y^{(1)} - \Delta\mu y^{(2)}$
19: $\Delta\phi = y^{(3)} - \Delta\mu y^{(4)}$
20: $\tilde{\psi} \leftarrow \tilde{\psi} + \Delta\psi$
21: $\tilde{\mu} \leftarrow \tilde{\mu} + \Delta\mu$
22: $\tilde{\phi} \leftarrow \tilde{\phi} + \Delta\phi$
23: $r_{\mathcal{F}} = \mathcal{F}(\tilde{\psi}, \tilde{\mu})$
24: $r_\phi = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi}$
25: $r_{ref} = \langle \hat{\phi}, \tilde{\phi} \rangle - 1$
26: **end while**
27: Return $\tilde{\psi}, \tilde{\mu}$

By solving an extended system (continuous symmetry)

Algorithm 6.5 FindBifExt2

Input $m_{ext} \in \mathbb{N}$, tolerances $\epsilon_{ext1}, \epsilon_{ext2} \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives $\mathcal{F}_\psi$, $\mathcal{F}_\mu$, $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, inner product $\langle \cdot, \cdot \rangle$, initial guesses $\tilde{\psi}, \tilde{\phi} \in \mathbb{C}^n$, $\tilde{\mu} \in \mathbb{R}$

Output Approximations $\tilde{\psi}$, $\tilde{\mu}$ for a bifurcation ponit of $\mathcal{F}$.

1: Define $\mathcal{F}_\psi$, $\mathcal{F}_\mu$, $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$ by (5.2), (5.4), (5.6) and (5.7) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: Define the function $\breve{\phi} : \mathbb{C}^n \times \mathbb{R}$
5: $r_{\mathcal{F}} = \mathcal{F}(\tilde{\psi}, \tilde{\mu})$
6: $r_\phi = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi}$
7: $r_{ref} = \langle \breve{\phi}(\tilde{\psi}, \tilde{\mu}), \tilde{\phi} \rangle$
8: $i = 0$
9: **while** $i < m_{ext}$ and $\sqrt{\|r_{\mathcal{F}}\|^2_{P(\tilde{x},\tilde{p})} + r^2_{ref}} > \epsilon_{ext1}$ and $\|r_\phi\|_{P(\tilde{x},\tilde{p})} > \epsilon_{ext2}$ **do**
10: $i \leftarrow i + 1$
11: Orthogonalize $r_{\mathcal{F}}$ to $\breve{\phi}(\tilde{\psi}, \tilde{\mu})$
12: Calculate $y^{(1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = -r_{\mathcal{F}}$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$
13: $b_0 = \mathcal{F}_\mu$
14: Orthogonalize b_0 to $\breve{\phi}(\tilde{\psi}, \tilde{\mu})$
15: Calculate $y^{(2)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = b_0$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle \cdot, \cdot \rangle$

16: $b_1 = -r_\phi - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(1)}$
17: Orthogonalize b_1 to $\breve{\phi}(\tilde{\psi}, \tilde{\mu})$
18: Calculate $y^{(3)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = b_1$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle\cdot,\cdot\rangle$
19: $b_2 = \mathcal{F}_{\psi\mu}(\tilde{\psi}, \tilde{\mu})\tilde{\phi} - \mathcal{F}_{\psi\psi}(\tilde{\psi}, \tilde{\mu})\tilde{\phi}y^{(2)}$
20: Orthogonalize b_2 to $\breve{\phi}(\tilde{\psi}, \tilde{\mu})$
21: Calculate $y^{(4)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $b = b_2$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle\cdot,\cdot\rangle$
22: Calculate $\Delta\mu$ by (6.30)
23: $\Delta\psi = y^{(1)} - \Delta\mu y^{(2)}$
24: $\Delta\phi = y^{(3)} - \Delta\mu y^{(4)}$
25: $\tilde{\psi} \leftarrow \tilde{\psi} + \Delta\psi$
26: $\tilde{\mu} \leftarrow \tilde{\mu} + \Delta\mu$
27: $\tilde{\phi} \leftarrow \tilde{\phi} + \Delta\phi$
28: $r_{\mathcal{F}} = \mathcal{F}(\tilde{\psi}, \tilde{\mu})$
29: $r_\phi = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})\tilde{\phi}$
30: $r_{ref} = \langle\breve{\phi}(\tilde{\psi}, \tilde{\mu}), \tilde{\phi}\rangle$
31: **end while**
32: Return $\tilde{\psi}, \tilde{\mu}$

Newton step length adaptation

Algorithm 6.6 FindBifNSA

Input $m_{NSA} \in \mathbb{N}$, maximal step size $\Delta s^+ \in \mathbb{R}_0^+$, tolerance $\epsilon_{NSA} \in \mathbb{R}_0^+$, functions $F : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $\mathcal{P} : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle\cdot,\cdot\rangle$, solutions $(\psi, \mu), (\psi', \mu') \in \mathbb{C}^n \times \mathbb{R}$

Output Approximation $(\tilde{\psi}, \tilde{\mu})$ of $\mathcal{F}(\psi, \mu) = 0$, number of used Newton iterations $\tilde{m}$.

1: Set $\mathcal{P} = \mathcal{I}$ if not specified
2: Set $\langle\cdot,\cdot\rangle = \langle\cdot,\cdot\rangle_2$ if not specified
3: Create deflation matrix $K \in \mathbb{C}^{n\times l_1}$ for $\mathcal{F}_\psi(\psi, \mu)$, with l_1 null vectors, induced by continuous symmetry
4: Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi, \mu)$, $\mathcal{P} = P(\psi, \mu)$ and given $\langle\cdot,\cdot\rangle$ and K
5: Create deflation matrix $K' \in \mathbb{C}^{n\times l_1}$ for $\mathcal{F}_\psi(\psi', \mu')$, with l_1 null vectors, induced by continuous symmetry
6: Calculate (L', W', U') by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi', \mu')$, $\mathcal{P} = P(\psi', \mu')$, $K = K'$ and given $\langle\cdot,\cdot\rangle$
7: $j = 0$
8: **while** $j < m_{NSA}$ and $|L'_{11}| < \epsilon_{NSA}$ **do**
9: $j \leftarrow j + 1$
10: $\breve{\psi} = \psi' - \psi$
11: $\breve{\mu} = \mu' - \mu$
12: Orthogonalize $\breve{\psi}$ to the column vectors of K

13: $\gamma = \sqrt{\|\breve{\psi}\|^2 + \breve{\mu}^2}$
14: $\dot{\psi} = \gamma^{-1}\breve{\psi}$
15: $\dot{\mu} = \gamma^{-1}\breve{\mu}$
16: $\Delta s \leftarrow -\frac{L'_{11}\gamma}{L'_{11}-L_{11}}$
17: $\Delta s \leftarrow \operatorname{sgn}(\Delta s)\min(|\Delta s|, \Delta s^+)$
18: $\tilde{\psi} = \psi' + \Delta s\dot{\psi}$
19: $\tilde{\mu} = \mu' + \Delta s\dot{\mu}$
20: Define $\mathcal{G}$ by (6.9)
21: Define $\mathcal{G}_\psi$ by (6.10)
22: Define $\mathcal{G}_\mu$ by (6.11)
23: Calculate (ψ', μ') by executing NewtonBlockMixed (algorithm 5.7) with $F = \mathcal{F}$, $G = \mathcal{G}$, $x^{(0)} = \tilde{\psi}$, $p^{(0)} = \tilde{\mu}$ and given P and $\langle\cdot,\cdot\rangle$
24: Create deflation matrix $K \in \mathbb{C}^{n\times l_1}$ for $\mathcal{F}_\psi(\psi', \mu')$, with l_1 null vectors, induced by continuous symmetry
25: Calculate $(\tilde{L}, \tilde{W}, \tilde{U})$ by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\tilde{\psi}, \tilde{\mu})$, $\mathcal{P} = P(\tilde{\psi}, \tilde{\mu})$ and given $\langle\cdot,\cdot\rangle$ and K
26: **if** NewtonBlockMixed did not converge or $|\tilde{L}_{11}| > |L'_{11}|$ **then**
27: $\Delta s \leftarrow \frac{1}{2}\Delta s$
28: Go back to line 18
29: **end if**
30: $\psi \leftarrow \psi'$
31: $\mu \leftarrow \mu'$
32: $L \leftarrow L'$
33: $\psi' \leftarrow \tilde{\psi}$
34: $\mu' \leftarrow \tilde{\mu}$
35: $L' \leftarrow \tilde{L}$
36: **end while**
37: Return $\tilde{\psi}, \tilde{\mu}$

Determination of the step size

Algorithm 6.7 AdaptStep

Input Bounds $m_{step} \in \mathbb{N}$, $\epsilon_{step} \in \mathbb{R}_0^+$, range $\Delta s^-, \Delta s^+ \in \mathbb{R}_0^+$ for step size, growth factors $g^-, g^+ \in \mathbb{R}_0^+$ for step size, current step size Δs, $m \in \mathbb{N}$, $\lambda \in \mathbb{R}$

Output Step length Δs to use for next pseudo-arclength step.

1: **if** $|\lambda| < \epsilon_{step}$ **then**
2: $\Delta s \leftarrow \Delta s^-$
3: **else**
4: **if** $m < m_{step}$ **then**
5: $\Delta s \leftarrow g^+\Delta s$
6: **else**
7: $\Delta s \leftarrow g^-\Delta s$
8: **end if**

9: $\Delta s \leftarrow \max(\Delta s, \Delta s^-)$
10: $\Delta s \leftarrow \min(\Delta s, \Delta s^+)$
11: **end if**
12: Return Δs

Analysis of stability

Algorithm 6.8 StabAnalysis

Input Tolerance $\epsilon_{stab} \in [0,1]$, matrix $P \in \mathbb{C}^{(n+1)\times n_P}$ containing points $(\psi^{(j)}, \mu^{(j)})$ as its columns, matrix $B \in \mathbb{C}^{(n+1)\times n_B}$ containing bifurcation points $(\hat{\psi}^{(j)}, \hat{\mu}^{(j)})$ as its columns, matrix $L \in \mathbb{R}^{k\times n_P}$ containing Ritz values $\tilde{\lambda}_1^{(j)}, \ldots, \tilde{\lambda}_k^{(j)}$ as its columns

Output Row matrix S containing entries $S_j = 1$ if P_j represents a stable point, $S_j = 0$ otherwise.

1: Initialize S as an empty 1×0 matrix
2: $I^{(l)} = \begin{pmatrix} 1 \end{pmatrix}$
3: Initialize $I^{(r)}$ as an empty 1×0 matrix
4: **for** $i = 1, \ldots, n_B$ **do**
5: Determine $p_+, p_- \in \{1, \ldots, n_P\}$ with $p_+ = p_- + 1$ the two points $P_{p_-}^T = (\psi^{(p_-)}, \mu^{(p_-)})$ and $P_{p_+}^T = (\psi^{(p_+)}, \mu^{(p_+)})$ closest to the bifurcation point $B_i^T = (\hat{\psi}^{(i)}, \hat{\mu}^{(i)})$
6: $I^{(r)} = \begin{pmatrix} I^{(r)} & p_- \end{pmatrix}$
7: $I^{(l)} = \begin{pmatrix} I^{(l)} & p_+ \end{pmatrix}$
8: **end for**
9: $I^{(r)} = \begin{pmatrix} I^{(r)} & n_P \end{pmatrix}$
10: **for** $i = 1, \ldots, n_B + 1$ **do**
11: $n_i = I_i^{(r)} - I_i^{(l)} + 1$
12: **for** $j = I_i^{(l)}, \ldots, I_i^{(r)}$ **do**
13: $s_j^{(i)} = 1$
14: **for** $l = 1, \ldots, k$ **do**
15: $\tilde{\lambda}_l^{(j)} = L_{lj}$
16: **if** $\tilde{\lambda}_l^{(j)} < 0$ **then**
17: $s_j^{(i)} \leftarrow 0$
18: Break
19: **end if**
20: **end for**
21: **end for**
22: $s = 1 - \frac{1}{n_i} \sum_{j=I_i^{(l)}}^{I_i^{(r)}} s_j^{(i)}$
23: **if** $s < \epsilon_{stab}$ **then**
24: Create matrix $S^{(i)} \in \mathbb{R}^{1\times m}$ with entries 1
25: **else**
26: Create matrix $S^{(i)} \in \mathbb{R}^{1\times m}$ with entries 0
27: **end if**

28: $S = \begin{pmatrix} S & S^{(i)} \end{pmatrix}$
29: **end for**
30: Return S

CHAPTER 7

Automatic exploration

"Gotta catch 'em all!"

– Pokémon –

Chapter highlights:

- We review several types of bifurcation points, and discuss their role in automatic exploration.
- We analyse an algorithm presented in [73], used for the construction of tangent directions to curves that emanate from certain bifurcation points.
- We derive a modified algorithm to use for dynamical systems with discrete symmetries.
- We show how the equivariant branching lemma is used to reduce computational work by incorporating prior knowledge on the system's symmetries.
- The results in this chapter are mainly based on the following references: [48, 55, 62, 73, 72].
- A journal article about the contents of this chapter and chapter 9 has been submitted (see [110]).

7.1 Introduction

In chapter 6 numerical methods for the approximation of solution curves were discussed. Given an initial solution $(\psi^{(0)}, \mu^{(0)})$ of a nonlinear equation $\mathcal{F}(\psi, \mu) = 0$ (with $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $n \in \mathbb{N}$) and a direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$, a finite set of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ... is constructed by application of the pseudo-arclength continuation algorithm. This set approximates a single solution curve through the provided point $(\psi^{(0)}, \mu^{(0)})$.

Applications like the Liouville-Bratu-Gelfand and the Ginzburg-Landau equations (see chapter 2) do not contain a single, but multiple, interconnected, solution curves. Typically these curves distinguish themselves in properties like symmetry and stability of their solutions [92].

Definition 7.1 ((Connected) solution landscape)**.** Given multiple solution curves $(\psi_1(s), \mu_1(s))$, $(\psi_2(s), \mu_2(s))$, $(\psi_3(s), \mu_3(s))$, ... of a nonlinear function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$. If the curves are interconnected, we call the set

$$\{(\psi_i(s), \mu_i(s)) \mid i = 1, 2, 3, \dots\}$$

a (connected) solution landscape of $\mathcal{F}$.

It is possible for solution landscapes to contain a lot of different curves. Some of the examples in chapter 9 show landscapes with more than 40 curves. Automatic exploration is essential for these applications. Given an initial solution $(\psi^{(0)}, \mu^{(0)})$ of $\mathcal{F}(\psi, \mu) = 0$, we want to derive techniques that automatically calculate a complete connected solution landscape through $(\psi^{(0)}, \mu^{(0)})$. Instead of generating a single curve by the pseudo-arclength continuation algorithm, the goal of the current chapter is to create an algorithm that generates a set of multiple, interconnected ones.

7.2 Requisites for automatic exploration

We again consider a general nonlinear function

$$\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (\psi, \mu) \to \mathcal{F}(\psi, \mu)$$

and a point $(\psi^{(0)}, \mu^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$ such that $\mathcal{F}(\psi^{(0)}, \mu^{(0)}) = 0$. As in previous chapters, we restrict ourselves to nonlinear functions with a Hermitian partial Jacobian $\mathcal{F}_\psi$. Pseudo-arclength continuation yields an approximation to a single solution curve $(\psi_1(s), \mu_1(s))$ through $(\psi^{(0)}, \mu^{(0)})$.

Automatic exploration of the connected solution landscape through this point is now realized in two main steps [62, 72]. The first step consists of identifying the points on $(\psi_1(s), \mu_1(s))$ that connect to other solution curves as well. These points will be called branch points (see definition 7.2). After the branch points are identified, the second step constructs the tangent directions to emerging solution curves.

These two steps are sufficient requisites for automatic exploration: the pseudo-arclength continuation method is reapplied, using the branch points and corresponding tangent directions along $(\psi_1(s), \mu_1(s))$, in order to generate new solution curves [62, 72]. The process of identifying branch points and constructing tangent directions is then repeated for these new curves, eventually yielding a complete connected solution landscape.

The algorithm for automatic exploration itself will be further discussed in chapter 8. In the remainder of this chapter we will focus on the two required steps of identifying branch points and constructing tangent directions. Note that the techniques allow for the construction of a landscape consisting of interconnected solution curves. It is not possible to generate curves that are in no way connected to the initial point $(\psi^{(0)}, \mu^{(0)})$.

7.2.1 Identification of branch points

Denote $\mathcal{F}_\Psi$ the full Jacobian of $\mathcal{F}$, this operator is defined (in a point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$) as

$$\mathcal{F}_\Psi(\psi, \mu) : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n : (v, q) \to \mathcal{F}_\psi(\psi, \mu)v + q\mathcal{F}_\mu(\psi, \mu).$$

We denote $\hat{\mathcal{N}}(\mathcal{F}_\Psi(\psi, \mu))$ the kernel of $\mathcal{F}_\Psi(\psi, \mu)$, where we exclude null vectors induced by possible continuous symmetries of $\mathcal{F}$. For the full kernel we use the notation $\mathcal{N}(\mathcal{F}_\Psi(\psi, \mu))$, the range of $\mathcal{F}_\Psi(\psi, \mu)$ is denoted by $\mathcal{R}(\mathcal{F}_\Psi(\psi, \mu))$.

Definition 7.2 (Branch point [72])**.** Consider a point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$ and a nonlinear function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$. We call (ψ, μ) a branch point of $\mathcal{F}$ if $\mathcal{F}(\psi, \mu) = 0$ and

$$\dim(\hat{\mathcal{N}}(\mathcal{F}_\Psi(\psi, \mu))) \geq 2.$$

The value $\mathfrak{mult}(\psi, \mu) = \dim(\hat{\mathcal{N}}(\mathcal{F}_\Psi(\psi, \mu))) - 1$ is called the multiplicity of the branch point.

A branch point $(\psi^{(b)}, \mu^{(b)})$ is a solution of the equation for which the full Jacobian contains a multi-dimensional kernel. As a consequence, branch points are a class of bifurcation points. As stated before, it are the points in which multiple solution curves intersect. We distinguish two types: if the kernel is two-dimensional (multiplicity 1), this intersection occurs for precisely two solution curves. If its dimension is higher than 2 (multiplicity > 1), more than two curves intersect in the same branch point. This last case typically occurs when the problem contains a (discrete) symmetry [73]. Except for branch points, we also consider a class of bifurcations called turning points.

Definition 7.3 (Turning point [72])**.** Consider a point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$ and a nonlinear function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$. We call (ψ, μ) a turning point of $\mathcal{F}$ if $\mathcal{F}(\psi, \mu) = 0$ and

$$\dim(\hat{\mathcal{N}}(\mathcal{F}_\psi(\psi, \mu))) = 1,$$
$$\dim(\hat{\mathcal{N}}(\mathcal{F}_\Psi(\psi, \mu))) = 1.$$

Since the Jacobian is self-adjoint for all of the applications considered in chapter 2, these are the only two types of bifurcation points that we will consider.

To identify branch points during the execution of the pseudo-arclength continuation method, first the bifurcation points on the approximated curve need to be calculated. This was already discussed in detail in chapter 6. For each calculated bifurcation point $(\psi^{(b)}, \mu^{(b)})$ the kernel of the full Jacobian $\mathcal{F}_\Psi(\psi^{(b)}, \mu^{(b)})$ needs to be analysed. If this kernel is one-dimensional the bifurcation is a turning point, otherwise it is a branch point.

Algorithm 7.2, provided on page 231 in appendix 7.6.2, contains pseudo-code for the analysis of the full Jacobian's kernel of a given (bifurcation) point $(\psi^{(b)}, \mu^{(b)})$. First the approximate null vectors $\phi_1, \ldots, \phi_m$ $(m \leq n)$ of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ are calculated (by application of algorithm 3.3, see page 43),

ignoring any induced by continuous symmetries. With m the amount of null vectors, the dimension of $\hat{\mathcal{N}}(\mathcal{F}_\Psi(\psi^{(b)}, \mu^{(b)}))$ is either m or $m+1$. Next, the algorithm checks whether a vector $v^{(0)} \in \mathbb{C}^n$ exists such that

$$\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})v^{(0)} = -\mathcal{F}_\mu(\psi^{(b)}, \mu^{(b)}). \tag{7.1}$$

Denoting $\breve{\phi}_1, \ldots, \breve{\phi}_m$ the null vectors of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})^*$, the adjoint of the partial Jacobian, this is checked by calculating $\xi_1, \ldots, \xi_m$, given by

$$\forall j = 1, \ldots, m : \xi_j = \langle \mathcal{F}_\mu(\psi^{(b)}, \mu^{(b)}), \breve{\phi}_j \rangle.$$

Note that we can choose $\breve{\phi}_i = \phi_i$ $(\forall i = 1, \ldots, m)$ when $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ is self-adjoint. If $|\xi_j|$ is sufficiently small for each $j = 1, \ldots, m$, the partial derivative $\mathcal{F}_\mu(\psi^{(b)}, \mu^{(b)})$ is considered to lie in the (approximate) range of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ [103], and the vector $v^{(0)}$ exists. In this case the kernel of $\mathcal{F}_\Psi(\psi^{(b)}, \mu^{(b)})$ is $m+1$-dimensional. If $|\xi_j|$ does not approximate zero for a certain $j \in \{1, \ldots, m\}$, the kernel has dimension m.

Algorithm 7.2 is part of algorithm 7.1 (page 230), used to determine the type of a given bifurcation point $(\psi^{(b)}, \mu^{(b)})$ and to construct its emerging (normalized) tangent directions. More details on this algorithm are provided throughout the chapter.

7.2.2 Analysis of tangent directions

After a branch point $(\psi^{(b)}, \mu^{(b)})$ has been identified, tangent directions to emerging solution curves need to be constructed. We ignore directions that lead to solution families induced by continuous symmetries (see section 6.3.1). Given a curve $(\psi(s), \mu(s))$ of $\mathcal{F}$, with $(\psi(s_0), \mu(s_0)) = (\psi^{(b)}, \mu^{(b)})$, the tangent direction to $(\psi(s), \mu(s))$ is defined by $(\dot{\psi}, \dot{\mu})$, with [73]

$$\dot{\psi} = \lim_{s \to s_0} \frac{1}{s}(\psi(s) - \psi^{(b)}), \tag{7.2}$$

$$\dot{\mu} = \lim_{s \to s_0} \frac{1}{s}(\mu(s) - \mu^{(b)}). \tag{7.3}$$

An important property of these directions is that they lie in the kernel of the full Jacobian, this will be derived in section 7.3. Given approximate null vectors $(\phi_1, \beta_1), \ldots, (\phi_{m+1}, \beta_{m+1}) \in \mathbb{C}^n \times \mathbb{R}$ of $\mathcal{F}_\Psi(\psi, \mu)$ (ignoring ones induced by continuous symmetries), this property allows us to write $(\psi(s), \mu(s))$ as a linear combination of these vectors:

$$\dot{\psi} = \sum_{j=1}^{m+1} \alpha_j \phi_j, \qquad \dot{\mu} = \sum_{j=1}^{m+1} \alpha_j \beta_j,$$

for certain $\alpha_1, \ldots, \alpha_{m+1} \in \mathbb{R}$. This reduces the problem of constructing tangent directions in $\mathbb{C}^n \times \mathbb{R}$ to approximating the kernel of $\mathcal{F}_\Psi(\psi, \mu)$ and determining the appropriate linear combinations [62, 72].

Finding such a combination is however far from trivial, the complexity of this problem strongly depends on the amount of approximate null vectors. Several cases and techniques will be discussed in further sections of the current chapter.

Remark 7.4. Instead of calculating the tangent directions to emerging curves, an alternative method consists of perturbing the system in the direction of the critical eigenvector. This technique is for example applied in [51] for the analysis of nonlinear engineering structures. Though not as robust as determining tangent directions, the method relies entirely on the Jacobian's kernel, which reduces computational work compared to the algorithms that will be discussed in the remainder of the current chapter. The method generally works fine for branch points with multiplicity 1.

Generalization to higher multiplicity branch points is mentioned as well in [51]. Due to the increased dimension of the Jacobian's kernel, it is however not possible to a priori know the required linear combination of critical eigenvectors the system should be perturbed in. This possibly leads to solution curves, that do emerge from the branch point, being missed.

Given the tangent directions $(\dot\psi^{(1)}, \dot\mu^{(1)})$, $(\dot\psi^{(2)}, \dot\mu^{(2)})$, $(\dot\psi^{(3)}, \dot\mu^{(3)})$, ... to respective solution curves $(\psi_1(s), \mu_1(s))$, $(\psi_2(s), \mu_2(s))$, $(\psi_3(s), \mu_3(s))$, ..., we still need to decide which directions should be used to generate further curves. We want to prevent a recalculation of the curve the branch points were calculated from, neither do we want two applications of the pseudo-arclength continuation method to yield curves that belong to the same group orbit (see definition 6.12) when a discrete symmetry is present.

Algorithm 7.3 (page 232) removes tangent directions from a given list, it is executed after their construction in algorithm 7.1 (page 230). First the group orbits are analysed, this is only required if $\mathcal{F}$ contains a discrete symmetry. If two directions $(\dot\psi^{(i)}, \dot\mu^{(i)})$ and $(\dot\psi^{(j)}, \dot\mu^{(j)})$ (with $j > i > 0$) satisfy

$$\dot\psi^{(i)} \approx g(\dot\psi^{(j)}), \qquad \dot\mu^{(i)} \approx \dot\mu^{(j)}$$

for a certain $g \in G$, the symmetry group of $\mathcal{F}$, direction $(\dot\psi^{(j)}, \dot\mu^{(j)})$ is removed.

To prevent the recalculation of an already constructed curve through the branch point, an approximation $(\dot\psi^{(0)}, \dot\mu^{(0)})$ of the direction to this curve should be provided as input in algorithm 7.3. In practice this tangent direction is approximated by (6.12), using two points $\psi^{(j)}$ and $\psi^{(j-1)}$ of the curve close to the branch point. The algorithm removes the tangent direction $(\dot\psi^{(i)}, \dot\mu^{(i)})$ for which the value

$$\min_{g\in G} \left(1 - |\langle \dot\psi^{(0)}, g(\dot\psi^{(i)})\rangle + \dot\mu^{(0)*}\dot\mu^{(i)}|\right)$$

is smallest. If $\mathcal{F}$ does not contain a dsicrete symmetry, the group action g is ignored in this expression.

7.2.3 Alternative exploration techniques

In order to automatically explore a connected solution landscape, we apply the strategy that consists of identifying branch points and analysing tangent directions to emerging curves. There are alternative approaches with the same purpose, an overview of some of these methods is given in the current section.

In [13] the elastic shell problem is considered. Instead of calculating solution curves for this equation exactly, they are locally approximated by Padé

approximants. Using this technique the nonlinear equation is replaced by a series of linear ones. The solutions to these systems are then used to construct approximations to the curves. The method is however not straightforwardly applicable to general problems, like the ones discussed in chapter 2. Furthermore, the method is only analysed for branch points with multiplicity 1 in [13], a possible extension to higher multiplicities is not discussed.

A method that can be applied to higher multiplicity branch points is discussed in [21]. Near such a point, known solutions are deflated from the nonlinear system in order to find new ones by application of the Newton method. The technique is applied to the Gross-Pitaevskii equation in [21]. Though only the nonlinear equation is adapted in this method, convergence of the Newton method is not guaranteed, possibly failing to find all of the emerging curves. Neither is it clear how null vectors induced by continuous symmetries are perturbed by this deflation, or how these should be handled when solving the underlying linear systems.

Another technique applicable to higher multiplicity branch points is discussed in [57]: in this paper the branch connecting equation associated with a given nonlinear problem is introduced. By performing numerical continuation on this new equation in an additional, artificial, parameter, solutions that lie at a fixed distance of a given branch point are found. Ideally the resulting curve passes through all of these solutions, this is however not always the case in practice [66]. Furthermore, numerical continuation of the branch connecting equation comes with a high additional computational cost, and might fail due to non-convergence of the Newton method.

Given a branch point of general multiplicity in a nonlinear equation, the generalized Lyapunov-Schmidt-Koiter technique is described in [66] to derive a reduced polynomial system, whose solutions are used to construct initial guesses for points on the emerging curves close to the branch point. The method is based on a Taylor expansion. The approach in [66] does not calculate tangent directions directly, but its derivation is similar to the one discussed in the current chapter. The method however misses some details to use it in practice: no robust criteria are derived to check the reduced polynomial system for isolated solutions. Furthermore, there is no mention on how the order of the used Taylor expansion is linked to such solutions existing in the reduced system.

Several methods for automatic exploration at branch points of multiplicity 2 and higher are suggested in [95]. One of these methods investigates possible symmetry breaking of emerging curves, which typically occurs at these kinds of bifurcations. Near these points the nonlinear equation is extended with a symmetry breaking condition, after application of the Newton method this yields solutions with a reduced symmetry group. The symmetry breaking condition typically demands two elements of the state that are equal at the bifurcation, to have a fixed non-zero distance at the system's solution. It is however not possible to know a priori what elements to choose to find all of the different emerging solution curves, possibly leading to curves being missed.

Other methods mentioned in [95] include establishing correctors that proceed parallel to the known branch, and a technique called unfolding (which is also discussed in [62]). Both methods are however not robust, increasing the possibility of curves being missed.

Contrary to the techniques mentioned above, the automatic exploration method based on branch points and tangent directions guarantees each emerging solution curve to be found. Note that in practice curves might still be missed, due to a possible bad choice of internal parameters (see section 8.4). However, the technique seems to be the most robust choice. In the remainder of the chapter the construction of tangent directions will be discussed. We will consider a general branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$ of $\mathcal{F}$. The evaluations of $\mathcal{F}$, $\mathcal{F}_\psi$, ... in $(\psi^{(b)}, \mu^{(b)})$ will be denoted as $\mathcal{F}^{(b)}$, $\mathcal{F}_\psi^{(b)}$, ...

7.3 Lyapunov-Schmidt reduction

A popular approach for analysing bifurcations is Lyapunov-Schmidt reduction [73]: in a neighbourhood of the bifurcation $(\psi^{(b)}, \mu^{(b)})$, the problem is reduced to low-dimensional systems of algebraic equations that contain all of the information of the bifurcations behaviour. The solutions of these systems are then used to construct the tangent directions. In Mei and Schwarzer [73] the method is applied to general problems, leading to three equations that form the base of a first tangent direction construction algorithm.

For generality, we will drop the assumption on self-adjointness of the partial Jacobian $\mathcal{F}_\psi$ in the remainder of the chapter.

7.3.1 Derivation of the base equations

A short overview of the deduction of these equations is given in the current section, based on the analysis done in Mei and Schwarzer [73]. The following assumptions are made:

- $\mathcal{F}_\psi^{(b)}$ is a Fredholm operator of index 0 [79] and zero is a semi-simple eigenvalue of it.
- Orthonormal bases for the kernels of $\mathcal{F}_\psi^{(b)}$ and $\mathcal{F}_\psi^{(b)*}$ (the adjoint partial Jacobian) have been determined:

$$\begin{aligned} \mathcal{N}\left(\mathcal{F}_\psi^{(b)}\right) &= \operatorname{span}\left(\phi_1, \phi_2, \ldots, \phi_{m'}\right), \\ \mathcal{N}\left(\mathcal{F}_\psi^{(b)*}\right) &= \operatorname{span}\left(\breve{\phi}_1, \breve{\phi}_2, \ldots, \breve{\phi}_{m'}\right) \end{aligned} \tag{7.4}$$

  for a certain $m' \in \mathbb{N}$. We assume $\|\phi_i\| = \|\breve{\phi}_i\| = 1$ for each $i = 1, \ldots, m'$. The first $m \leq m'$ vectors of these bases are chosen perpendicular to the null vectors induced by possible continuous symmetries. We have

$$\begin{aligned} \hat{\mathcal{N}}\left(\mathcal{F}_\psi^{(b)}\right) &= \operatorname{span}\left(\phi_1, \phi_2, \ldots, \phi_m\right), \\ \hat{\mathcal{N}}\left(\mathcal{F}_\psi^{(b)*}\right) &= \operatorname{span}\left(\breve{\phi}_1, \breve{\phi}_2, \ldots, \breve{\phi}_m\right). \end{aligned} \tag{7.5}$$

  For the applications described in chapter 2 the partial Jacobian $\mathcal{F}_\psi^{(b)}$ is self-adjoint, implying $\mathcal{N}\left(\mathcal{F}_\psi^{(b)}\right) = \mathcal{N}\left(\mathcal{F}_\psi^{(b)*}\right)$ for these problems.

- We have
$$\mathcal{F}_{\mu}^{(b)} \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)}\right).$$
Note that if this assumption does not hold and m equals 1, the bifurcation $(\psi^{(b)}, \mu^{(b)})$ is in fact a turning point and no tangent directions need to be calculated.

In practice, the kernels $\hat{\mathcal{N}}\left(\mathcal{F}_{\psi}^{(b)}\right)$ and $\hat{\mathcal{N}}\left(\mathcal{F}_{\psi}^{(b)*}\right)$ are approximated using algorithm 3.3 (page 43).

A solution curve $(\psi(s), \mu(s))$ with $(\psi(s_0), \mu(s_0)) = (\psi^{(b)}, \mu^{(b)})$ is considered. The fundamental theorem of linear algebra [103] states that the space $\mathbb{C}^n$ can be decomposed into $\mathcal{N}\left(\mathcal{F}_{\psi}^{(b)}\right)$ and $\mathcal{R}\left(\mathcal{F}_{\psi}^{(b)*}\right)$, this implies that we can write $\psi(s)$ and $\mu(s)$ as

$$\begin{aligned}
\psi(s) &= \psi^{(b)} + \sum_{i=1}^{m'} s\alpha_i\phi_i + sw(\alpha_1, \ldots, \alpha_{m'}, s) && \text{with } w \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)*}\right), \\
\mu(s) &= \mu^{(b)} + s\beta(s) && \text{with } \beta \in \mathbb{R}.
\end{aligned}$$

Applying a Taylor expansion, this yields

$$\begin{aligned}
\psi(s) &= \psi^{(b)} + \sum_{i=1}^{m'} s\alpha_i\phi_i + \sum_{k=1}^{l} s^k w_k + \mathcal{O}(s^{l+1}), \\
\mu(s) &= \mu^{(b)} + \sum_{k=1}^{l} s^k \beta_k + \mathcal{O}(s^{l+1}) \\
&\text{with } l \in \mathbb{N},\ w_1, \ldots, w_l \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)*}\right) \text{ and } \beta_1, \ldots, \beta_l \in \mathbb{R}.
\end{aligned} \tag{7.6}$$

The tangent direction $(\dot{\psi}, \dot{\mu})$ to the solution curve $(\psi(s), \mu(s))$ is given by

$$\begin{aligned}
\dot{\psi} &= \lim_{s \to s_0} \frac{1}{s}\left(\psi(s) - \psi^{(b)}\right) = \sum_{i=1}^{m'} \alpha_i\phi_i + w_1, \\
\dot{\mu} &= \lim_{s \to s_0} \frac{1}{s}\left(\mu(s) - \mu^{(b)}\right) = \beta_1.
\end{aligned} \tag{7.7}$$

We ignore directions that lead to solution families induced by continuous symmetries. This is done by orthogonalizing $\dot{\psi}$ to any null vectors induced by these. Such vectors were given in (7.4) by $\phi_{m+1}, \ldots, \phi_{m'}$ and $\breve{\phi}_{m+1}, \ldots, \breve{\phi}_{m'}$. As a consequence, we set $\alpha_i = 0$ for $i > m$. (7.6) becomes

$$\begin{aligned}
\psi(s) &= \psi^{(b)} + \sum_{i=1}^{m} s\alpha_i\phi_i + \sum_{k=1}^{l} s^k w_k + \mathcal{O}(s^{l+1}), \\
\mu(s) &= \mu^{(b)} + \sum_{k=1}^{l} s^k \beta_k + \mathcal{O}(s^{l+1}).
\end{aligned} \tag{7.8}$$

We rewrite equation $\mathcal{F}(\psi, \mu) = 0$, again using a Taylor expansion.

$$\begin{aligned} \mathcal{F}(\psi,\mu) &= \mathcal{F}_\Psi^{(b)} \begin{pmatrix} \psi - \psi^{(b)} \\ \mu - \mu^{(b)} \end{pmatrix} + R(\psi - \psi^{(b)}, \mu - \mu^{(b)}) \\ &= \sum_{k=1}^{l} s^k \left(\mathcal{F}_\psi^{(b)} w_k + \mathcal{F}_\mu^{(b)} \beta_k + r_k \right) + \mathcal{O}(s^{l+1}), \end{aligned} \tag{7.9}$$

with $R(\psi - \psi^{(b)}, \mu - \mu^{(b)})$ given by

$$R(\psi - \psi^{(b)}, \mu - \mu^{(b)}) = \sum_{j=2}^{\infty} \frac{1}{j!} \mathcal{D}_\Psi^j \mathcal{F}^{(b)} \begin{pmatrix} \sum_{i=1}^{m} s\alpha_i \phi_i + \sum_{k=1}^{l} s^k w_k + \mathcal{O}(s^{l+1}) \\ \sum_{k=1}^{l} s^k \beta_k + \mathcal{O}(s^{l+1}) \end{pmatrix}^j .$$

The term r_k in (7.9) represents the coefficient of s^k in $R(\psi - \psi^{(b)}, \mu - \mu^{(b)})$ ($\forall k = 1, \ldots, l$), this vector is explicitly given by

$$r_k = \sum_{j=2}^{k} \frac{1}{j!} \sum_{\substack{(k_1,\ldots,k_j) \\ \in \mathcal{K}_j^{(k)}}} \mathcal{D}_\Psi^j \mathcal{F}^{(b)} \begin{pmatrix} x_{k_1} \\ \beta_{k_1} \end{pmatrix} \begin{pmatrix} x_{k_2} \\ \beta_{k_2} \end{pmatrix} \ldots \begin{pmatrix} x_{k_j} \\ \beta_{k_j} \end{pmatrix} \tag{7.10}$$

$$\text{with } x_1 = \sum_{i=1}^{m} \alpha_i \phi_i + w_1, \qquad \forall p = 2, \ldots, k-1 : x_p = w_p.$$

The set $\mathcal{K}_j^{(k)}$ that appears in the equation is defined by ($\forall k = 1, \ldots, l$, $j = 2, \ldots, k$)

$$\mathcal{K}_j^{(k)} = \left\{ (k_1, \ldots, k_j) \in \mathbb{N}^j \mid \sum_{p=1}^{j} k_p = k, \forall p = 1 \ldots j : k_p \in \{1 \ldots k - j + 1\} \right\}. \tag{7.11}$$

The following three equations are now derived in Mei and Schwarzer [73] (for $k = 1, \ldots, l$):

$$\mathcal{F}_\psi^{(b)} w_k + \mathcal{F}_\mu^{(b)} \beta_k = -r_k, \tag{7.12}$$

$$\forall j = 1, \ldots, m : \langle \phi_j, w_k \rangle = 0, \tag{7.13}$$

$$\forall j = 1, \ldots, m : \langle \breve{\phi}_j, r_k \rangle = 0. \tag{7.14}$$

These equations form the base of the upcoming algorithms. Note that, since $r_1 = 0$, the first equation implies that (w_1, β_1) is a null vector of $\mathcal{F}_\Psi^{(b)}$. As a consequence, (7.7) states that the tangent directions are constructed as a linear combination of vectors in $\hat{\mathcal{N}}\left(\mathcal{F}_\Psi^{(b)}\right)$. More details on Lyapunov-Schmidt reduction can be found in e.g. Mei and Schwarzer [73] and Mei [72].

7.3.2 General algorithm for the construction of tangent directions

In Mei and Schwarzer [73] a general algorithm is deduced to construct reduced systems of equations for $\alpha_1, \ldots, \alpha_m, \beta_{k-1}$ $(k = 2, 3, \ldots)$. Together with (7.7) the solutions of these systems yield the tangent directions. The algorithm is based on equations (7.12), (7.13) and (7.14), which are used to write the terms r_k and w_k as polynomials in the variables $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_{k-1}$. Under the assumptions

$$\forall j = 1, \ldots, m : t_j = \frac{\partial}{\partial \psi} \mathcal{F}_{\Psi}^{(b)} \begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix} \phi_j \notin \mathcal{R}\left(\mathcal{F}_{\Psi}^{(b)}\right), \tag{7.15}$$

$$\forall q_1, \ldots, q_m \in \mathbb{R} : \sum_{j=1}^{m} q_j \langle \breve{\phi}_k, t_i \rangle = 0 \text{ for each } k \iff q_1 = \cdots = q_m = 0, \tag{7.16}$$

$$t_0 = \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix} \begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix} \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)}\right), \tag{7.17}$$

it is possible to write the term β_k $(k = 1, 2, \ldots)$ as a polynomial in the variables $\alpha_1, \ldots, \alpha_m$ as well. The vector $v^{(0)}$ is implicitly defined by $w_1 = \beta_1 v^{(0)}$. Assumptions (7.15) and (7.16) imply that $\ker\left(\mathcal{F}_{\psi}^{(b)*}\right)$ is spanned by the projections of $t_1, \ldots, t_m$ on this kernel. The method described in Mei and Schwarzer [73] is given by algorithm 7.5 (rewritten to emphasize the construction of the reduced systems).

Algorithm 7.5 (Creation of tangent directions based on Mei and Schwarzer [73])**.**
Initial:
For $k = 1$, use the equation

$$\mathcal{F}_{\psi}^{(b)} w_1 = -\mathcal{F}_{\mu}^{(b)} \beta_1$$

to write $w_1 \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)*}\right)$ as a polynomial in β_1.
Iteration:
For $k = 2, 3, \ldots$ do:
Step 1: Use the polynomial expression of w_{k-1} (and β_{k-2} if $k \geq 3$) together with (7.10) to write r_k as a polynomial in $\alpha_1, \ldots, \alpha_m, \beta_{k-1}$.
Step 2: Substitute this polynomial expression in the equation

$$\forall j = 1, \ldots, m : \langle \breve{\phi}_j, r_k \rangle = 0.$$

This yields a reduced system of equations for $\alpha_1, \ldots, \alpha_m, \beta_{k-1}$.
Step 3: Check whether the system has real and isolated solutions $\alpha_1(\beta_{k-1}), \ldots, \alpha_m(\beta_{k-1})$ for β_{k-1} in an open interval. Solve the system for these solutions, each one corresponds to a tangent direction of a different solution curve by application of (7.7).
Step 4a: If only isolated solutions exist in the reduced system, stop the algorithm.

Step 4b: If non-isolated solutions exist as well, use the system to write β_{k-1} as a polynomial of $\alpha_1, \ldots, \alpha_m$ and use this result to write r_k as a polynomial in these variables as well. Substitute this expression for r_k into the system

$$\begin{aligned} &\mathcal{F}_\psi^{(b)} w_k + \mathcal{F}_\mu^{(b)} \beta_k = -r_k, \\ &\forall j = 1, \ldots, m : \langle \phi_j, w_k \rangle = 0 \end{aligned}$$

and solve it to find a polynomial expression for w_k in $\alpha_1, \ldots, \alpha_m, \beta_k$. ◇

The complexity of the polynomial expressions in algorithm 7.5 strongly depends on the dimension m of $\hat{\mathcal{N}}\left(\mathcal{F}_\psi^{(b)}\right)$. No general exact expressions were derived in Mei and Schwarzer [73] for the case $m > 1$.

7.3.3 The algebraic branching equation

If the kernel $\hat{\mathcal{N}}(\mathcal{F}_\psi^{(b)})$ is one-dimensional ($m = 1$), algorithm 7.5 only requires a single iteration: the first reduced system that is constructed does not contain non-isolated solutions [73]. In this case the algorithm is equivalent to the algebraic branching equation [61, 3, 10, 62, 73]. The reduced system for α_1 and β_1 is given by

$$\begin{aligned} &a\alpha_1^2 + b\alpha_1\beta_1 + c\beta_1^2 = 0 \qquad (7.18) \\ &\text{with } a = \langle \breve{\phi}_1, \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)} \phi_1 \phi_1 \rangle, \\ &\qquad b = \langle \breve{\phi}_1, \mathcal{F}_{\psi\psi}^{(b)} \phi_1 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} \phi_1 \rangle, \\ &\qquad c = \langle \breve{\phi}_1, \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)} v^{(0)} v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} v^{(0)} + \frac{1}{2}\mathcal{F}_{\mu\mu}^{(b)} \rangle, \end{aligned}$$

where $v^{(0)} \in \mathcal{R}\left(\mathcal{F}_\psi^{(b)*}\right)$ is solved from the equation

$$\mathcal{F}_\psi^{(b)} v^{(0)} = -\mathcal{F}_\mu^{(b)}. \qquad (7.19)$$

Equation (7.18) is solved for α_1 and β_1. Together with $w_1 = \beta_1 v^{(0)}$ the tangent directions are constructed by application of (7.7).

Algorithm 7.4 on page 233 in appendix 7.6.2 contains pseudo-code that constructs the two (normalized) tangent directions for the case $m = 1$, based on the algebraic branching equation (7.18). The solution $v^{(0)}$ of the linear system (7.19) is approximated by deflated GMRES (algorithm 3.4, page 44) in practice.

7.4 Construction of tangent directions in presence of discrete symmetries

For problems that exhibit discrete symmetry, branch points $(\psi^{(b)}, \mu^{(b)})$ arise for which $\hat{\mathcal{N}}(\mathcal{F}_\psi^{(b)})$ is multi-dimensional [73, 72], complicating the polynomials used in algorithm 7.5. In Mei and Schwarzer [73] no exact expressions for these

polynomials were derived and hence only implicit reduced systems of equations are given, which limits the use of the algorithm in its current form by numerical software. Furthermore, it is not specified how these systems can be checked for real and isolated solutions.

In the current section we derive exact polynomial expressions that yield explicit systems for the case $m = 2$. A simple rule is used to check these for real and isolated solutions. The derivation yields an updated algorithm used when the dimension m of the kernel $\hat{\mathcal{N}}(\mathcal{F}_\psi^{(b)})$ equals 2 (see algorithm 7.6).

Only the cases $m = 1$ and $m = 2$ are encountered when constructing tangent directions for the applications described in chapter 2. The cases for $m \geq 3$ will not be discussed, but we note that an algorithm for these cases can be derived in a similar way.

7.4.1 Construction of a reduced system of equations

Set $m = 2$, assumptions (7.15) and (7.16) allow us to choose $\breve{\phi}_1$ and $\breve{\phi}_2$ such that:

$$\begin{aligned}
&\frac{\partial}{\partial\psi}\mathcal{F}_\Psi^{(b)}\begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix}\phi_1 = q_1\breve{\phi}_1 + u_1 && \text{with } u_1 \in \mathcal{R}\left(\mathcal{F}_\psi^{(b)}\right), q_1 \in \mathbb{R}_0, \\
&\frac{\partial}{\partial\psi}\mathcal{F}_\Psi^{(b)}\begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix}\phi_2 = q_2\breve{\phi}_1 + q_3\breve{\phi}_2 + u_2 && \text{with } u_2 \in \mathcal{R}\left(\mathcal{F}_\psi^{(b)}\right), q_2 \in \mathbb{R}, q_3 \in \mathbb{R}_0.
\end{aligned} \tag{7.20}$$

A modification of algorithm 7.5 for the case $m = 2$ is given by algorithm 7.6. In this last algorithm the coefficients required to write the different polynomial expressions are derived in order to determine explicit reduced systems of equations for $\alpha_1, \ldots, \alpha_m, \beta_{k-1}$. A detailed derivation of algorithm 7.6 is provided in appendix 7.6.1.

Algorithm 7.6 (Creation of tangent directions for the $m = 2$ case)**.**
Initial:
Step 1: Solve the equation

$$\mathcal{F}_\psi^{(b)} v^{(0)} = -\mathcal{F}_\mu^{(b)}, \qquad v^{(0)} \in \mathcal{R}\left(\mathcal{F}_\psi^{(b)*}\right) \subset \mathbb{C}^n$$

by applying deflated GMRES (algorithm 3.4, page 44). Note that w_1 (defined in (7.8)) can be written as

$$w_1 = \beta_1 v^{(0)}. \tag{7.21}$$

Step 2: Calculate the following terms in the space $\mathbb{C}^n$:

$$\begin{aligned}
&y_0^{(2)} = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)}\phi_2\phi_2, && y_1^{(2)} = \mathcal{F}_{\psi\psi}^{(b)}\phi_1\phi_2, && y_2^{(2)} = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)}\phi_1\phi_1, \\
&t_1 = \mathcal{F}_{\psi\psi}^{(b)}\phi_1 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)}\phi_1, && t_2 = \mathcal{F}_{\psi\psi}^{(b)}\phi_2 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)}\phi_2, && \\
&t_0 = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)} v^{(0)} v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} v^{(0)} + \frac{1}{2}\mathcal{F}_{\mu\mu}^{(b)}. &&&&
\end{aligned}$$

Note that r_2 (defined in (7.10)) can be written as

$$r_2 = \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} y_i^{(2)} + \beta_1 \alpha_1 t_1 + \beta_1 \alpha_2 t_2 + \beta_1^2 t_0. \tag{7.22}$$

Step 3: Calculate the scalars

$$\begin{aligned} &\forall j = 1,2, i = 0,1,2 : a_i^{(2,j)} = \langle \breve{\phi}_j, y_i^{(2)} \rangle, \\ &b^{(1)} = \langle \breve{\phi}_1, t_1 \rangle, \qquad b^{(2)} = \langle \breve{\phi}_2, t_2 \rangle, \qquad b^{(3)} = \langle \breve{\phi}_1, t_2 \rangle. \end{aligned}$$

These values form the coefficients of the following reduced system of equations:

$$\begin{cases} \sum\limits_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} a_i^{(2,1)} + \beta_1 \alpha_1 b^{(1)} + \beta_1 \alpha_2 b^{(3)} = 0, \\ \sum\limits_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} a_i^{(2,2)} + \beta_1 \alpha_2 b^{(2)} = 0. \end{cases} \tag{7.23}$$

Step 4: Check whether $b^{(2)} a_0^{(2,1)} = b^{(3)} a_0^{(2,2)}$, $a_2^{(2,2)} = 0$ and $\forall i = 1,2 : b^{(2)} a_i^{(2,1)} = b^{(1)} a_{i-1}^{(2,2)} + b^{(3)} a_i^{(2,2)}$.
Step 5a: If this is the case, stop the algorithm: system (7.23) only contains isolated solutions, these correspond to the tangent directions.
Step 5b: If this is not the case, the only isolated solution of (7.23) is given by $(\alpha_1, \alpha_2, \beta_1) = (0, 0, a)$ with $a \in \mathbb{R}$. The system contains non-isolated solutions as well. Set $\kappa_{-1}^{(1)} = \kappa_2^{(1)} = 0$ and calculate

$$\begin{aligned} &\forall i = 0,1 : \kappa_i^{(1)} = -\frac{a_i^{(2,2)}}{b^{(2)}}, \\ &z_0^{(2)} = y_0^{(2)} + \kappa_0^{(1)} t_2 + {\kappa_0^{(1)}}^2 t_0, \quad z_1^{(2)} = y_1^{(2)} + \kappa_0^{(1)} t_1 + \kappa_1^{(1)} t_2 + 2\kappa_0^{(1)} \kappa_1^{(1)} t_0, \\ &z_2^{(2)} = y_2^{(2)} + \kappa_1^{(1)} t_1 + {\kappa_1^{(1)}}^2 t_0, \\ &q_0^{(1)} = \kappa_0^{(1)} v^{(0)} + \phi_2, \qquad\qquad q_1^{(1)} = \kappa_1^{(1)} v^{(0)} + \phi_1. \end{aligned}$$

Note that β_1, r_2 and x_1 (defined in (7.8) and (7.10)) can respectively be written as

$$\beta_1 = \sum_{i=0}^{1} \alpha_1^i \alpha_2^{1-i} \kappa_i^{(1)}, \tag{7.24}$$

$$r_2 = \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} z_i^{(2)}, \tag{7.25}$$

$$x_1 = \sum_{i=0}^{1} \alpha_1^i \alpha_2^{1-i} q_i^{(1)}. \tag{7.26}$$

Iteration:
For $k = 3, 4, \ldots$ do:
Step 1: Solve the equations

$$\forall i = 0, \ldots, k-1 : \mathcal{F}_{\psi}^{(b)} v_i^{(k-1)} = -z_i^{(k-1)}, \qquad v_i^{(k-1)} \in \mathcal{R}\left(\mathcal{F}_{\psi}^{(b)}\right) \subset \mathbb{C}^n$$

by applying deflated GMRES (algorithm 3.4, page 44). Note that w_{k-1} (defined in (7.8)) can be written as

$$w_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} v_i^{(k-1)} + \beta_{k-1} v^{(0)}. \tag{7.27}$$

Step 2: Calculate the following terms in the space $\mathbb{C}^n$:

$$\begin{aligned}
&\forall i = 0, \ldots, k : \\
&y_i^{(k)} = \sum_{\substack{(i_1,i_2) \\ \in \mathcal{I}_{1,k-1}^{(i,2)}}} \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_{i_1}^{(1)} \\ \kappa_{i_1}^{(1)} \end{pmatrix} \begin{pmatrix} v_{i_2}^{(k-1)} \\ 0 \end{pmatrix} \\
&\quad + \frac{1}{2} \sum_{k_1=2}^{k-2} \sum_{\substack{(i_1,i_2) \\ \in \mathcal{I}_{k_1,k-k_1}^{(i,2)}}} \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_{i_1}^{(k_1)} \\ \kappa_{i_1}^{(k_1)} \end{pmatrix} \begin{pmatrix} q_{i_2}^{(k-k_1)} \\ \kappa_{i_2}^{(k-k_1)} \end{pmatrix} \\
&\quad + \sum_{j=3}^{k} \frac{1}{j!} \sum_{\substack{(k_1,\ldots,k_j) \\ \in \mathcal{K}_j^{(k)}}} \sum_{\substack{(i_1,\ldots,i_j) \\ \in \mathcal{I}_{k_1,\ldots,k_j}^{(i,j)}}} \mathcal{D}_{\Psi}^{j} \mathcal{F}^{(b)} \begin{pmatrix} q_{i_1}^{(k_1)} \\ \kappa_{i_1}^{(k_1)} \end{pmatrix} \begin{pmatrix} q_{i_2}^{(k_2)} \\ \kappa_{i_2}^{(k_2)} \end{pmatrix} \cdots \begin{pmatrix} q_{i_j}^{(k_j)} \\ \kappa_{i_j}^{(k_j)} \end{pmatrix}.
\end{aligned} \tag{7.28}$$

The set $\mathcal{K}_j^{(k)}$ (for $j = 3, \ldots, k$) is defined by (7.11), $\mathcal{I}_{k_1,\ldots,k_j}^{(i,j)}$ (for $i = 0, \ldots, k$, $j = 3, \ldots, k$, $k_1, \ldots, k_j \in \mathcal{K}_j^{(k)}$) by

$$\mathcal{I}_{k_1,\ldots,k_j}^{(i,j)} = \left\{ (i_1, \ldots, i_j) \in \mathbb{N}^j \mid \sum_{p=1}^{j} i_p = i, \forall p = 1 \ldots j : i_p \in \{0, \ldots, k_p\} \right\}. \tag{7.29}$$

Note that r_k (defined in (7.10)) can be written as

$$r_k = \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} y_i^{(k)} + \beta_{k-1}\alpha_1 \left(t_1 + 2\kappa_1^{(1)} t_0\right) + \beta_{k-1}\alpha_2 \left(t_2 + 2\kappa_0^{(1)} t_0\right). \tag{7.30}$$

Step 3: Calculate the scalars

$$\forall j = 1, 2, i = 0, \ldots, k : a_i^{(k,j)} = \langle \breve{\phi}_j, y_i^{(k)} \rangle.$$

These values form the coefficients of the following reduced system of equations:

$$\begin{cases} \sum\limits_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,1)} + \beta_{k-1}\alpha_1 b^{(1)} + \beta_{k-1}\alpha_2 b^{(3)} = 0, \\ \sum\limits_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,2)} + \beta_{k-1}\alpha_2 b^{(2)} = 0. \end{cases} \tag{7.31}$$

Step 4: Check whether $b^{(2)}a_0^{(k,1)} = b^{(3)}a_0^{(k,2)}$, $a_k^{(k,2)} = 0$ and $\forall i = 1, \ldots, k$: $b^{(2)}a_i^{(k,1)} = b^{(1)}a_{i-1}^{(k,2)} + b^{(3)}a_i^{(k,2)}$.
Step 5a: If this is the case, stop the algorithm: system (7.31) only contains isolated solutions, these correspond to the tangent directions.
Step 5b: If this is not the case, the only isolated solution of (7.31) is given by $(\alpha_1, \alpha_2, \beta_{k-1}) = (0, 0, a)$ with $a \in \mathbb{R}$. The system contains non-isolated solutions as well. Set $\kappa_{-1}^{(k-1)} = \kappa_k^{(k-1)} = 0$ and calculate

$$\begin{aligned}
&\forall i = 0, \ldots, k-1 : \kappa_i^{(k-1)} = -\frac{a_i^{(k,2)}}{b^{(2)}}, \\
&\forall i = 0, \ldots, k : z_i^{(k)} = y_i^{(k)} + \kappa_{i-1}^{(k-1)} \left(t_1 + 2\kappa_1^{(1)} t_0\right) + \kappa_i^{(k-1)} \left(t_2 + 2\kappa_0^{(1)} t_0\right), \\
&\forall i = 0, \ldots, k-1 : q_i^{(k-1)} = v_i^{(k-1)} + \kappa_i^{(k-1)} v^{(0)}.
\end{aligned}$$

Note that β_{k-1}, r_k and x_{k-1} (defined in (7.8) and (7.10)) can respectively be written as

$$\beta_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} \kappa_i^{(k-1)}, \tag{7.32}$$

$$r_k = \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} z_i^{(k)}, \tag{7.33}$$

$$\beta_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} q_i^{(k-1)}. \tag{7.34}$$

◇

Algorithm 7.6 gives rise to reduced systems of equations of the form

$$f(\alpha_1, \alpha_2, \beta_{k-1}) = \begin{cases} \sum\limits_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,1)} + \beta_{k-1}\alpha_1 b^{(1)} + \beta_{k-1}\alpha_2 b^{(3)} = 0, \\ \sum\limits_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,2)} + \beta_{k-1}\alpha_2 b^{(2)} = 0 \end{cases} \tag{7.35}$$

with $k \geq 2$ and known coefficients $a_0^{(k,1)}$, …, $a_k^{(k,1)}$, $a_0^{(k,2)}$, …, $a_k^{(k,2)}$, $b^{(1)}$, $b^{(2)}$, $b^{(3)}$. The system constructed by executing the initial steps ($k = 2$) always contains an isolated solution $(\alpha_1, \alpha_2, \beta_1) = (0, 0, a)$ ($a \in \mathbb{R}$). Application of (7.7) and (7.21) leads to a first tangent direction. If this system contains other isolated solutions as well, the corresponding tangent directions are constructed with the same formulas.

The algorithm is continued if this initial reduced system also contains non-isolated solutions, possibly yielding further systems with a sole isolated solution $(\alpha_1, \alpha_2, \beta_{k-1}) = (0, 0, a)$ ($a \in \mathbb{R}$). Relation (7.24) results in $\beta_1 = 0$, consequently (7.7) yields the zero vector and does not actually correspond to a real tangent direction.

Eventually the algorithm will find a system that only contains isolated solutions. Details on how to solve such a system will be provided in section

7.4.2. Tangent directions are constructed from these solutions by application of (7.7), (7.21) and (7.24).

The case $m = 2$ typically occurs when the underlying problem exhibits discrete symmetry, and the branch point is symmetric as well [73, 72]. The amount of required iterations in algorithm 7.6 depends on this symmetry group. For a branch point invariant under the actions of C_p or D_p ($p \geq 3$), the reduced system (7.35) has isolated solutions for $k = p - 1$, constructed after $p - 2$ iterations.

The system itself is also invariant under the actions of C_p, respectively D_p in this case. Due to this invariance, most solutions of (7.35) result in tangent directions to solution curves in the same group orbit. After the construction of the directions, algorithm 7.3 (page 232) is however executed. As stated before, this algorithm selects a single direction for each group orbit. For p even/odd this results in respectively 3 or 2 different directions.

Pseudo-code for algorithm 7.6 is given by algorithm 7.7 on page 236 in appendix 7.6.2. It is part of algorithm 7.5 (page 233), used for the construction of (normalized) tangent directions to solution curves emerging from a given branch point $(\psi^{(b)}, \mu^{(b)})$ for the case $\dim\left(\hat{\mathcal{N}}\left(\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\right)\right) = 2$. An efficient method to calculate the terms (7.28) in algorithm 7.7 is given by the pseudo-code of algorithm 7.9 (page 239). The sets (7.11) and (7.29) that appear in (7.28) are created by algorithms 7.10 (page 240) and 7.11 (page 240).

Algorithm 7.5 was used for e.g. the results of the Ginzburg-Landau equation applied to triangle- and star-shaped materials in sections 9.5.2 and 9.5.4. In both examples, it yielded the required tangent directions for each branch point $(\psi^{(b)}, \mu^{(b)})$ with $\dim\left(\hat{\mathcal{N}}\left(\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\right)\right) = 2$.

7.4.2 Solving the reduced system of equations

By applying algorithm 7.6 eventually a reduced system of equations of the form (7.35) is constructed, for a certain $k \geq 2$ and known coefficients, such that the system only contains isolated solutions. We had not yet discussed how this system is solved.

To find the isolated solutions we fix β_{k-1}. This does not reduce the amount of tangent directions eventually constructed, since these directions are determined up to normalization. Though in some cases it is possible to solve (7.35) exactly (for e.g. $k = 2$ solving this equation is similar to determining the intersection of two conics, for which multiple algorithms exist (see e.g. [84])), we will approximate its solutions with a Newton algorithm (see section 4.2). This requires the Jacobian of (7.35), given by

$$\begin{pmatrix} \frac{\partial f(\alpha_1,\alpha_2,\beta_{k-1})}{\partial \alpha_1} & \frac{\partial f(\alpha_1,\alpha_2,\beta_{k-1})}{\partial \alpha_2} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{k} i\alpha_1^{i-1}\alpha_2^{k-i}a_i^{(k,1)} + \beta_{k-1}b^{(1)} & \sum_{i=0}^{k-1}(k-i)\alpha_1^{i}\alpha_2^{k-1-i}a_i^{(k,1)} + \beta_{k-1}b^{(3)} \\ \sum_{i=1}^{k} i\alpha_1^{i-1}\alpha_2^{k-i}a_i^{(k,2)} & \sum_{i=0}^{k-1}(k-i)\alpha_1^{i}\alpha_2^{k-1-i}a_i^{(k,2)} + \beta_{k-1}b^{(2)} \end{pmatrix}. \tag{7.36}$$

In practice the solutions are found by applying the Newton algorithm for a high amount of different initial guesses. Together with this Newton solver, algorithm 7.6 allows for the construction of the tangent directions arising from branch points $(\psi^{(b)}, \mu^{(b)})$ with a kernel $\hat{\mathcal{N}}(\mathcal{F}_\psi^{(b)})$ of dimension 2.

Pseudo-code, for solving a reduced system with given coefficients, is provided by algorithm 7.8 on page 238 in appendix 7.6.2. It is executed in algorithm 7.5 (page 233) after the coefficients for the system have been constructed.

7.5 Finding tangent directions with the equivariant branching lemma

For the considered applications of chapter 2, it is always possible to apply algorithm 7.6 for branch points $(\psi^{(b)}, \mu^{(b)})$ with a kernel $\hat{\mathcal{N}}(\mathcal{F}_\psi^{(b)})$ of dimension 2. Because the amount of required iterations of the algorithm depends on the symmetry group of the problem, it becomes less practical for problems that are strongly symmetrical: if the branch point is invariant under the actions of C_p or D_p for a certain $p \geq 3$, $p-2$ iterations ($k = p-1$ in algorithm 7.6) might be required before a reduced system that only contains isolated solutions is obtained [73]. Before this system is found, $((p-1)p-2)/2$ linear systems with the partial Jacobian $\mathcal{F}_\psi^{(b)}$ need to be solved. This is unwanted for $p \gg 1$.

For $p \geq 4$ one can still apply the algorithm to determine an equation for β_1 and w_1 as a polynomial in the variables α_1 and α_2 (see (7.21) and (7.24)), this requires only a single linear system to be solved. For D_p symmetric problems the unknowns α_1 and α_2 can then alternatively be determined by application of the equivariant branching lemma (EBL) [48, 55]. This lemma links the symmetry group of the problem to predict the symmetries of solution curves that emerge at branch points. In Schlömer [92] the EBL was used for the prediction of the symmetry groups of emerging curves, but it was not used for the calculation of the tangent directions themselves.

This alternative method for the construction of tangent directions (for the case $m = 2$) will be described in the current section. Note that if the symmetry group of the problem is given by C_p (for a certain $p \geq 3$), the alternative approach cannot be used.

7.5.1 Absence of continuous symmetry

We start the section by describing the equivariant branching lemma for problems derived from a partial differential equation, and then show how it is applied in the construction of tangent directions.

Definition 7.7 (Group invariance of a point [48, 72])**.** A point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$ is considered invariant under the actions of a symmetry group G' if

$$\forall g \in G' : g(\psi) = \psi.$$

Definition 7.8 (Isotropy subgroup [47, 48])**.** Consider a symmetry group G and a point $(\psi, \mu) \in \mathbb{C}^n \times \mathbb{R}$. The isotropy subgroup $G' \subseteq G$ of (ψ, μ) is defined by

$$G' = \{g \in G : g(\psi) = \psi\}.$$

We consider a solution curve $(\psi(s), \mu(s))$ of a function $\mathcal{F}$ with a branch point $(\psi^{(b)}, \mu^{(b)})$. The function does not contain any continuous symmetries, but is invariant under the actions of a compact Lie group G (see definition 6.8). The isotropy subgroup of the branch point is given by $G' \subseteq G$. The equivariant branching lemma gives necessary conditions for the existence of symmetry-breaking curves emanating from $(\psi^{(b)}, \mu^{(b)})$.

Lemma 7.9 (Equivariant branching lemma for PDE's [48])**.** Let G be a compact Lie group, and consider a function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$ that is invariant under the actions of G. Assume $(\psi(s), \mu(s))$ is a solution curve of $\mathcal{F}$ with $(\psi(0), \mu(0)) = (\psi^{(b)}, \mu^{(b)})$ a branch point, and let $G' \subseteq G$ be its isotropy subgroup. Denote $\lambda(s)$ the eigenvalue of $\mathcal{F}_\psi(\psi(s), \mu(s))$ associated with the bifurcation ($\lambda(0) = 0$). Consider $G'' \subseteq G'$. If

- G' acts absolutely irreducible on $\mathcal{N}\left(\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\right)$
- The eigenvalue $\lambda(s)$ crosses the origin with non-zero speed ($\lambda'(0) \neq 0$),
- G'' is an axial subgroup: the space $\mathrm{Fix}(G'') = \{v \in \mathcal{N}\left(\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\right) \mid \forall g \in G'' : g(v) = v\}$ is one-dimensional,

then there exists an unique curve of solutions emanating from $(\psi^{(b)}, \mu^{(b)})$ that are precisely invariant under the actions of G''.

Proof. Define $Q = I - \sum_{i=1}^{m} \langle \phi_i, \cdot \rangle \breve{\phi}_i$ the projection from $\mathbb{C}^n$ onto $\mathcal{R}\left(\mathcal{F}_\psi^{(b)}\right)$. Using Lyapunov-Schmidt reduction [48], the curve (ψ, μ) is written as

$$\begin{aligned} \psi &= \psi^{(b)} + v + w(v, \beta) && \text{with } v \in \mathcal{N}\left(\mathcal{F}_\psi^{(b)}\right), w \in \mathcal{R}\left(\mathcal{F}_\psi^{(b)^*}\right), \\ \mu &= \mu^{(b)} + \beta && \text{with } \beta \in \mathbb{R}, \end{aligned}$$

where $w(v, \beta)$ is determined uniquely as a function of v and β by solving

$$Q\mathcal{F}(\psi^{(b)} + v + w, \mu^{(b)} + \beta) = 0.$$

The assumptions imply that the classic equivariant branching lemma [48, 55] can be applied to the function

$$f : \mathcal{N}\left(\mathcal{F}_\psi^{(b)}\right) \times \mathbb{R} \to \mathbb{C}^n : (v, \beta) \to (I - Q)\mathcal{F}(\psi^{(b)} + v + w(v, \beta), \mu^{(b)} + \beta).$$

Hence there exists a unique curve $(v(\beta), \beta)$ of solutions of $f(v, \beta) = 0$, emanating from $(v, \beta) = (0, 0)$, where the symmetry of the solutions is G''. Using G'-invariance of both $\mathcal{F}$ and $(\psi^{(b)}, \mu^{(b)})$, existence of a unique curve $(\psi(s), \mu(s))$ emanating from $(\psi^{(b)}, \mu^{(b)})$ is implied, where the symmetry of $(\psi(s), \mu(s))$ is G''. □

For the applications described in the thesis, we only consider certain dihedral Lie groups: $G = D_p$ for $p \geq 4$. Let $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$ be a branch point such that $\mathcal{N}(\mathcal{F}_\psi^{(b)})$ is two-dimensional, and let the isotropy subgroup G'

be given by D_q ($q \leq p$). Consider an axial subgroup $G'' \subseteq D_q$ and let the assumptions of lemma 7.9 hold. The lemma predicts that a unique solution curve with precisely G'' as its group invariance emerges from $(\psi^{(b)}, \mu^{(b)})$.

Choose $g \in G''$, such that at least one vector of ϕ_1, ϕ_2 and $v^{(0)}$ (see section 7.3) is not invariant under g. We will now use the knowledge of this symmetry to calculate the unknowns α_1 and α_2 that construct the tangent direction $(\dot{\psi}, \dot{\mu})$, given by (7.7), to this curve.

Let (ψ, μ) be a point of the G''-symmetric solution curve that emerges from $(\psi^{(b)}, \mu^{(b)})$. We have $g(\psi) = \psi$. If (ψ, μ) is chosen sufficiently close to $(\psi^{(b)}, \mu^{(b)})$, we have

$$\begin{cases} \psi \approx \psi^{(b)} + \epsilon\dot{\psi}, \\ \mu \approx \mu^{(b)} + \epsilon\dot{\mu} \end{cases} \tag{7.37}$$

for a certain $\epsilon \in \mathbb{R}$, $\epsilon \ll 1$. The tangent direction $(\dot{\psi}, \dot{\mu})$ is given by (see (7.7),(7.21) and (7.24))

$$\begin{cases} \dot{\psi} = \alpha_1 \left(\phi_1 + \kappa_1^{(1)} v^{(0)}\right) + \alpha_2 \left(\phi_2 + \kappa_0^{(1)} v^{(0)}\right), \\ \dot{\mu} = \alpha_1 \kappa_1^{(1)} + \alpha_2 \kappa_0^{(1)} \end{cases} \tag{7.38}$$

with unknowns $\alpha_1, \alpha_2 \in \mathbb{R}$. $v^{(0)}$, $\kappa_0^{(1)}$ and $\kappa_1^{(1)}$ are constructed by the initial steps of algorithm 7.6. Since $g(\psi) = \psi$ and $g(\psi^{(b)}) = \psi^{(b)}$, equation (7.37) implies $g(\dot{\psi}) = \dot{\psi}$ as well. This result is combined with a least-squares method to yield the following formulas for α_1 and α_2:

$$\begin{aligned} &\alpha_1 = -\langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(2)}) - \chi^{(2)} \rangle, \\ &\alpha_2 = \langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(1)}) - \chi^{(1)} \rangle, \\ &\text{with } \chi^{(1)} = \phi_1 + \kappa_1^{(1)} v^{(0)}, \qquad \chi^{(2)} = \phi_2 + \kappa_0^{(1)} v^{(0)}. \end{aligned} \tag{7.39}$$

Substitution of α_1 and α_2 in (7.38) gives the desired tangent direction to the g-symmetric solution curve.

The unknowns α_1 and α_2 are calculated by application of the formula (7.39). No additional linear systems or applications of derivatives of $\mathcal{F}$ are required, strongly reducing the amount of computational work compared to further execution of algorithm 7.6.

7.5.2 Presence of continuous symmetry

The presence of continuous symmetries complicates the application of the equivariant branching lemma. We again consider a solution curve $(\psi(s), \mu(s))$ of $\mathcal{F}$, this time the function is invariant under both the actions of a finite-dimensional Lie group H (representing the continuous symmetry) and the group D_p, with again $p \geq 4$. Let $(\psi(s_0), \mu(s_0)) = (\psi^{(b)}, \mu^{(b)})$ be a branch point, and assume the eigenvalue associated with this point crosses the origin with non-zero speed.

Though the branch point might not be immediately invariant under an action $\varphi : G' \times \mathbb{C}^n \to \mathbb{C}^n$ of a subgroup $G' \subseteq D_p$, it is possible that a different

member of its solution family is:

$$\exists h_{G'} \in H : \forall g \in G' : g(h_{G'}(\psi^{(b)})) = h_{G'}(\psi^{(b)}).$$

In this case the branch point is invariant under an alternative action φ', defined by

$$\varphi' : G' \times \mathbb{C}^n \to \mathbb{C}^n : (g, \psi) \to \varphi'(g, \psi) = h_{G'}^{-1}(g(h_{G'}(\psi))).$$

It is still possible to apply the equivariant branching lemma, granted we can find $h_{G'} \in H$ that transforms $\psi^{(b)}$ to a representative that is invariant under the original action φ of G'. In practice this element is determined by application of a least-squares method. Taking an element $g \in G'$ for which $\varphi(g, \psi^{(b)}) \neq \psi^{(b)}$, $h_{G'}$ is chosen such that

$$h_{G'} = \arg\min_{h \in H} \left\| g(h(\psi^{(b)})) - h(\psi^{(b)}) \right\|^2.$$

For several cases it is possible to derive an exact formula for $h_{G'}$, an example is given below for the Ginzburg-Landau equation, where $H = S^1$ represents the phase symmetry.

Example 7.10. Consider the Ginzburg-Landau equation, described in section 2.5, applied to a general D_p-symmetric ($p \geq 4$) material and magnetic field. The function $\mathcal{F}$ is given by (2.20), it is invariant under the actions φ of $S^1 \times D_p$, defined in proposition 2.3. We denote $D_p = \langle \tau_\omega, \sigma \rangle$, with $\omega = 2\pi/p$, and $S^1 = \{\theta\eta \mid \eta \in [-\pi, \pi]\}$ as in this proposition.

Consider a branch point $(\psi^{(b)}, \mu^{(b)})$ of the equation and a subgroup $G' \subseteq D_p$ such that $\sigma \in G'$ and

$$\exists \eta \in [-\pi, \pi] : \forall g \in G' : \varphi(g, \theta_\eta(\psi^{(b)})) = \theta_\eta(\psi^{(b)}).$$

If $\psi^{(b)}$ is not invariant under the actions φ of G', a G'-invariant representative $\psi'^{(b)}$ is constructed by $\psi'^{(b)} = \theta_\eta(\psi^{(b)})$. The value $\eta \in [-\pi, \pi]$ that sets the element θ_η is calculated by

$$\eta = \frac{1}{2} \arctan \left(\frac{\mathfrak{Re}(\psi^{(b)})^T \mathfrak{Im}(\sigma(\psi^{(b)})) - \mathfrak{Im}(\psi^{(b)})^T \mathfrak{Re}(\sigma(\psi^{(b)}))}{\mathfrak{Re}(\psi^{(b)})^T \mathfrak{Re}(\sigma(\psi^{(b)})) + \mathfrak{Im}(\psi^{(b)})^T \mathfrak{Im}(\sigma(\psi^{(b)}))} \right).$$

◇

Denote $(\psi'^{(b)}, \mu'^{(b)}) = (h_{G'}(\psi^{(b)}), \mu^{(b)})$, the representative of the branch point invariant under the original actions φ. The equivariant branching lemma is now applied as in section 7.5.1: each axial isotropy subgroup $G'' \subseteq G'$ implies a solution curve emanating from $(\psi'^{(b)}, \mu'^{(b)})$, with precisely G'' as its group invariance. We choose such a subgroup and an element $g \in G''$ such that at least one vector among $h_{G'}(\phi_1)$, $h_{G'}(\phi_2)$ and $h_{G'}(v^{(0)})$ is not invariant under g.

The tangent direction to the G''-symmetric solution curve emerging from $(\psi'^{(b)}, \mu'^{(b)})$ is given by $(h_{G'}(\dot{\psi}), \dot{\mu})$, with (see (7.7),(7.21) and (7.24))

$$\begin{cases} h_{G'}(\dot{\psi}) = \alpha_1 \left(h_{G'}(\phi_1) + \kappa_1^{(1)} h_{G'}(v^{(0)}) \right) + \alpha_2 \left(h_{G'}(\phi_2) + \kappa_0^{(1)} h_{G'}(v^{(0)}) \right), \\ \dot{\mu} = \alpha_1 \kappa_1^{(1)} + \alpha_2 \kappa_0^{(1)}. \end{cases} \tag{7.40}$$

$v^{(0)}$, $\kappa_0^{(1)}$ and $\kappa_1^{(1)}$ are again constructed by the initial steps of algorithm 7.6. A least-squares formula for the remaining unknowns α_1 and α_2 is derived similar as to section 7.5.1:

$$\begin{aligned}
&\alpha_1 = -\langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(2)}) - \chi^{(2)} \rangle, \\
&\alpha_2 = \langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(1)}) - \chi^{(1)} \rangle, \\
&\text{with } \chi^{(1)} = h_{G'}(\phi_1) + \kappa_1^{(1)} h_{G'}(v^{(0)}), \qquad \chi^{(2)} = h_{G'}(\phi_2) + \kappa_0^{(1)} h_{G'}(v^{(0)}).
\end{aligned} \tag{7.41}$$

Substitution of α_1 and α_2 in (7.40) yields the desired tangent direction, to the G''-symmetric solution curve emerging from $(\psi'^{(b)}, \mu'^{(b)})$. The same values are used to construct a tangent direction to a curve emanating from $(\psi^{(b)}, \mu^{(b)})$ by application of (7.38).

Again no additional linear systems or applications of derivatives of $\mathcal{F}$ are required when calculating α_1 and α_2, strongly reducing the amount of computational work.

Pseudo-code for the construction of tangent directions, based on the equivariant branching lemma, is provided by algorithm 7.6 (page 234). Given elements of axial isotropy subgroups, the algorithm constructs directions that are invariant under these symmetries. First the initial steps of algorithm 7.6 are executed to calculate $v^{(0)}$, $\kappa_0^{(1)}$ and $\kappa_1^{(1)}$, afterwards (7.39) or (7.41) is applied to determine α_1 and α_2. The directions are then constructed by application of (7.38), and they are normalized. If required, an adjusted representative for the branch point is determined before the calculation of α_1 and α_2.

Algorithm 7.6 is part of algorithm 7.1 (page 230), where it can be executed as an alternative for algorithm 7.5 (page 233). It was used in e.g. sections 9.5.1 and 9.5.3, where results for the Ginzburg-Landau equation applied to respectively a square- and pentagon-shaped material are discussed. For each branch point $(\psi^{(b)}, \mu^{(b)})$ with $\dim\left(\hat{\mathcal{N}}\left(\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})\right)\right) = 2$ it was able to construct the required tangent directions.

7.6 Appendices

7.6.1 Derivation of algorithm 7.6

A detailed derivation of algorithm 7.6 is provided in the current section. The derivation requires lemma 7.11.

Lemma 7.11. Consider a system of equations with unknowns α_1 and α_2:

$$\begin{cases} \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i + b\alpha_1 + e\alpha_2 = 0, \\ \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} c_i + d\alpha_2 = 0 \end{cases} \tag{7.42}$$

with $k \geq 2$, $a_0, \ldots, a_k$, $c_0, \ldots, c_k$, $e \in \mathbb{R}$ and $b, d \in \mathbb{R}_0$. This system contains non-isolated solutions if and only if the relations

$$\begin{aligned} & da_0 = ec_0, \\ & c_k = 0, \\ & \forall i = 1 \ldots k : da_i = bc_{i-1} + ec_i \end{aligned}$$

hold. Moreover, in this case the system has a single isolated solution, given by $(\alpha_1, \alpha_2) = (0, 0)$.

Proof.
Step 1: Suppose that the system of equations contains non-isolated solutions. Since this system can be regarded as the intersection of two algebraic curves, Cramer's theorem [24] can be applied to show that the curves must be degenerate. The degeneracy implies

$$\begin{aligned} & \exists 1 \leq l \leq k-1, f_i^{(m)} (m = 0 \ldots l, i = 0 \ldots m), g_i^{(m)} (m = 0 \ldots k-l, i = 0 \ldots m), \\ & \quad h_i^{(m)} (m = 0 \ldots k-l, i = 0 \ldots m) : \\ & \sum_{i=0}^{k} a_i \alpha_1^i \alpha_2^{k-i} + b\alpha_1 + e\alpha_2 = \left(\sum_{m=0}^{l} \sum_{i=0}^{m} f_i^{(m)} \alpha_1^i \alpha_2^{m-i} \right) \left(\sum_{m=0}^{k-l} \sum_{i=0}^{m} g_i^{(m)} \alpha_1^i \alpha_2^{m-i} \right), \end{aligned} \tag{7.43}$$

$$\sum_{i=0}^{k} c_i \alpha_1^i \alpha_2^{k-i} + d\alpha_2 = \left(\sum_{m=0}^{l} \sum_{i=0}^{m} f_i^{(m)} \alpha_1^i \alpha_2^{m-i} \right) \left(\sum_{m=0}^{k-l} \sum_{i=0}^{m} h_i^{(m)} \alpha_1^i \alpha_2^{m-i} \right). \tag{7.44}$$

Comparing the coefficients of α_1 and α_2 in (7.43) and (7.44), it follows that $f_0^{(0)} \neq 0$. Similar comparisons of other coefficients can be combined with an induction argument to prove the statements

$$\forall m = 0 \ldots k-l : dg_0^{(m)} = eh_0^{(m)}, \tag{7.45}$$

$$\forall m = 0 \ldots k-l : h_m^{(m)} = 0, \tag{7.46}$$

$$\forall m = 1 \ldots k-l, i = 1 \ldots m : dg_i^{(m)} = bh_{i-1}^{(m)} + eh_i^{(m)}. \tag{7.47}$$

Using (7.45), we get

$$da_0 = df_0^{(l)} g_0^{(k-l)} = ef_0^{(l)} h_0^{(k-l)} = ec_0.$$

Next, (7.46) implies that

$$c_k = f_l^{(l)} h_{k-l}^{(k-l)} = 0.$$

For ease of notation, define

$$\forall l+1 \leq i \leq k : f_i^{(l)} = 0,$$
$$\forall k-l+1 \leq i \leq k : g_i^{(k-l)} = h_i^{(k-l)} = 0.$$

Since $h_{k-l}^{(k-l)} = 0$, we have

$$\forall i = 1 \dots k : dg_i^{(k-l)} = bh_{i-1}^{(k-l)} + eh_i^{(k-l)}.$$

Together with (7.45),(7.46) and (7.47), this implies ($\forall i = 1 \dots k$)

$$da_i = d\sum_{j=0}^{i} f_j^{(l)} g_{i-j}^{(k-l)} = b\sum_{j=0}^{i-1} f_j^{(l)} h_{i-1-j}^{(k-l)} + e\sum_{j=0}^{i} f_j^{(l)} h_{i-j}^{(k-l)} = bc_{i-1} + ec_i.$$

Step 2: We have yet to show that

$da_0 = ec_0, c_k = 0, \forall i = 1 \dots k : da_i = bc_{i-1} + ec_i$
$\Rightarrow$ The system of equations (7.42) contains non-isolated solutions and a single isolated solution given by $(\alpha_1, \alpha_2) = (0, 0)$.

We first rewrite the equations for the algebraic curves. The one for the first curve becomes

$$\sum_{i=0}^{k} a_i \alpha_1^i \alpha_2^{k-i} + b\alpha_1 + e\alpha_2 = 0 \iff \left(\sum_{i=0}^{k-1} c_i \alpha_1^i \alpha_2^{k-1-i} + d\right)(b\alpha_1 + e\alpha_2) = 0.$$

The equation for the second curve can be rewritten as

$$\sum_{i=0}^{k} c_i \alpha_1^i \alpha_2^{k-i} + d\alpha_2 = 0 \iff \left(\sum_{i=0}^{k-1} c_i \alpha_1^i \alpha_2^{k-1-i} + d\right)\alpha_2 = 0.$$

Solutions (α_1, α_2) of system (7.42) satisfy

$$b\alpha_1 + e\alpha_2 = 0 \text{ and } \alpha_2 = 0 \text{ or } \sum_{i=0}^{k-1} c_i \alpha_1^i \alpha_2^{k-1-i} + d = 0.$$

The equation $\sum_{i=0}^{k-1} c_i \alpha_1^i \alpha_2^{k-1-i} + d = 0$ again describes an algebraic curve. The solutions that satisfy this equation hence form a continuous curve and are non-isolated. The solution for which

$$b\alpha_1 + e\alpha_2 = 0 \text{ and } \alpha_2 = 0$$

satisfies $(\alpha_1, \alpha_2) = (0, 0)$. This solution is isolated since $(\alpha_1, \alpha_2) = (0, 0)$ does not belong to the curve described by $\sum_{i=0}^{k-1} c_i \alpha_1^i \alpha_2^{k-1-i} + d = 0$ (because $d \neq 0$). □

The three equations

$$\mathcal{F}_{\psi}^{(b)} w_k + \mathcal{F}_{\mu}^{(b)} \beta_k = -r_k, \tag{7.48}$$

$$\forall j = 1, 2 : \langle \phi_j, w_k \rangle_{\mathbb{R}} = 0, \tag{7.49}$$

$$\forall j = 1, 2 : \langle \breve{\phi}_j, r_k \rangle_{\mathbb{R}} = 0, \tag{7.50}$$

with $k = 1, \ldots, l$, are crucial for the derivation of algorithm 7.6. These equations were derived in section 7.3.1. The terms w_k and β_k appear in the order l Taylor expansion of $\psi(s)$ and $\mu(s)$ (see (7.8)). ϕ_1 and ϕ_2 are null vectors of $\mathcal{F}_{\psi}^{(b)}$, $\breve{\phi}_1$ and $\breve{\phi}_2$ of $\mathcal{F}_{\psi}^{(b)*}$. The terms r_k are given by the formula ($\forall k = 1, \ldots, l$)

$$r_k = \sum_{j=2}^{k} \frac{1}{j!} \sum_{\substack{(k_1,\ldots,k_j) \\ \in \mathcal{K}_j^{(k)}}} \mathcal{D}_{\Psi}^{j} \mathcal{F}^{(b)} \begin{pmatrix} x_{k_1} \\ \beta_{k_1} \end{pmatrix} \begin{pmatrix} x_{k_2} \\ \beta_{k_2} \end{pmatrix} \ldots \begin{pmatrix} x_{k_j} \\ \beta_{k_j} \end{pmatrix} \tag{7.51}$$

$$\text{with } x_1 = \sum_{i=1}^{2} \alpha_i \phi_i + w_1 \qquad \forall p = 2, \ldots, k-1 : x_p = w_p.$$

The terms $v^{(0)}$, t_1, t_2, t_0, $b^{(1)}$, $b^{(2)}$, $y_i^{(k)}$, $a_i^{(k,1)}$, $a_i^{(k,2)}$, $\kappa_i^{(k)}$, $z_i^{(k)}$ and $q_i^{(k)}$ ($k = 1 \ldots l, i = 0 \ldots k$) that appear in the derivation are defined as in algorithm 7.6.

derivation of algorithm 7.6. **Initial**: Together with $r_1 = 0$, equation (7.48) implies

$$\mathcal{F}_{\psi}^{(b)} w_1 = -\mathcal{F}_{\mu}^{(b)} \beta_1.$$

Define $v^{(0)}$ as

$$\mathcal{F}_{\psi}^{(b)} v^{(0)} = -\mathcal{F}_{\mu}^{(b)} \qquad v^{(0)} \in \operatorname{im}\left(\mathcal{F}_{\psi}^{(b)*}\right),$$

this leads to the following expressions for w_1 and x_1:

$$w_1 = \beta_1 v^{(0)}, \tag{7.52}$$

$$x_1 = \alpha_1 \phi_1 + \alpha_2 \phi_2 + \beta_1 v^{(0)}. \tag{7.53}$$

Note that the condition $v^{(0)} \in \operatorname{im}\left(\mathcal{F}_{\psi}^{(b)*}\right)$ is necessary to satisfy (7.49). Formula (7.51) for $k = 2$ yields

$$r_2 = \frac{1}{2} \mathcal{D}_{\Psi}^{2} \mathcal{F}^{(b)} \begin{pmatrix} x_1 \\ \beta_1 \end{pmatrix} \begin{pmatrix} x_1 \\ \beta_1 \end{pmatrix}.$$

Substitution of x_1 and bilinearity of the Hessian lead to the expression

$$r_2 = \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} y_i^{(2)} + \beta_1 \sum_{i=1}^{2} \alpha_i t_i + \beta_1^2 t_0, \tag{7.54}$$

with $y_0^{(2)}$, $y_1^{(2)}$, $y_2^{(2)}$, t_1, t_2 and t_0 defined in algorithm 7.6. Substitution of this expression in equation (7.50) leads to the system

$$\begin{cases} \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} \langle \breve{\phi}_1, y_i^{(2)} \rangle + \beta_1 \sum_{i=1}^{2} \alpha_i \langle \breve{\phi}_1, t_i \rangle + \beta_1^2 \langle \breve{\phi}_1, t_0 \rangle = 0, \\ \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} \langle \breve{\phi}_2, y_i^{(2)} \rangle + \beta_1 \sum_{i=1}^{2} \alpha_i \langle \breve{\phi}_2, t_i \rangle + \beta_1^2 \langle \breve{\phi}_2, t_0 \rangle = 0. \end{cases} \tag{7.55}$$

Together with the fundamental theorem of linear algebra [103], assumptions (7.15), (7.16) and (7.17) allow us to choose $\breve{\phi}_1$ and $\breve{\phi}_2$ such that

$$\begin{aligned} &\langle \breve{\phi}_1, t_1 \rangle \neq 0, && && \langle \breve{\phi}_1, t_0 \rangle = 0, \\ &\langle \breve{\phi}_2, t_1 \rangle = 0, && \langle \breve{\phi}_2, t_2 \rangle \neq 0, && \langle \breve{\phi}_2, t_0 \rangle = 0. \end{aligned}$$

With these choices, (7.55) becomes

$$\begin{cases} \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} a_i^{(2,1)} + \beta_1 \alpha_1 b^{(1)} + \beta_1 \alpha_2 b^{(3)} = 0, \\ \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} a_i^{(2,2)} + \beta_1 \alpha_2 b^{(2)} = 0, \end{cases} \tag{7.56}$$

where $b^{(1)}$, $b^{(2)}$, $b^{(3)}$, $a_0^{(2,1)}$, $a_1^{(2,1)}$, $a_2^{(2,1)}$, $a_0^{(2,2)}$, $a_1^{(2,2)}$ and $a_2^{(2,2)}$ are defined as in algorithm 7.6. Lemma 7.11 can now be applied to check this system for non-isolated solutions. If only isolated solutions exist, the algorithm is stopped. In the other case, we have

$$b^{(2)} a_0^{(2,1)} = b^{(3)} a_0^{(2,2)}, \quad a_2^{(2,2)} = 0 \quad \forall i = 1, 2 : b^{(2)} a_i^{(2,1)} = b^{(1)} a_{i-1}^{(2,2)} + b^{(3)} a_i^{(2,2)}$$

and continue by solving (7.56) for β_1:

$$\sum_{i=0}^{1} \alpha_1^i \alpha_2^{1-i} a_i^{(2,2)} + \beta_1 b^{(2)} = 0 \iff \beta_1 = \sum_{i=0}^{1} \alpha_1^i \alpha_2^{1-i} \kappa_i^{(1)},$$

with $\kappa_0^{(1)}$ and $\kappa_1^{(1)}$ defined in algorithm 7.6. Substitution of β_1 in (7.54) and (7.53) eventually leads to the following expressions for r_2 and x_1:

$$r_2 = \sum_{i=0}^{2} \alpha_1^i \alpha_2^{2-i} z_i^{(2)}, \qquad x_1 = \sum_{i=0}^{1} \alpha_1^i \alpha_2^{1-i} q_i^{(1)},$$

with $z_0^{(2)}$, $z_1^{(2)}$, $z_2^{(2)}$, $q_0^{(1)}$, $q_1^{(1)}$ defined in algorithm 7.6.

Iteration: Let $k \geq 3$ and assume terms $z_i^{(k-1)} \in \mathbb{C}^n$ (for $i = 0 \ldots k-1$), $q_i^{(j)} \in \mathbb{C}^n$ (for $j = 1 \ldots k-2, i = 0 \ldots j$) and $\kappa_i^{(j)} \in \mathbb{R}$ (for $j = 1 \ldots k-2, i = 0 \ldots j$) have been derived such that

$$r_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} z_i^{(k-1)}, \tag{7.57}$$

$$\forall j = 1 \ldots k-2 : x_j = \sum_{i=0}^{j} \alpha_1^i \alpha_2^{j-i} q_i^{(j)}, \tag{7.58}$$

$$\forall j = 1 \ldots k-2 : \beta_j = \sum_{i=0}^{j} \alpha_1^i \alpha_2^{j-i} \kappa_i^{(j)}. \tag{7.59}$$

Equation (7.48) implies

$$\mathcal{F}_\psi^{(b)} w_{k-1} = -\sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} z_i^{(k-1)} - \mathcal{F}_\mu^{(b)} \beta_{k-1}.$$

Define $v_0^{(k-1)} \dots v_{k-1}^{(k-1)}$ as

$$\forall i = 0 \dots k-1 : \mathcal{F}_\psi^{(b)} v_i^{(k-1)} = -z_i^{(k-1)} \qquad v_i^{(k-1)} \in \operatorname{im}\left(\mathcal{F}_\psi^{(b)*}\right),$$

this leads to the following expression for w_{k-1} $(= x_{k-1})$:

$$x_{k-1} = w_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} v_i^{(k-1)} + \beta_{k-1} v^{(0)}. \tag{7.60}$$

Rewriting formula (7.51) after substitution of (7.58), (7.59) and (7.60), and defining the terms $y_0^{(k)}, \dots, y_k^{(k)}$ as in algorithm 7.6, one can show that

$$\begin{aligned} r_k \;=\; & \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} y_i^{(k)} + \beta_{k-1}\alpha_1 \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_1^{(1)} \\ \kappa_1^{(1)} \end{pmatrix} \begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix} \\ & + \beta_{k-1}\alpha_2 \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_0^{(1)} \\ \kappa_0^{(1)} \end{pmatrix} \begin{pmatrix} v^{(0)} \\ 1 \end{pmatrix}. \end{aligned}$$

The coefficients of $\beta_{k-1}\alpha_1$ and $\beta_{k-1}\alpha_2$ can be rewritten as $t_1 + 2\kappa_1^{(1)} t_0$ and $t_2 + 2\kappa_0^{(1)} t_0$. This leads to the following expression for r_k:

$$r_k = \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} y_i^{(k)} + \beta_{k-1}\alpha_1 \left(t_1 + 2\kappa_1^{(1)} t_0\right) + \beta_{k-1}\alpha_2 \left(t_2 + 2\kappa_0^{(1)} t_0\right). \tag{7.61}$$

Substitution of this expression in equation (7.50) leads to the system

$$\begin{cases} \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} \langle \breve{\phi}_1, y_i^{(k)} \rangle + \beta_{k-1}\alpha_1 \left(\langle \breve{\phi}_1, t_1 \rangle + 2\kappa_1^{(1)} \langle \breve{\phi}_1, t_0 \rangle \right) \\ \qquad\qquad + \beta_{k-1}\alpha_2 \left(\langle \breve{\phi}_1, t_2 \rangle + 2\kappa_0^{(1)} \langle \breve{\phi}_1, t_0 \rangle \right) = 0, \\ \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} \langle \breve{\phi}_2, y_i^{(k)} \rangle + \beta_{k-1}\alpha_1 \left(\langle \breve{\phi}_2, t_1 \rangle + 2\kappa_1^{(1)} \langle \breve{\phi}_2, t_0 \rangle \right) \\ \qquad\qquad + \beta_{k-1}\alpha_2 \left(\langle \breve{\phi}_2, t_2 \rangle + 2\kappa_0^{(1)} \langle \breve{\phi}_2, t_0 \rangle \right) = 0. \end{cases} \tag{7.62}$$

Using

$$\begin{aligned} & b^{(1)} = \langle \breve{\phi}_1, t_1 \rangle \neq 0, && b^{(3)} = \langle \breve{\phi}_1, t_2 \rangle, && \langle \breve{\phi}_1, t_0 \rangle = 0, \\ & \langle \breve{\phi}_2, t_1 \rangle = 0, && b^{(2)} = \langle \breve{\phi}_2, t_2 \rangle \neq 0, && \langle \breve{\phi}_2, t_0 \rangle = 0, \end{aligned}$$

equation (7.62) becomes

$$\begin{cases} \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,1)} + \beta_{k-1}\alpha_1 b^{(1)} + \beta_{k-1}\alpha_2 b^{(3)} = 0, \\ \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} a_i^{(k,2)} + \beta_{k-1}\alpha_2 b^{(2)} = 0, \end{cases} \tag{7.63}$$

where $a_0^{(k,1)}, \dots, a_k^{(k,1)}$ and $a_0^{(k,2)}, \dots, a_k^{(k,2)}$ are defined as in algorithm 7.6. Lemma 7.11 can again be applied to check this system for non-isolated solutions. If only isolated solutions exist, the algorithm is stopped. In the other case, we have

$$\begin{aligned} & b^{(2)} a_0^{(k,1)} = b^{(3)} a_0^{(k,2)}, \quad a_k^{(k,2)} = 0, \\ & \forall i = 1, \dots, k : b^{(2)} a_i^{(k,1)} = b^{(1)} a_{i-1}^{(k,2)} + b^{(3)} a_i^{(k,2)} \end{aligned}$$

and continue by solving (7.63) for β_{k-1}:

$$\sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} a_i^{(k,2)} + \beta_{k-1} b^{(2)} = 0 \iff \beta_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} \kappa_i^{(k-1)},$$

with $\kappa_0^{(k-1)}, \ldots, \kappa_{k-1}^{(k-1)}$ defined in algorithm 7.6. Substitution of β_{k-1} in (7.61) and (7.60) eventually leads to the following expressions for r_k and x_{k-1}:

$$r_k = \sum_{i=0}^{k} \alpha_1^i \alpha_2^{k-i} z_i^{(k)}, \qquad x_{k-1} = w_{k-1} = \sum_{i=0}^{k-1} \alpha_1^i \alpha_2^{k-1-i} q_i^{(k-1)},$$

with $z_0^{(k)}, \ldots, z_k^{(k)}$ and $q_0^{(k-1)}, \ldots, q_{k-1}^{(k-1)}$ defined in algorithm 7.6. With the expressions for r_k, x_{k-1} and β_{k-1} now available, the iteration can be continued for $k \to k+1$. □

7.6.2 Pseudo-code

Main algorithm

Algorithm 7.1 BifurcationAnalysis

Input functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, choice *ConstructDim*3 of algorithm for construction in 3-dimensional kernel case, inner product $\langle \cdot, \cdot \rangle$, (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$, (approximate) direction of known branch $t^{(0)} \in \mathbb{C}^n \times \mathbb{R}$, list $G = [g_1, \ldots, g_k]$ of group actions (with $g_1 = e$ the identity element) representing discrete symmetry, set H of group actions representing continuous symmetry.

Output Type of bifurcation *type* $\in$ {'none', 'turning point', 'branch point'} and matrix T' containing approximate tangent directions to curves emanating from $(\psi^{(b)}, \mu^{(b)})$.

Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
Set $G = [e]$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Set $H = \{e\}$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Calculate $(dim, v^{(0)}, W, \breve{U})$ by executing IdentifyBranchPoint (algorithm 7.2) with given $\mathcal{F}$, P, $\langle \cdot, \cdot \rangle$ and $(\psi^{(b)}, \mu^{(b)})$
if $dim = 0$ **then**
    type = 'none'
    Set T' an empty $n + 1 \times 0$ matrix
else if $dim = 1$ **then**
    type = 'turning point'
    Set T' an empty $n + 1 \times 0$ matrix
else if $dim = 2$ **then**
    type = 'branch point'
    Calculate T by executing ABE (algorithm 7.4) with $\phi = W$, $\breve{\phi} = \breve{U}$ and given $\mathcal{F}$, $\langle \cdot, \cdot \rangle$, $v^{(0)}$ and $(\psi^{(b)}, \mu^{(b)})$
else if $dim = 3$ **then**
    type = 'branch point'
    if *ConstructDim*3 is the algorithm LS **then**
        Calculate T by executing LS (algorithm 7.5) with $\phi_1 = W_1$, $\phi_2 = W_2$, $\breve{\phi}_1 = \breve{U}_1$, $\breve{\phi}_2 = \breve{U}_2$ and given $\mathcal{F}$, P, $\langle \cdot, \cdot \rangle$, $v^{(0)}$ and $(\psi^{(b)}, \mu^{(b)})$
    else
        Calculate T by executing EBL (algorithm 7.6) with $\phi_1 = W_1$, $\phi_2 = W_2$, $\breve{\phi}_1 = \breve{U}_1$, $\breve{\phi}_2 = \breve{U}_2$ and given $\mathcal{F}$, $\langle \cdot, \cdot \rangle$, $v^{(0)}$, $(\psi^{(b)}, \mu^{(b)})$, G and H
    end if
end if
if $dim \geq 2$ **then**
    Calculate T' by executing RemoveDirections (algorithm 7.3) with given $t^{(0)}$, T and G
end if
Return *type*, T'

Branch point identification

Algorithm 7.2 IdentifyBranchPoint

Input Tolerance $\epsilon_{ortho} \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives $\mathcal{F}_\psi$, $\mathcal{F}_\mu$, inner product $\langle \cdot, \cdot \rangle$, (approximate) bifurcation point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$.

Output Dimension $dim \in \mathbb{N}$ of $\hat{N}(\mathcal{F}_\Psi(\psi^{(b)}, \mu^{(b)}))$, vector $v^{(0)} \in \mathbb{C}^n$, matrix W and $\breve{U}$ with approximate null vectors of respectively $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ and $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})^*$.

Define $\mathcal{F}_\psi$, $\mathcal{F}_\mu$ by (5.2) and (5.4) if not specified
Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
Create deflation matrix $K \in \mathbb{C}^{n \times l_1}$ with l_1 null vectors of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$, induced by continuous symmetry
Calculate (L, W, U) by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$, $\mathcal{P} = P(\psi^{(b)}, \mu^{(b)})$ and given $\langle \cdot, \cdot \rangle$ and K
Set dim equal to the size of matrix L
if $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$ is self-adjoint with respect to $\langle \cdot, \cdot \rangle$ **then**
    $\breve{K} = K$
    $\breve{L} = L$
    $\breve{W} = W$
    $\breve{U} = U$
else
    Create deflation matrix $\breve{K} \in \mathbb{C}^{n \times l_1}$ with l_1 null vectors of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})^*$, induced by continuous symmetry
    Calculate $(\breve{L}, \breve{W}, \breve{U})$ by executing RitzRestart (algorithm 3.3) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})^*$, $K = \breve{K}$ and given $\langle \cdot, \cdot \rangle$
end if
$\breve{K} \leftarrow \begin{pmatrix} \breve{K} & \breve{U} \end{pmatrix}$
$cond = True$
$r = \mathcal{F}_\mu(\psi^{(b)}, \mu^{(b)})$
$r' = \|r\|^{-1} r$
for $j = 1, \ldots, l_1 + dim$ **do**
    $\xi_j = \langle r', \breve{K}_j \rangle$
    if $|\xi_j| > \epsilon_{ortho}$ **then**
        $cond = False$
        Break
    end if
end for
if $cond$ **then**
    $dim \leftarrow dim + 1$
    Calculate $v^{(0)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$, $b = -r$, $\mathcal{P} = P(\psi^{(b)}, \mu^{(b)})$, $K = \breve{K}$ and given $\langle \cdot, \cdot \rangle$
else
    Set $v^{(0)}$ a zero $n \times 1$ matrix
end if
Return dim, $v^{(0)}$, W, $\breve{U}$

Removal of duplicate tangent directions

Algorithm 7.3 RemoveDirections

Input Tolerance $\epsilon_{comp} \in \mathbb{R}_0^+$, (approximate) tangent direction $t^{(0)} \in \mathbb{C}^n$, matrix $T \in \mathbb{C}^{n \times l}$ containing (approximate) tangent directions, list $G = [g_1, \ldots, g_k]$ of group actions (with $g_1 = e$ the identity element) representing discrete symmetry.

Output Copy T' of matrix T, with approximations to $t^{(0)}$ and conjugate directions removed.

Set $G = [e]$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Define $\langle \cdot, \cdot \rangle_{ext} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{R} : ((\psi_1, \mu_1), (\psi_2, \mu_2)) \to \langle \psi_1, \psi_2 \rangle + \mu_1^* \mu_2$
Initialize T' an empty $n + 1 \times 0$ matrix
$l' = 0$
Initialize empty list $I = [\ ]$
for g in G **do**
  for $i = 1, \ldots, l - 1$ **do**
    for $j = i + 1, \ldots, l$ **do**
      if i and j are not contained in I **then**
        if $1 - |\langle T_i, g(T_j) \rangle_{ext}| < \epsilon_{comp}$ **then**
          $I = \begin{bmatrix} I & j \end{bmatrix}$
        end if
      end if
    end for
  end for
end for
for $j = 1, \ldots, l$ **do**
  if j is not contained in I **then**
    $l' \leftarrow l' + 1$
    $T' = \begin{pmatrix} T' & T_j \end{pmatrix}$
  end if
end for
$index = 0$
$d = \infty$
for $j = 1, \ldots, l'$ **do**
  for g in G **do**
    $d' = 1 - |\langle t^{(0)}, g(T'_j) \rangle_{ext}|$
    if $d' < d$ **then**
      $index = j$
      $d = d'$
    end if
  end for
end for
Remove T'_{index} from T'
Return T'

Algebraic branching equation

Algorithm 7.4 ABE

Input function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, partial derivatives $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$, inner product $\langle \cdot, \cdot \rangle$, vectors $v^{(0)}, \phi, \breve{\phi} \in \mathbb{C}^n$, (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$.

Output Matrix T containing approximate tangent directions to curves emanating from $(\psi^{(b)}, \mu^{(b)})$.

1: Define $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$ by (5.6), (5.7) and (5.8) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: $a = \langle \breve{\phi}, \frac{1}{2}\mathcal{F}^{(b)}_{\psi\psi}\phi\phi \rangle$
5: $b = \langle \breve{\phi}, \mathcal{F}^{(b)}_{\psi\psi}\phi v^{(0)} + \mathcal{F}^{(b)}_{\psi\mu}\phi \rangle$
6: $c = \langle \breve{\phi}, \frac{1}{2}\mathcal{F}^{(b)}_{\psi\psi}v^{(0)}v^{(0)} + \mathcal{F}^{(b)}_{\psi\mu}v^{(0)} + \frac{1}{2}\mathcal{F}^{(b)}_{\mu\mu} \rangle$
7: $\alpha_1^{(1)} = \frac{1}{2a}\left(-b + \sqrt{b^2 - 4ac}\right)$
8: $\alpha_1^{(2)} = \frac{1}{2a}\left(-b - \sqrt{b^2 - 4ac}\right)$
9: $\dot{\psi}'^{(1)} = \alpha_1^{(1)}\phi + v^{(0)}$
10: $\dot{\mu}'^{(1)} = 1$
11: $\gamma_1 = \sqrt{\|\dot{\psi}'^{(1)}\|^2 + \dot{\mu}'^{(1)2}}$
12: $\dot{\psi}^{(1)} = \gamma_1^{-1}\dot{\psi}'^{(1)}$
13: $\dot{\mu}^{(1)} = \gamma_1^{-1}\dot{\mu}'^{(1)}$
14: $\dot{\psi}'^{(2)} = \alpha_1^{(2)}\phi + v^{(0)}$
15: $\dot{\mu}'^{(2)} = 1$
16: $\gamma_2 = \sqrt{\|\dot{\psi}'^{(2)}\|^2 + \dot{\mu}'^{(2)2}}$
17: $\dot{\psi}^{(2)} = \gamma_2^{-1}\dot{\psi}'^{(2)}$
18: $\dot{\mu}^{(2)} = \gamma_2^{-1}\dot{\mu}'^{(2)}$
19: $T = \begin{pmatrix} \dot{\psi}^{(1)} & \dot{\psi}^{(2)} \\ \dot{\mu}^{(1)} & \dot{\mu}^{(2)} \end{pmatrix}$
20: Return T

Lyapunov-Schmidt reduction-based tangent construction

Algorithm 7.5 LS

Input functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, inner product $\langle \cdot, \cdot \rangle$, vectors $v^{(0)}, \phi_1, \phi_2, \breve{\phi}_1, \breve{\phi}_2 \in \mathbb{C}^n$, (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$.

Output Matrix T containing approximate tangent directions to curves emanating from $(\psi^{(b)}, \mu^{(b)})$.

1: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
2: Calculate $(k, A^{(1)}, A^{(2)}, B, \kappa_0^{(1)}, \kappa_1^{(1)})$ by executing CalculateCoefficients (algorithm 7.7) with given $\mathcal{F}$, P, $\langle \cdot, \cdot \rangle$, $v^{(0)}$, ϕ_1, ϕ_2, $\breve{\phi}_1$, $\breve{\phi}_2$ and $(\psi^{(b)}, \mu^{(b)})$
3: Calculate S by executing SolveReducedSystem (algorithm 7.8) with given $A^{(1)}$, $A^{(2)}$ and B

Initialize T an empty $n+1 \times 0$ matrix
if $k = 2$ **then**
  for $(\alpha_1, \alpha_2, \beta)$ in S **do**
    $\dot{\mu}' = \beta$
    $\dot{\psi}' = \alpha_1 \phi_1 + \alpha_2 \phi_2 + \beta v^{(0)}$
    $\gamma = \sqrt{\|\dot{\psi}'\|^2 + \dot{\mu}'^2}$
    $\dot{\psi} = \gamma^{-1} \dot{\psi}'$
    $\dot{\mu} = \gamma^{-1} \dot{\mu}'$
    $T \leftarrow \begin{pmatrix} T & \dot{\psi} \\ & \dot{\mu} \end{pmatrix}$
  end for
else
  $\dot{\mu}' = 1$
  $\dot{\psi}' = v^{(0)}$
  $\gamma = \sqrt{\|\dot{\psi}'\|^2 + \dot{\mu}'^2}$
  $\dot{\psi} = \gamma^{-1} \dot{\psi}'$
  $\dot{\mu} = \gamma^{-1} \dot{\mu}'$
  $T \leftarrow \begin{pmatrix} T & \dot{\psi} \\ & \dot{\mu} \end{pmatrix}$
  for $(\alpha_1, \alpha_2, \beta)$ in S **do**
    $\dot{\mu}' = \alpha_1 \kappa_1^{(1)} + \alpha_2 \kappa_0^{(1)}$
    $\dot{\psi}' = \alpha_1 \phi_1 + \alpha_2 \phi_2 + \dot{\mu}' v^{(0)}$
    $\gamma = \sqrt{\|\dot{\psi}'\|^2 + \dot{\mu}'^2}$
    $\dot{\psi} = \gamma^{-1} \dot{\psi}'$
    $\dot{\mu} = \gamma^{-1} \dot{\mu}'$
    $T \leftarrow \begin{pmatrix} T & \dot{\psi} \\ & \dot{\mu} \end{pmatrix}$
  end for
end if
Return T

Equivariant branching lemma-based tangent construction

Algorithm 7.6 EBL

Input Tolerance $\epsilon_{comp} \in \mathbb{R}_0^+$, function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, partial derivatives $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$, inner product $\langle \cdot, \cdot \rangle$, vectors $v^{(0)}, \phi_1, \phi_2, \breve{\phi}_1, \breve{\phi}_2 \in \mathbb{C}^n$, (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$, list $G = [g_1, \ldots, g_k]$ of group actions (with $g_1 = e$ the identity element) representing discrete symmetry, set H of group actions representing continuous symmetry.

Output Matrix T containing approximate tangent directions to curves emanating from $(\psi^{(b)}, \mu^{(b)})$.

Define $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$ by (5.6), (5.7) and (5.8) if not specified

Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
Set $H = \{e\}$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Redefine $\breve{\phi}_1$ and $\breve{\phi}_2$ such that (7.20) holds if necessary
$y_0^{(2)} = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)}\phi_2\phi_2$
$y_1^{(2)} = \mathcal{F}_{\psi\psi}^{(b)}\phi_1\phi_2$
$y_2^{(2)} = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)}\phi_1\phi_1$
$t_1 = \mathcal{F}_{\psi\psi}^{(b)}\phi_1 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)}\phi_1$
$t_2 = \mathcal{F}_{\psi\psi}^{(b)}\phi_2 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)}\phi_2$
$t_0 = \frac{1}{2}\mathcal{F}_{\psi\psi}^{(b)}v^{(0)}v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)}v^{(0)} + \frac{1}{2}\mathcal{F}_{\mu\mu}^{(b)}$
$b^{(1)} = \langle \breve{\phi}_1, t_1 \rangle$
$b^{(2)} = \langle \breve{\phi}_2, t_2 \rangle$
$b^{(3)} = \langle \breve{\phi}_1, t_2 \rangle$
for $i = 1, 2, 3$ **do**
    $a_i^{(2,1)} = \langle \breve{\phi}_1, y_i^{(2)} \rangle$
    $a_i^{(2,2)} = \langle \breve{\phi}_2, y_i^{(2)} \rangle$
end for
$d = \min(|b^{(2)}a_0^{(2,1)} - b^{(3)}a_0^{(2,2)}|, |a_2^{(2,2)}|)$
$d \leftarrow \min(d, |b^{(1)}a_0^{(2,2)} + b^{(3)}a_1^{(2,2)} - b^{(2)}a_1^{(2,1)}|)$
$d \leftarrow \min(d, |b^{(1)}a_1^{(2,2)} + b^{(3)}a_0^{(2,2)} - b^{(2)}a_0^{(2,1)}|)$
Initialize T an empty $n + 1 \times 0$ matrix
if $d < \epsilon_{comp}$ **then**
    $A^{(1)} = \begin{pmatrix} a_0^{(2,1)} & a_1^{(2,1)} & a_2^{(2,1)} \end{pmatrix}$
    $A^{(2)} = \begin{pmatrix} a_0^{(2,2)} & a_1^{(2,2)} & a_2^{(2,2)} \end{pmatrix}$
    $B = \begin{pmatrix} b^{(1)} & b^{(2)} & b^{(3)} \end{pmatrix}$
    Calculate S by executing SolveReducedSystem (algorithm 7.8) with given $A^{(1)}$, $A^{(2)}$ and B
    for $(\alpha_1, \alpha_2, \beta)$ in S **do**
        $\dot{\mu}' = \beta$
        $\dot{\psi}' = \alpha_1\phi_1 + \alpha_2\phi_2 + \beta v^{(0)}$
        $\gamma = \sqrt{\|\dot{\psi}'\|^2 + \dot{\mu}'^2}$
        $\dot{\psi} = \gamma^{-1}\dot{\psi}'$
        $\dot{\mu} = \gamma^{-1}\dot{\mu}'$
        $T \leftarrow \begin{pmatrix} T & \dot{\psi} \\ & \dot{\mu} \end{pmatrix}$
    end for
else
    $\kappa_0^{(1)} = -\frac{a_0^{(2,2)}}{b^{(2)}}$
    $\kappa_1^{(1)} = -\frac{a_1^{(2,2)}}{b^{(2)}}$
    if $g(\psi^{(b)}) = \psi^{(b)}$ for each action g in G **then**
        $h_G = e$
    else
        Choose $g \in G$ such that $g(\psi^{(b)}) \neq \psi^{(b)}$

42: $h_G = \arg\min_{h \in H} \|g(h(\psi^{(b)})) - h(\psi^{(b)})\|^2$
43: **end if**
44: $\psi'^{(b)} = h_G(\psi^{(b)})$
45: $\chi^{(1)} = h_G(\phi_1) + \kappa_1^{(1)} h_G(v^{(0)})$
46: $\chi^{(2)} = h_G(\phi_2) + \kappa_0^{(1)} h_G(v^{(0)})$
47: **for** axial subgroups $G'' \subset G$ **do**
48: Choose $g \in G''$ such that $g(\chi^{(1)}) \neq \chi^{(1)}$ or $g(\chi^{(2)}) \neq \chi^{(2)}$
49: $\alpha_1 = -\langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(2)}) - \chi^{(2)} \rangle$
50: $\alpha_2 = \langle g(\chi^{(1)}) - \chi^{(1)}, g(\chi^{(1)}) - \chi^{(1)} \rangle$
51: $\dot{\psi}' = \alpha_1(\phi_1 + \kappa_1^{(1)} v^{(0)}) + \alpha_2(\phi_2 + \kappa_0^{(1)} v^{(0)})$
52: $\dot{\mu}' = \alpha_1 \kappa_1^{(1)} + \alpha_2 \kappa_0^{(1)}$
53: $\gamma = \sqrt{\|\dot{\psi}'\|^2 + \dot{\mu}'^2}$
54: $\dot{\psi} = \gamma^{-1} \dot{\psi}'$
55: $\dot{\mu} = \gamma^{-1} \dot{\mu}'$
56: $T \leftarrow \begin{pmatrix} T & \dot{\psi} \\ & \dot{\mu} \end{pmatrix}$
57: **end for**
58: **end if**
59: Return T

Algorithm 7.6

Algorithm 7.7 CalculateCoefficients

Input tolerance $\epsilon_{comp} \in \mathbb{R}_0^+$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, partial derivatives $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$, inner product $\langle \cdot, \cdot \rangle$, vectors $v^{(0)}, \phi_1, \phi_2, \breve{\phi}_1, \breve{\phi}_2 \in \mathbb{C}^n$, (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$.

Output $k \in \mathbb{N}$, matrices $A^{(1)}, A^{(2)} \in \mathbb{R}^{k+1}$ and $B \in \mathbb{R}^3$ containing coefficients for a reduced system of equations (of the form (7.35)), values $\kappa_0^{(1)}, \kappa_1^{(1)} \in \mathbb{R}$.

1: Define $\mathcal{F}_{\psi\psi}$, $\mathcal{F}_{\psi\mu}$, $\mathcal{F}_{\mu\mu}$ by (5.6), (5.7) and (5.8) if not specified
2: Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
3: Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
4: Redefine $\breve{\phi}_1$ and $\breve{\phi}_2$ such that (7.20) holds if necessary
5: $k = 2$
6: $y_0^{(2)} = \frac{1}{2} \mathcal{F}_{\psi\psi}^{(b)} \phi_2 \phi_2$
7: $y_1^{(2)} = \mathcal{F}_{\psi\psi}^{(b)} \phi_1 \phi_2$
8: $y_2^{(2)} = \frac{1}{2} \mathcal{F}_{\psi\psi}^{(b)} \phi_1 \phi_1$
9: $t_1 = \mathcal{F}_{\psi\psi}^{(b)} \phi_1 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} \phi_1$
10: $t_2 = \mathcal{F}_{\psi\psi}^{(b)} \phi_2 v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} \phi_2$
11: $t_0 = \frac{1}{2} \mathcal{F}_{\psi\psi}^{(b)} v^{(0)} v^{(0)} + \mathcal{F}_{\psi\mu}^{(b)} v^{(0)} + \frac{1}{2} \mathcal{F}_{\mu\mu}^{(b)}$
12: $b^{(1)} = \langle \breve{\phi}_1, t_1 \rangle$

$b^{(2)} = \langle \breve{\phi}_2, t_2 \rangle$
$b^{(3)} = \langle \breve{\phi}_1, t_2 \rangle$
for $i = 1, 2, 3$ **do**
  $a_i^{(2,1)} = \langle \breve{\phi}_1, y_i^{(2)} \rangle$
  $a_i^{(2,2)} = \langle \breve{\phi}_2, y_i^{(2)} \rangle$
end for
$d = \min(|b^{(2)} a_0^{(2,1)} - b^{(3)} a_0^{(2,2)}|, |a_2^{(2,2)}|)$
$d \leftarrow \min(d, |b^{(1)} a_0^{(2,2)} + b^{(3)} a_1^{(2,2)} - b^{(2)} a_1^{(2,1)}|)$
$d \leftarrow \min(d, |b^{(1)} a_1^{(2,2)} + b^{(3)} a_0^{(2,2)} - b^{(2)} a_0^{(2,1)}|)$
if $d < \epsilon_{comp}$ **then**
  $reduced = True$
  $\kappa_0^{(1)}$=0
  $\kappa_1^{(1)}$=0
else
  $reduced = False$
  $\kappa_{-1}^{(1)} = 0$
  $\kappa_0^{(1)} = -\frac{a_0^{(2,2)}}{b^{(2)}}$
  $\kappa_1^{(1)} = -\frac{a_1^{(2,2)}}{b^{(2)}}$
  $\kappa_2^{(1)} = 0$
  $z_0^{(2)} = y_0^{(2)} + \kappa_0^{(1)} t_2 + {\kappa_0^{(1)}}^2 t_0$
  $z_1^{(2)} = y_1^{(2)} + \kappa_0^{(1)} t_1 + \kappa_1^{(1)} t_2 + 2\kappa_0^{(1)} \kappa_1^{(1)} t_0$
  $z_2^{(2)} = y_2^{(2)} + \kappa_1^{(1)} t_1 + {\kappa_1^{(1)}}^2 t_0$
  $q_0^{(1)} = \kappa_0^{(1)} v^{(0)} + \phi_2$
  $q_1^{(1)} = \kappa_1^{(1)} v^{(0)} + \phi_1$
end if
Create deflation matrix $\breve{K} \in \mathbb{C}^{n \times l_1}$ with l_1 null vectors of $\mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})^*$, induced by continuous symmetry
$\breve{K} \leftarrow \begin{pmatrix} \breve{K} & \breve{\phi}_1 & \breve{\phi}_2 \end{pmatrix}$
while not *reduced* **do**
  $k \leftarrow k + 1$
  for $i = 0, \ldots, k-1$ **do**
    Calculate $v_i^{(k-1)}$ by executing GMRES (algorithm 3.4) with $\mathcal{A} = \mathcal{F}_\psi(\psi^{(b)}, \mu^{(b)})$, $b = -z_i^{(k-1)}$, $\mathcal{P} = P(\psi^{(b)}, \mu^{(b)})$, $K = \breve{K}$ and given $\langle \cdot, \cdot \rangle$
  end for
  for $i = 0, \ldots, k$ **do**
    Calculate $y_i^{(k)}$ by executing CalculateY (algorithm 7.9) with given $\mathcal{F}$, i and k
    $a_i^{(k,1)} = \langle \breve{\phi}_1, y_i^{(k)} \rangle$
    $a_i^{(k,2)} = \langle \breve{\phi}_2, y_i^{(k)} \rangle$
  end for
  $d = \min(|b^{(2)} a_0^{(k,1)} - b^{(3)} a_0^{(k,2)}|, |a_k^{(k,2)}|)$
  for $i = 1, \ldots, k$ **do**
    $d \leftarrow \min(d, |b^{(1)} a_{i-1}^{(k,2)} + b^{(3)} a_i^{(k,2)} - b^{(2)} a_i^{(k,1)}|)$
  end for

54: **if** $d < \epsilon_{comp}$ **then**
55: $reduced = True$
56: **else**
57: $\kappa_{-1}^{(k-1)} = 0$
58: $\kappa_{k}^{(k-1)} = 0$
59: **for** $i = 0, \ldots, k-1$ **do**
60: $\kappa_i^{(k-1)} = -\frac{a_i^{(k,2)}}{b^{(2)}}$
61: $z_i^{(k)} = y_i^{(k)} + \kappa_{i-1}^{(k-1)} \left(t_1 + 2\kappa_1^{(1)} t_0\right) + \kappa_i^{(k-1)} \left(t_2 + 2\kappa_0^{(1)} t_0\right)$
62: $q_i^{(k-1)} = v_i^{(k-1)} + \kappa_i^{(k-1)} v^{(0)}$
63: **end for**
64: $z_k^{(k)} = y_k^{(k)} + \kappa_{k-1}^{(k-1)} \left(t_1 + 2\kappa_1^{(1)} t_0\right) + \kappa_k^{(k-1)} \left(t_2 + 2\kappa_0^{(1)} t_0\right)$
65: **end if**
66: **end while**
67: $A^{(1)} = \begin{pmatrix} a_0^{(k,1)} & a_1^{(k,1)} & \ldots & a_k^{(k,1)} \end{pmatrix}$
68: $A^{(2)} = \begin{pmatrix} a_0^{(k,2)} & a_1^{(k,2)} & \ldots & a_k^{(k,2)} \end{pmatrix}$
69: $B = \begin{pmatrix} b^{(1)} & b^{(2)} & b^{(3)} \end{pmatrix}$
70: Return k, $A^{(1)}$, ,$A^{(2)}$, B, $\kappa_0^{(1)}$, $\kappa_1^{(1)}$

Solve reduced system of equations

Algorithm 7.8 SolveReducedSystem

Input $m_{rand} \in \mathbb{N}$, tolerance $\epsilon_{comp} \in \mathbb{R}_0^+$, matrices $A^{(1)}, A^{(2)} \in \mathbb{R}^{k+1}$ and $B \in \mathbb{R}^3$ containing coefficients for the reduced system of equations.

Output List S with elements $(\alpha_1, \alpha_2, \beta) \in \mathbb{R}^3$ corresponding to solutions of the reduced system (given by (7.35)).

1: **for** $i = 0, \ldots, k$ **do**
2: $a_i^{(k,1)} = A_i^{(1)}$
3: $a_i^{(k,2)} = A_i^{(2)}$
4: **end for**
5: $b^{(1)} = B_1$
6: $b^{(2)} = B_2$
7: $b^{(3)} = B_3$
8: Define f by (7.6)
9: Define f' by (7.36)
10: Initialize empty list $S = [\,]$
11: **for** $j = 1, \ldots, m_{rand}$ **do**
12: Initialize random values $\alpha_1^{(0)}$, $\alpha_2^{(0)}$ and $\beta^{(0)}$
13: Calculate $(\alpha_1, \alpha_2, \beta)$ by executing NewtonStandard (algorithm 4.1) with $F = f$, $F_x = f'$ and $x^{(0)} = (\alpha_1^{(0)}, \alpha_2^{(0)}, \beta^{(0)})$
14: $new = True$
15: **for** $(\alpha_1', \alpha_2', \beta')$ in S **do**

if $|\alpha_1' - \alpha_1| > \epsilon_{comp}$ and $|\alpha_2' - \alpha_2| > \epsilon_{comp}$ and $|\beta' - \beta| > \epsilon_{comp}$ **then**
  $new \leftarrow False$
  Break
end if
end for
if *new* **then**
  $S = \begin{bmatrix} S & (\alpha_1, \alpha_2, \beta) \end{bmatrix}$
end if
end for
Return S

Calculation of $y_i^{(k)}$ terms in algorithm 7.5

Algorithm 7.9 CalculateY

Input Function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, partial derivatives of $\mathcal{F}$, $i, k \in \mathbb{N}$, vectors $v_{i'}^{(k-1)}, q_{i'}^{(k')} \in \mathbb{C}^n$ and values $\kappa_{i'}^{(k')} \in \mathbb{R}$.

Output Vector $y_i^{(k)} \in \mathbb{C}^n$.

Define partial derivatives of $\mathcal{F}$ by finite difference formulas if not specified
$y_i^{(k)} = \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_0^{(1)} \\ \kappa_0^{(1)} \end{pmatrix} \begin{pmatrix} v_i^{(k-1)} \\ 0 \end{pmatrix} + \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_1^{(1)} \\ \kappa_1^{(1)} \end{pmatrix} \begin{pmatrix} v_{i-1}^{(k-1)} \\ 0 \end{pmatrix}$
for $k_1 = 2, \ldots, k-2$ **do**
  $k_2 = k - k_1$
  for $i_1 = 0, \ldots, \min(k_1, i)$ **do**
    $i_2 = i - i_1$
    if $i_2 \leq k_2$ and $k_1 \leq k_2$ and ($k_1 \neq k_2$ or $i_1 \leq i_2$) **then**
      $\tilde{y} = \mathcal{F}_{\Psi\Psi}^{(b)} \begin{pmatrix} q_{i_1}^{(k_1)} \\ \kappa_{i_1}^{(k_1)} \end{pmatrix} \begin{pmatrix} q_{i_2}^{(k_2)} \\ \kappa_{i_2}^{(k_2)} \end{pmatrix}$
      if $k_1 = k_2$ and $i_1 = i_2$ **then**
        $y_i^{(k)} = y_i^{(k)} + \frac{1}{2}\tilde{y}$
      else
        $y_i^{(k)} = y_i^{(k)} + \tilde{y}$
      end if
    end if
  end for
end for
for $j = 3, \ldots, k$ **do**
  Calculate $\mathcal{K}$ by executing IndicesK (algorithm 7.10) with given k and j
  for K in $\mathcal{K}$ **do**
    Calculate $\mathcal{I}$ by executing IndicesK (algorithm 7.11) with given i, j and K
    for I in $\mathcal{I}$ **do**
      Set c the amount of possible permutations of the set $\{(K_1, I_1), \ldots, (K_j, I_j)\}$

23: $y_i^{(k)} = y_i^{(k)} + \frac{c}{j!}\mathcal{D}_\Psi^j \mathcal{F}^{(b)} \begin{pmatrix} q_{I_1}^{(K_1)} \\ \kappa_{I_1}^{(K_1)} \end{pmatrix} \begin{pmatrix} q_{I_2}^{(K_2)} \\ \kappa_{I_2}^{(K_2)} \end{pmatrix} \cdots \begin{pmatrix} q_{I_j}^{(K_j)} \\ \kappa_{I_j}^{(K_j)} \end{pmatrix}$
24: **end for**
25: **end for**
26: **end for**
27: Return $y_i^{(k)}$

Calculation of the set $\mathcal{K}_k^{(j)}$

Algorithm 7.10 IndicesK

Input $k, j \in \mathbb{N}$.
Output The set $\mathcal{K} = \mathcal{K}_k^{(j)}$.

1: Initialize empty list $\mathcal{K} = \left[\ \right]$
2: Initialize an empty index set $K = \left\{\ \right\}$
3: Set m the length of K
4: **if** $m < j - 1$ **then**
5: **if** $m \neq 0$ **then**
6: $p_0 = K_m$
7: **else**
8: $p_0 = 1$
9: **end if**
10: **for** $p = p_0, \ldots, k - \sum_{l=1}^{m} K_l - j + m + 1$ **do**
11: $K' = K$
12: $K' = \begin{bmatrix} K' & p \end{bmatrix}$
13: Execute the current algorithm starting from line 3, with $K = K'$
14: **end for**
15: **else**
16: $p = k - \sum_{l=1}^{m} K_l$
17: **if** $K_{j-1} \leq p$ **then**
18: $K = \begin{bmatrix} K & p \end{bmatrix}$
19: $\mathcal{K} = \begin{bmatrix} \mathcal{K} & K \end{bmatrix}$
20: **end if**
21: **end if**
22: Return $\mathcal{K}$

Calculation of the set $\mathcal{I}_{k_1,\ldots,k_j}^{(i,j)}$

Algorithm 7.11 IndicesI

Input $i, j \in \mathbb{N}$, element $K = (k_1, \ldots, k_j)$ of (7.11) for given j.
Output The set $\mathcal{I} = \mathcal{I}_{k_1,\ldots,k_j}^{(i,j)}$.

1: Initialize empty list $\mathcal{I} = \left[\ \right]$

2: Initialize an empty index set $I = \{\ \}$
3: Set m the length of I
4: **if** $m < j - 1$ **then**
5: **if** $m \neq 0$ and $K_{m+1} = K_m$ **then**
6: $p_0 = I_m$
7: **else**
8: $p_0 = 0$
9: **end if**
10: **for** $p = p_0, \ldots, \min(K_{m+1}, i - \sum_{l=1}^{m} I_l)$ **do**
11: $I' = I$
12: $I' = \begin{bmatrix} I' & p \end{bmatrix}$
13: Execute the current algorithm starting from line 3, with $I = I'$
14: **end for**
15: **else**
16: $p = i - \sum_{l=1}^{m} I_l$
17: **if** $p \leq K_j$ **then**
18: $I = \begin{bmatrix} I & p \end{bmatrix}$
19: **if** $K_{j-1} \neq K_j$ or $I_{j-1} \leq I_j$ **then**
20: $\mathcal{I} = \begin{bmatrix} \mathcal{I} & I \end{bmatrix}$
21: **end if**
22: **end if**
23: **end if**
24: Return $\mathcal{I}$

CHAPTER 8

Implementation in Python

"One! Two! Five!"

– King Arthur –
Monty Python and the Holy Grail

Chapter highlights:

- We describe the last details required to incorporate automatic exploration techniques in numerical continuation methods.
- We analyse techniques to prevent the recalculation of solution curves.
- An implementation of the algorithm has been made as an extension to the Python package PyNCT [36].
- We discuss the structure and other details of this implementation.

8.1 Introduction

In this chapter we will discuss the last details of automatic exploration. Based on the algorithms described in chapters 6 and 7, a method is established that constructs a solution landscape of interconnected curves from an initial solution. An essential part of the method is the construction of a list that contains elements with information of curves that have yet to be constructed. Recalculation of (conjugates of) curves will be prevented by analysing this list at several points in the algorithm.

An implementation of the automatic exploration method has been made in Python, as an extension of the package PyNCT [36]. Details of this implementation will be discussed in this chapter as well.

8.2 Automatic exploration in practice

In chapter 6 we discussed the approximation of solution curves and its bifurcation points. Given an initial solution $(\psi^{(0)}, \mu^{(0)})$ and direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$, a set

of points $(\psi^{(0)}, \mu^{(0)})$, $(\psi^{(1)}, \mu^{(1)})$, $(\psi^{(2)}, \mu^{(2)})$, ... is constructed by application of the pseudo-arclength continuation method. These points belong to a curve through $(\psi^{(0)}, \mu^{(0)})$ in the given direction. Pseudo-code of this method was provided by algorithm 6.1 (page 193). This algorithm does not only apply the pseudo-arclength continuation method, but also analyses the stability along the curve and approximates its bifurcation points.

Chapter 7 focussed on the requisites for automatic exploration. Given a bifurcation point $(\psi^{(b)}, \mu^{(b)})$ of an approximated solution curve, algorithm 7.1 (page 230), described in the chapter, analyses the type of bifurcation. In the case that $(\psi^{(b)}, \mu^{(b)})$ is a branch point, the tangent directions to curves emanating from this point are approximated as well. The direction to the curve the point was found at is excluded from these approximations.

In the current section we will describe how algorithms 6.1 and 7.1 are combined to generate a method for the construction of a solution landscape of interconnected curves. Pseudo-code for the described method is provided by algorithm 8.1 on page 252 in the appendix.

8.2.1 Construction of a connected solution landscape

Consider a general nonlinear function $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$ with Hermitian partial Jacobian $\mathcal{F}_\psi$. To construct a connected solution landscape for this function, an initial guess $(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)})$ of a solution for $\mathcal{F}(\psi, \mu) = 0$ needs to be provided. This guess is corrected with a Newton algorithm (without block elimination) to a first solution $(\psi^{(0)}, \mu^{(0)})$. Since we do not know whether this first point is a bifurcation, the split Newton method with mixed terms (SNE), described in section 4.8 (pseudo-code given by algorithm 4.7 on page 106), should be used. Note that this method is equivalent to the deflated Newton algorithm (described in section 4.3) if $(\psi^{(0)}, \mu^{(0)})$ is not a bifurcation.

The corrected point $(\psi^{(0)}, \mu^{(0)})$ is used as input in algorithm 6.1 (page 193) to construct an approximation to a first solution curve. Though it is possible to calculate a tangent direction in this point exactly (see e.g. Keller [61]), we will use the direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)}) = (0, 1)$ to initialize the curve instead. Algorithm 6.1 is executed twice: once in the direction $(\dot{\psi}^{(0)}, \dot{\mu}^{(0)})$, and once in $(-\dot{\psi}^{(0)}, -\dot{\mu}^{(0)})$, in order to generate both sides of the curve. Bifurcation points are approximated and stability is analysed in algorithm 6.1 as well.

The approximate bifurcation points are analysed by algorithm 7.1 (page 230): for each point the type is identified by deriving the dimension of the Jacobian's kernel. For branch points, tangent directions to emanating curves are approximated. The direction to the already constructed curve is excluded as described in section 7.2.2.

For each direction $(\dot{\psi}, \dot{\mu})$ leading to a new curve, algorithm 6.1 is reapplied twice. As input for the initial point the corresponding branch point is used, $(\dot{\psi}, \dot{\mu})$ and $(-\dot{\psi}, -\dot{\mu})$ are used as the initial directions. These applications construct an approximation to a different curve, together with its bifurcation points and stability. The process of analysing bifurcations and constructing tangent directions is repeated, until no new bifurcation points are found, or after a maximal amount $m_C \in \mathbb{N}$ of curves have been approximated.

8.2.2 Preventing recalculation of curves

In its current form, the method described in section 8.2.1 allows for single curves to be approximated multiple times. This leads to unnecessary computational work, and a slower overall algorithm. In the current section we address this problem, we identify two causes.

A first possible recalculation occurs when a curve reconnects with itself, an example is given by the circle equation (see section 9.2). To prevent algorithm 6.1 (page 193) from regenerating already calculated points of such curves, a condition function C_{start} should be provided as input, defined by

$$\begin{aligned} &C_{start} : \mathbb{C}^n \times \mathbb{R} \to \{True, False\} : \\ &(\psi^{(k)}, \mu^{(k)}) \to \begin{cases} True & k \geq m_{start} \text{ and } \mu^{(k)} \approx \mu^{(0)} \\ & \text{and } \exists g \in G, h \in H : g(\psi^{(k)}) \approx \psi^{(0)}, \\ False & \text{otherwise,} \end{cases} \end{aligned} \tag{8.1}$$

with given $m_{start} \in \mathbb{N}$, G and H the groups that respectively describe possible discrete and continuous symmetries of $\mathcal{F}$. The function C_{start} is called the start point condition. If no discrete (continuous) symmetry is present, $G = [e]$ ($H = [e]$) with e the identity group action is chosen. The start point condition is defined such that pseudo-arclength continuation in algorithm 6.1 is stopped when the last generated point $(\psi^{(k)}, \mu^{(k)})$ (for a certain $k \in \mathbb{N}$) approximates $(\psi^{(0)}, \mu^{(0)})$, and $k \geq m_{start}$. The value $m_{start} \in \mathbb{N}$ is chosen such that the condition is not immediately satisfied at the start of the curve approximation. The current end point at the other side of the curve is used as $(\psi^{(0)}, \mu^{(0)})$.

The second cause of recalculation of curves is due to multiple interconnections between two branch points. An example is given by the equation that describes the intersection of two cylinders (see section 9.3). Two (or more) branch points are found with a tangent direction that leads to the same solution curve, which would be approximated multiple times.

To prevent this sort of recalculation, a list A is initialized in algorithm 8.1 (page 252). This list consists of elements of the form $((\psi^{(b)}, \mu^{(b)}), (\dot{\psi}, \dot{\mu}))$ with an (approximate) branch point $(\psi^{(b)}, \mu^{(b)}) \in \mathbb{C}^n \times \mathbb{R}$ and an (approximate) tangent direction $(\dot{\psi}, \dot{\mu}) \in \mathbb{C}^n \times \mathbb{R}$. Each element in the list A corresponds to a curve that has yet to be approximated.

The list is initialized with the element $((\psi^{(0)}, \mu^{(0)}), (\dot{\psi}^{(0)}, \dot{\mu}^{(0)}))$. Whenever algorithm 6.1 is executed, the first element of A is used as its input. After a solution curve is approximated and its bifurcation points analysed (by algorithm 7.1, page 230), algorithm 8.1 loops over the approximated branch points and tangent directions to update the list. If such a branch point $(\psi^{(b)}, \mu^{(b)})$ approximates one in A, the element of A with the corresponding tangent direction is removed. If no approximations to the branch point appear in the list, elements of the form $((\psi^{(b)}, \mu^{(b)}), (\dot{\psi}, \dot{\mu}))$, with $(\dot{\psi}, \dot{\mu})$ the tangent directions to new solution curves, are added. Afterwards the element of A that lead to the last approximated curve's construction is removed as well.

Note that if $\mathcal{F}$ contains a discrete symmetry (described by the group G), we should also look for conjugates $(g(\psi^{(b)}), \mu^{(b)})$ with $g \in G$ when comparing the branch point $(\psi^{(b)}, \mu^{(b)})$ with the ones in A.

By keeping track of a list that indicates curves that have yet to be determined, multiple approximations of the same curve are prevented. Whenever a curve connects with an already found branch point, the corresponding element in the list is removed.

8.3 Structure of the implementation

Algorithm 8.1 has been implemented in Python, as an extension to the package PyNCT (Python Numerical Continuation Toolbox) [36]. Originally this package was developed for numerical continuation in auxine transport models [34, 35], a biology problem. It did not contain algorithms for the automatic exploration of connected solution landscapes.

The main files of the extended implementation are 'SolutionDiagram.py', 'Continuer.py' and 'Point.py'. A solution landscape is represented by the 'SolutionDiagram' class in 'SolutionDiagram.py'. Curves are represented by the 'Continuer' class in 'Continuer.py'. Approximated curves are contained in the solution landscape object as attributes.

A single solution point of the considered equation is represented by the 'Point' class in 'Point.py'. Each solution curve object contains multiple point objects in its attributes. Bifurcation points of a curve are added as an attribute as well. These points are represented by a subclass 'BifurcationPoint' of 'Point', described in the file 'BifurcationPoint.py'.

Numerical methods for approximating eigenpairs, solving linear and nonlinear systems, approximating tangent directions, ..., as described in chapters 3, 4, 5 and 7, are implemented in the files of the 'solvers' module. Required solvers are created as objects at the start of a connected solution landscape's construction, and are called when necessary. A solver for the analysis of stability (described in chapter 6) is part of the 'solvers' module as well. Note that, since they are used in multiple stages of the algorithms, approximated eigenpairs are stored as an attribute to point objects. Instead of recalculation, eigenpairs are recycled when possible.

To construct a connected solution landscape for an equation $\mathcal{F}$, the nonlinear system needs to be described by a module. The 'demo_superconductors' module contains the description of the Ginzburg-Landau equation (see section 2.5) for example.

Each such module should at least contain the files 'system.py', 'doContinuation.py' and 'parameters.json'. The file 'system.py' contains a 'system' class, used to construct an object with information of the function $\mathcal{F}$, its derivatives and properties (e.g. the inner product to use, possible preconditioning, possible self-adjointness of the partial Jacobian $\mathcal{F}_\psi$,...). The file 'doContinuation.py' contains a function 'doContinuation', when executed this function constructs the connected solution landscape of $\mathcal{F}$. Files of the form 'parameters.json' contain information on the internal parameters (see section 8.4 for an overview)

that should be used, and has to be provided as input for the function 'doContinuation'. An initial guess $(\psi^{(0)}, \mu^{(0)})$ is constructed in the 'doContinuation' function itself.

Other files that should be provided in the nonlinear system modules are 'Save.py' (used to save the constructed landscape to a h5 file), 'buildSolutionDiagram.py' (used to construct a landscape from a h5 file given as input) and 'plottools.py'. This last file is used to generate (interactive) plots of the solution landscape, either provided as input as a landscape object, or as a h5 file.

Whenever a new nonlinear system is studied, a new module should be created for the construction of its solution landscape. Templates for this purpose are provided by the 'demo_template' module. The files 'SolutionDiagram.py', 'Continuer.py', 'Point.py', 'BifurcationPoint.py' and module 'solvers' contain general code, independent of the considered nonlinear system.

8.4 Overview of internal parameters

Multiple numerical methods (e.g. GMRES, Newton-Krylov,...) are used when constructing a connected solution landscape by the PyNCT implementation. Each of these methods rely on several internal parameters, which are often problem-dependent. An overview of parameters that need to be provided is given in table 8.1

Table 8.1: Overview of internal parameters that need to be provided in PyNCT when constructing a connected solution landscape

$m_C \in \mathbb{N}$	Maximal amount of curves approximated for the connected solution landscape (by algorithm 8.1, page 252).
$m_P \in \mathbb{N}$	Maximal amount of points approximated for a single solution curve (by algorithm 6.1, page 193).
$m_{start} \in \mathbb{N}$	boundary on amount of points before the start point condition (provided as input by algorithm 8.1 in algorithm 6.1) should be checked.
$\epsilon_{det1} \in \mathbb{R}_0^+$	Tolerance used when checking the first condition of the near bifurcation condition (by algorithm 6.3, page 195).
$\epsilon_{det2} \in \mathbb{R}_0^+$	Tolerance used when checking the second condition of the near bifurcation condition (by algorithm 6.3, page 195).
$\Delta s^- \in \mathbb{R}_0^+$	Minimal step length to use when constructing a solution curve (by algorithm 6.1, page 193).
$\Delta s^+ \in \mathbb{R}_0^+$	Maximal step length to use when constructing a solution curve (by algorithm 6.1, page 193) or executing NSA (algorithm 6.6, page 198).

$m_{step} \in \mathbb{N}$	Threshold in algorithm 6.7 (page 199) to check if the step length should be decreased, due to slow convergence of the Newton method.
$\epsilon_{step} \in \mathbb{R}_0^+$	Threshold in algorithm 6.7 (page 199) to check if the step length should be set to minimal, due to a bifurcation possibly being near.
$g^- \in]0, 1]$	Factor the step length is decreased with, when the Newton method converges slowly.
$g^+ \in \mathbb{R}\ (\geq 1)$	Factor the step length is increased with, when the Newton method does not converge slowly.
$\epsilon_{stab} \in [0, 1]$	Threshold in algorithm 6.8 (page 200) for allowed percentage of unstable points in a stable approximate subcurve.
$\epsilon_{ortho} \in \mathbb{R}_0^+$	Tolerance used to check orthogonality between two vectors. Used in algorithm 7.2 (page 231).
$m_{rand} \in \mathbb{N}$	Amount of initial guesses used when solving a small nonlinear system of equations with the Newton algorithm. Used in algorithm 7.8 (page 238).
$m_{NSA} \in \mathbb{N}$	Maximal amount of NSA steps allowed in algorithm 6.6 (page 198).
$m_{bif} \in \mathbb{N}$	Maximal amount of Newton steps allowed in each NSA step.
$\epsilon_{NSA} \in \mathbb{R}_0^+$	Tolerance used for convergence in algorithm 6.6 (page 198).
$\epsilon_{bif} \in \mathbb{R}_0^+$	Tolerance used for the Newton method in each NSA step.
$\epsilon_{\tilde{\kappa}} \in \mathbb{R}_0^+$	Threshold used to choose sort of step in split (block) Newton method with mixed terms.
$m_{ext} \in \mathbb{N}$	Maximal amount of Newton steps allowed in algorithms 6.4 (page 196) and 6.5 (page 197).
$\epsilon_{ext1} \in \mathbb{R}_0^+$	Tolerance used in algorithms 6.4 (page 196) and 6.5 (page 197), to check whether the guess is a point of the solution curve and satisfies the extra established condition.
$\epsilon_{ext2} \in \mathbb{R}_0^+$	Tolerance used in algorithms 6.4 (page 196) and 6.5 (page 197), to check whether the guess contains an approximate null vector.

$m_{New} \in \mathbb{N}$	Maximal amount of Newton steps allowed when executing a pseudo-arclength continuation step (algorithm 6.2, page 195).
$\epsilon_{New} \in \mathbb{R}_0^+$	Tolerance used in the Newton method (when executed for a pseudo-arclength continuation step).
$m_{lin} \in \mathbb{N}$	Maximal amount of Arnoldi steps allowed in GMRES (algorithm 3.4, page 44).
$\epsilon_{lin} \in \mathbb{R}_0^+$	Tolerance used in GMRES.
$\epsilon_{eig1} \in \mathbb{R}_0^+$	Tolerance used to check convergence of a Ritz pair (in algorithm 3.3, page 43).
$\epsilon_{eig2} \in \mathbb{R}_0^+$	Tolerance used to decide whether the magnitude of a converged Ritz value is sufficiently low to be accepted. Used for Ritz pairs in CheckDetection, AdaptStep and StabAnalysis (algorithms 6.3 on page 195, 6.7 on page 199 and 6.8 on page 200).
$\epsilon_{defl} \in \mathbb{R}_0^+$	Tolerance used to decide whether the magnitude of a converged Ritz value is sufficiently low to be accepted. Used for Ritz pairs in split Newton methods, tangent direction constructions and for deflation.
$k_{eig} \in \mathbb{N}$	Maximal amount of Ritz pairs to calculate. Used for Ritz pairs in CheckDetection, AdaptStep and StabAnalysis (algorithms 6.3 on page 195, 6.7 on page 199 and 6.8 on page 200).
$k_{defl} \in \mathbb{N}$	Maximal amount of Ritz pairs to calculate. Used for Ritz pairs in split Newton methods, tangent direction constructions and for deflation.
$n_{eig} \in \mathbb{N}$	Maximal amount of allowed restarts in algorithm 3.3 (page 43) for each Ritz pair.
$m_{eig} \in \mathbb{N}$	Maximal amount of Arnoldi steps allowed in each restart of algorithm 3.3 (page 43).
$m_{NCG1} \in \mathbb{N}$	Maximal amount of allowed Newton iterations for line search part of NCG (algorithm 3.5, page 45).
$\epsilon_{NCG1} \in \mathbb{N}$	Tolerance for line search part of NCG.
$m_{NCG2} \in \mathbb{N}$	Maximal amount of allowed NCG steps.
$\epsilon_{NCG2} \in \mathbb{N}$	Tolerance for searched minimum in NCG.
$m_{CN} \in \mathbb{N}$	Amount of calculated time steps in algorithm 3.6 (page 47).

$\Delta t \in \mathbb{R}_0^+$	Size of time steps used in algorithm 3.6 (page 47).
$\epsilon_{comp1} \in \mathbb{R}_0^+$	Tolerance used when comparing an approximate bifurcation point with a point that satisfies the near bifurcation condition.
$\epsilon_{comp2} \in \mathbb{R}_0^+$	Tolerance used when comparing two general points. Used in the start point condition.
$\epsilon_{comp3} \in \mathbb{R}_0^+$	Tolerance used when comparing two approximate bifurcation points.
$\epsilon_{comp4} \in \mathbb{R}_0^+$	Tolerance used when comparing two tangent directions.
$\epsilon_{comp5} \in \mathbb{R}_0^+$	Tolerance used when comparing coefficients of a small nonlinear system.
$\epsilon_{comp6} \in \mathbb{R}_0^+$	Tolerance used when comparing solutions of a small nonlinear system.

Though problem-dependent, these parameters are often related. For example, the tolerance ϵ_{defl} should be chosen lower than ϵ_{det1}, but higher than ϵ_{NSA}.

The internal parameters should be chosen with care when PyNCT is used. A bad choice possibly leads to the failure of generating *all* curves of the problem's connected solution landscape. Note that the optimal choice for some of the parameters might differ depending on the curve being generated. Optimally the internal parameters should be retuned accordingly whenever a new curve is approximated (this might require some trial and error). Partial landscapes can be saved in PyNCT, and reloaded after the used 'parameters.json' file is updated (see section 8.3).

8.5 Notes on parallelization

The current Python implementation of algorithm 8.1 (page 252) cannot be run in parallel. Multiple parts can however be parallelized. Except for distribution of the memory over multiple nodes, which speeds up algorithms like GMRES and the calculation of Ritz pairs, multiple tasks can be performed in parallel as well.

A first such task occurs when a Newton method is combined with block elimination. In this case each Newton step requires the solution of two linear systems (e.g. (5.20) and (5.21) for the deflated block Newton method). These two systems are independent, and can be solved in parallel. Solving linear systems is typically computationally the most expensive part of the Newton algorithms. For the standard and deflated block Newton methods, parallelization would reduce the computational cost to almost the same level as the methods without block elimination.

Another task that can be performed in parallel are the two applications of pseudo-arclength continuation that occur at the start of a curve approxima-

tion in algorithm 8.1. Given an initial point $(\psi^{(0)}, \mu^{(0)})$ and direction $(\dot{\psi}, \dot{\mu})$, the curve through this point in the given direction is approximated by two applications of algorithm 6.1 (page 193): once in the direction $(\dot{\psi}, \dot{\mu})$, and once in $(-\dot{\psi}, -\dot{\mu})$, such that both sides of the curve are approximated. These two applications can be performed in parallel, the start point condition (8.1) should be adjusted appropriately.

In its current form, algorithm 6.1 first approximates a curve, bifurcation points are searched afterwards. Points that satisfy the near bifurcation condition are used as initial guesses for this purpose. These two tasks can be parallelized as well: when a point that satisfies the near bifurcation condition is found, a numerical method to find the corresponding bifurcation should immediately be started, simultaneous with the further approximation of the solution curve. When such a bifurcation is found, it should also be immediately analysed by algorithm 7.1 (page 230), and an appropriate element should be added to the list A, that contains input to use for curves yet to be approximated.

A final parallelization of tasks can be implemented using the list A itself: as soon as an element is added to this list, the corresponding curve can be approximated in parallel with other tasks. This way multiple curves are approximated simultaneously, greatly improving the speed of the connected solution landscape's construction. To prevent multiple curves from being generated, the end points (in two directions) of all approximate curves should be constantly compared (which can be done in parallel as well): if two end points approximate each other, the corresponding curves should be combined. Note that the comparison of end points should account for possible presence of discrete symmetry.

8.6 Appendix

Automatic exploration of a connected solution landscape

Algorithm 8.1 ConstructLandscape

Input $m_C, m_{start} \in \mathbb{N}$, functions $\mathcal{F} : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n$, $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n)$, list $C = [C_1, \ldots, C_l]$ of condition functions, inner product $\langle \cdot, \cdot \rangle$, initial guess $(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)}) \in \mathbb{C}^n \times \mathbb{R}$, list $G = [g_1, \ldots, g_k]$ of group actions (with $g_1 = e$ the identity element) representing discrete symmetry, set H of group actions representing continuous symmetry.

Output List L that represents the connected solution landscape of $\mathcal{F}$ through the point $(\psi^{(0)}, \mu^{(0)})$, a solution near $(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)})$. Each element (P, B, S) of L corresponds to a different curve with points P, bifurcations B and stability results S as column vectors.

Set $P : \mathbb{C}^n \times \mathbb{R} \to \mathcal{C}(\mathbb{C}^n) : (x, p) \to \mathcal{I}$ if not specified
Set $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_2$ if not specified
Set $G = [e]$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Set $H = \{e\}$ if not specified, with group action $e : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}^n \times \mathbb{R} : t \to t$.
Set $C = \begin{bmatrix} \end{bmatrix}$ an empty list if not specified
$\mu^{(0)} = \tilde{\mu}^{(0)}$
Calculate $\psi^{(0)}$ by executing NewtonMixed (algorithm 4.7) with $F = \mathcal{F}(\cdot, \mu^{(0)})$, $x^{(0)} = \tilde{\psi}^{(0)}$, $P = P(\cdot, \mu^{(0)})$ and given $\langle \cdot, \cdot \rangle$
$A = \begin{bmatrix} ((\psi^{(0)}, \mu^{(0)}), (0, 1)) \end{bmatrix}$
Initialize empty list $L = \begin{bmatrix} \end{bmatrix}$
$m = 0$
while $m < m_C$ and A is not empty **do**
    $m \leftarrow m + 1$
    Choose $((\psi, \mu), (\dot{\psi}, \dot{\mu}))$ the first element of A
    Define C_{start} by (8.1) with $(\psi'^{(0)}, \mu'^{(0)}) = (\psi, \mu)$ and given m_{start}
    $C' = \begin{bmatrix} C & C_{start} \end{bmatrix}$
    Calculate (P, B, S) by executing CurveCalculation (algorithm 6.1) with $C = C'$, $\psi^{(0)} = \psi$, $\mu^{(0)} = \mu$, $\dot{\psi}^{(0)} = \dot{\psi}$, $\dot{\mu}^{(0)} = \dot{\mu}$ and given $\mathcal{F}$, P and $\langle \cdot, \cdot \rangle$
    if the condition C_{start} is not satisfied **then**
        Set $(\psi^{(e)}, \mu^{(e)})$ the last column of P
        Define C_{start} by (8.1) with $(\psi'^{(0)}, \mu'^{(0)}) = (\psi^{(e)}, \mu^{(e)})$ and given m_{start}
        $C' = \begin{bmatrix} C & C_{start} \end{bmatrix}$
        Calculate (P', B', S') by executing CurveCalculation (algorithm 6.1) with $C = C'$, $\psi^{(0)} = \psi$, $\mu^{(0)} = \mu$, $\dot{\psi}^{(0)} = -\dot{\psi}$, $\dot{\mu}^{(0)} = -\dot{\mu}$ and given $\mathcal{F}$, P and $\langle \cdot, \cdot \rangle$
        Revert the columns of P', B' and S'
        $P \leftarrow \begin{pmatrix} P' & P \end{pmatrix}$
        $B \leftarrow \begin{pmatrix} B' & B \end{pmatrix}$

25: $S \leftarrow \begin{pmatrix} S' & S \end{pmatrix}$
26: $L = \begin{bmatrix} L & (P, B, S) \end{bmatrix}$
27: **end if**
28: Set k the amount of bifurcation points in B
29: **for** $j = 1, \ldots, k$ **do**
30: Set $(\psi^{(b)}, \mu^{(b)})$ the kth column of B
31: Define $G' \subseteq G$ the isotropy subgroup of $\psi^{(b)}$
32: Set $(\psi^{(1)}, \mu^{(1)})$ and $(\psi^{(2)}, \mu^{(2)})$ the points (as columns of P) closest to $(\psi^{(b)}, \mu^{(b)})$
33: $\dot{\psi} = \psi^{(2)} - \psi^{(1)}$
34: $\dot{\mu} = \mu^{(2)} - \mu^{(1)}$
35: $\gamma = \sqrt{\|\dot{\psi}\|^2 + \dot{\mu}^2}$
36: $\dot{\psi} \leftarrow \gamma^{-1}\dot{\psi}$
37: $\dot{\mu} \leftarrow \gamma^{-1}\dot{\mu}$
38: Calculate $(type, T)$ by executing BifurcationAnalysis (algorithm 7.1) with $t^{(0)} = (\dot{\psi}, \dot{\mu})$ $G = G'$ and given $\mathcal{F}$, P, $\langle \cdot, \cdot \rangle$, $(\psi^{(b)}, \mu^{(b)})$ and H
39: **if** $type$ = 'branch point' **then**
40: Set l the amount of tangent directions in T
41: **for** $i = 1, \ldots, l$ **do**
42: Set $(\dot{\psi}, \dot{\mu})$ the lth column of T
43: $new = True$
44: **for** $((\psi'^{(b)}, \mu'^{(b)}), (\dot{\psi}', \dot{\mu}'))$ in A **do**
45: **if** $\mu^{(b)} \approx \mu'^{(b)}$ and $\dot{\mu} \approx \pm\dot{\mu}'$ and $\exists h \in H, g \in G$ such that $g(\psi^{(b)}) \approx h(\psi'^{(b)})$ and $g(\dot{\psi}) \approx h(\dot{\psi}')$ **then**
46: $new \leftarrow False$
47: Remove the element $((\psi'^{(b)}, \mu'^{(b)}), (\dot{\psi}', \dot{\mu}'))$ from A
48: Break
49: **end if**
50: **end for**
51: **if** new **then**
52: $A = \begin{bmatrix} A & ((\psi^{(b)}, \mu^{(b)}), (\dot{\psi}, \dot{\mu})) \end{bmatrix}$
53: **end if**
54: **end for**
55: **end if**
56: **end for**
57: **end while**
58: Return L

CHAPTER 9

Numerical results

"Since the mathematicians have invaded the theory of relativity, I do not understand it myself anymore."
– ***Albert Einstein*** –

Chapter highlights:

- We provide and discuss connected solution landscapes for the dynamical systems and examples of chapter 2.
- These landscapes were generated by the techniques derived throughout the thesis (by application of the Python package PyNCT), indicating their effectiveness.
- A journal article about the contents of this chapter and chapter 7 has been submitted (see [110]).

9.1 Introduction

Connected solution landscapes for the dynamical systems and other examples of chapter 2 will be discussed in this chapter. The landscapes were generated by application of the Python package PyNCT, which is based on the algorithms derived in chapters 6, 7 and 8. By applying the established techniques, we were able to generate connected solution landscapes with multiple (e.g. 40+) curves. Modules for the discussed problems have been added as examples in PyNCT.

9.2 Equation of a circle

The first example is described in section 2.2. The function $\mathcal{F}$, given by (2.2), describes the equation of a circle. Though it lacks physical relevance, the example is used to test the implementation.

Application of algorithm 8.1 (page 252), in practice performed by the package PyNCT, yields the connected solution landscape of figure 9.1. The values $\tilde{\psi}^{(0)} = 1$, $\mu^{(0)} = 0$ were used to initialize the method. Note the gap in the circle

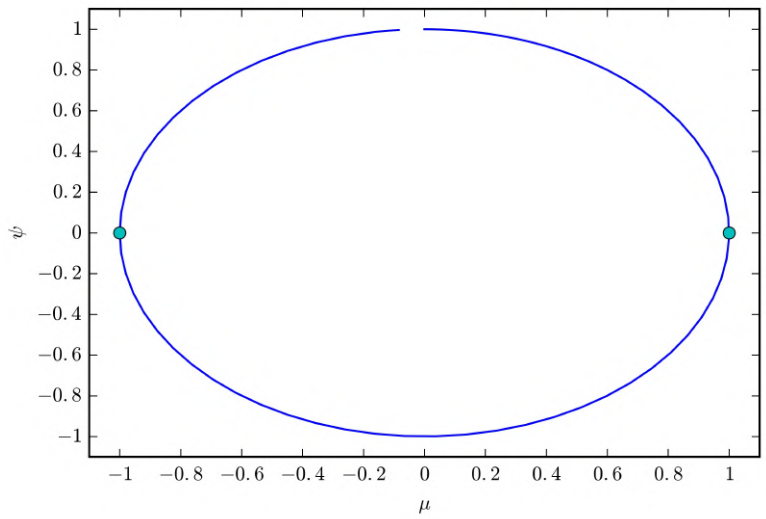


Figure 9.1: Connected solution landscape for the circle equation. Blue dots indicate bifurcation points.

around this point, this is a consequence of the start point condition (see (8.1)) that is satisfied near the end of the calculated curve. For the approximation of bifurcation points, the Newton step length adaptation method (algorithm 6.6, page 198) was performed.

Figure 9.1 shows a single solution curve $(\psi(s), \mu(s))$, with bifurcation points at $(\psi, \mu) = (0, -1)$ and $(\psi, \mu) = (0, 1)$. Both bifurcations are turning points. The solution curve satisfies

$$\psi(s) = \pm\sqrt{1 - \mu(s)},$$

indicating that the implementation of the pseudo-arclength continuation algorithm in PyNCT works correctly.

Note that, though dense linear algebra is easily applied for this example, we used numerical methods based on sparse linear algebra (algorithms 3.3, page 43 (RitzRestart), algorithm 3.4, page 44 (GMRES), ...). This was done for the purpose of testing the implementation of these methods.

9.3 Intersection of two cylinders

A second example is described in section 2.3, where a function $\mathcal{F}$ that represents the intersection of two cylinders is given (see (2.4)). We again apply the PyNCT package to generate the connected solution landscape of this function, this yields figure 9.2. The values

$$(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)}) = \left(\begin{pmatrix} 1 & 1 \end{pmatrix}^T, 0\right)$$

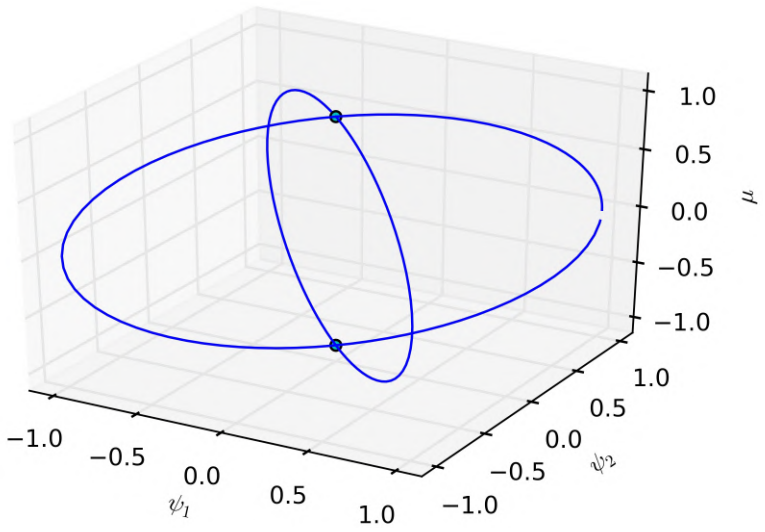


Figure 9.2: Connected solution landscape for the intersection of two cylinders. Blue dots indicate bifurcation points.

were used as guess for the initial point of the landscape, the Newton step length adaptation method (algorithm 6.6, page 198) was executed for the approximation of bifurcation points.

The solution landscape consists of two curves, and indeed represents the intersection of two cylinders (see figure 2.1). Two bifurcation points were approximated, given by the values

$$(\psi, \mu) = \left(\begin{pmatrix} 0 & 0 \end{pmatrix}^T, 1 \right), \qquad (\psi, \mu) = \left(\begin{pmatrix} 0 & 0 \end{pmatrix}^T, -1 \right).$$

Both bifurcations are branch points.

Similar to the circle equation (discussed in section 9.2), the solution landscape lacks physical relevance, but indicates that the implementation in PyNCT works correctly. Note that we again used methods based on sparse linear algebra for testing purposes, though the equation allows for dense linear algebra to be applied.

9.4 The Liouville-Bratu-Gelfand equation

Connected solution landscapes for the Liouville-Bratu-Gelfand equation will be discussed in the current section. Though the solutions are physically relevant, we will not discuss them in detail. The goal of using PyNCT on this equation is to test the application of the code to a nonlinear problem derived from a dynamical system.

The required function $\mathcal{F}$ for the algorithms is described by (2.7) in section 2.4. We make a distinction between the equation applied to a line segment

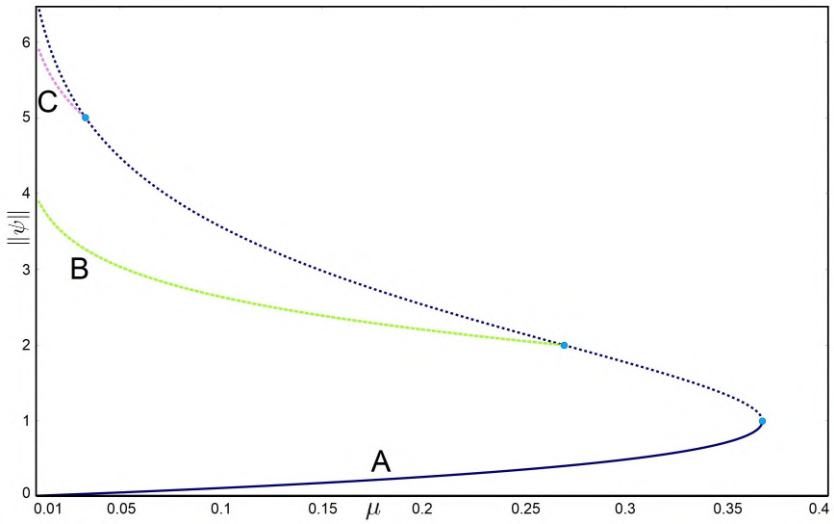


Figure 9.3: Connected solution landscape for the Liouville-Bratu-Gelfand equation on a line segment $[-0.5, 0.5]$. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.4.

($\Omega = [-0.5, 0.5]$) and to a square domain ($\Omega = [-0.5, 0.5]^2$). For both cases the package PyNCT is applied to an initial guess

$$(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)}) = (0, 0.01).$$

The preconditioner defined by (2.10) is used, and inner products (2.8) and (2.9) are respectively applied for the line segment and the square domain case. Algorithm 6.4 (solving an extended nonlinear system, see page 196) is used for the approximation of bifurcation points. The condition C, given by

$$C : \mathbb{C}^n \times \mathbb{R} \to \{True, False\} : (\psi^{(k)}, \mu^{(k)}) \to \begin{cases} True & \mu^{(k)} < 0.01, \\ False & \text{otherwise}, \end{cases} \tag{9.1}$$

is used as an additional condition to stop the pseudo-arclength continuation method, it is applied after each step to the last generated point $(\psi^{(k)}, \mu^{(k)})$ (for a certain $k \in \mathbb{N}$) of the curve.

9.4.1 Applied to a line segment

We first apply the PyNCT package to the equation defined on the domain $\Omega = [-0.5, 0.5]$, this represents a line segment. We use $n = 100$ discretization points. The generated solution landscape is given by figure 9.3.

With the current settings, 3 solution curves are constructed. Representative solutions for each of the curves are given in figure 9.4. Except for the two branch points (at approxiate values $\mu \approx 0.27$ and $\mu \approx 0.03$) that connect these curves, the algorithm finds a turning point at $\mu \approx 0.37$ for curve A.

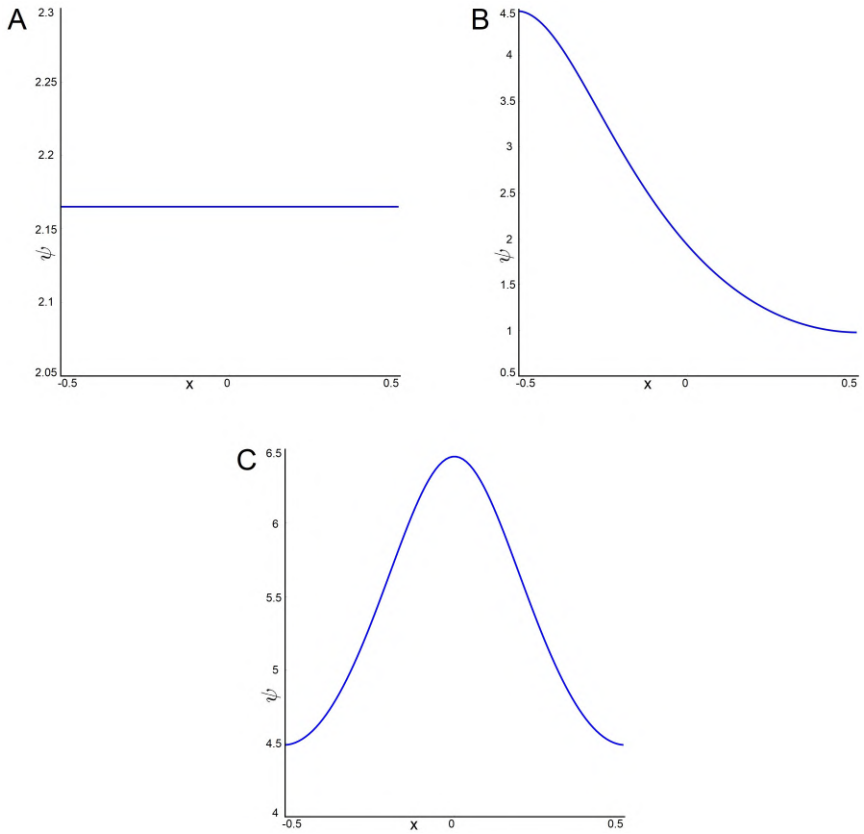


Figure 9.4: Representative solutions for the different curves of figure 9.3

9.4.2 Applied to a square domain

The branch points of the examples considered in the previous sections of this chapter never induced a partial Jacobian with kernel dimension greater than 1. The construction of their connected solution landscape did not require algorithms 7.5 (page 233) or 7.6 (page 234), which were derived for the 2-dimensional kernel case.

Contrary to these examples, the Liouville-Bratu-Gelfand equation applied to a square domain $\Omega = [-0.5, 0.5]^2$ does contain a discrete symmetry that induces branch points that require this case. The function $\mathcal{F}$ is invariant under the actions of D_4 for this problem. We apply the PyNCT package on the problem, using a discretization of $n = 900$ points.

Both branch points that induce Jacobians with a 1- and 2-dimensional kernel are encountered during the application. For the 1-dimensional case the required tangent directions are constructed by application of algorithm 7.4 (page 233). The construction of these directions for the 2-dimensional case is handled by algorithm 7.5. Application of PyNCT yields the connected solution landscape provided by figure 9.5. Several close-ups of this landscape are provided

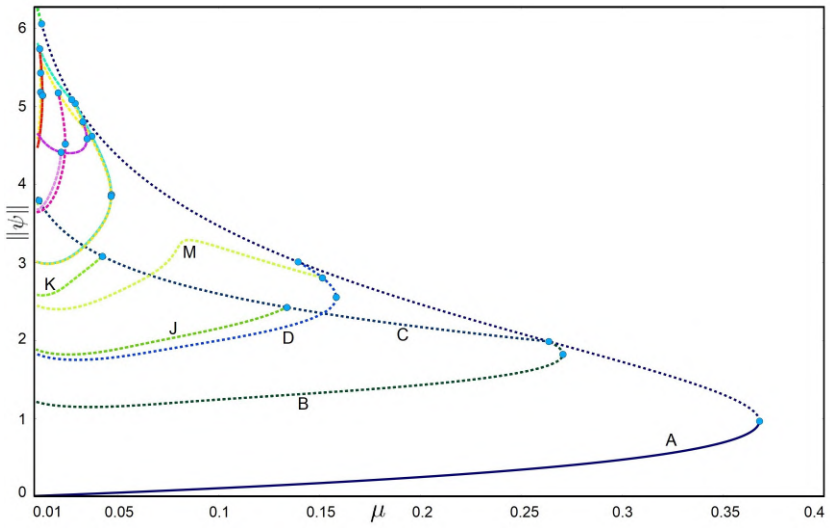


Figure 9.5: Connected solution landscape for the Liouville-Bratu-Gelfand equation on a square domain $[-0.5, 0.5]^2$. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.7.

by figure 9.6. Note that solution curves E and F are not distinguishable on the figures, due to the solutions belonging to these curves having exactly the same norm $\|\psi\|$ for each value of μ. This is also the case for curves Q and R, and U and V.

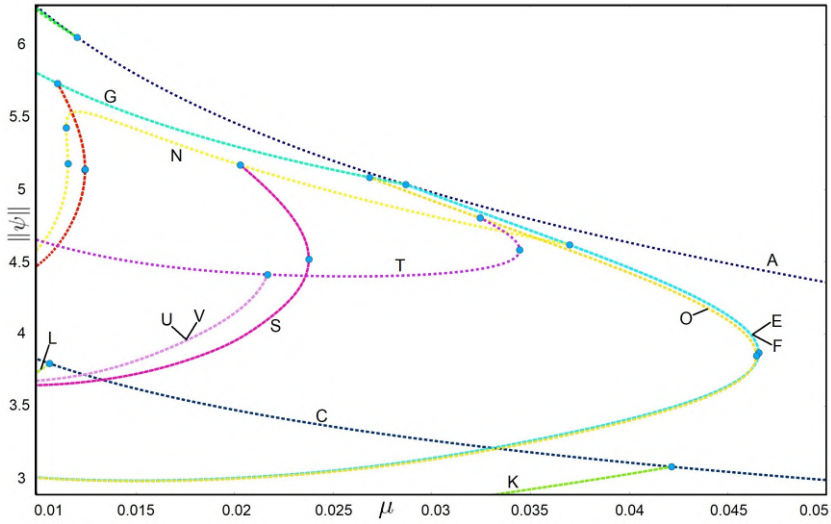
A total of 22 curves are constructed, representative solutions are given in figure 9.7. The curves are connected through 14 different branch points. Together with the 13 turning points, a total of 27 bifurcations were discovered.

Note that the connected solution landscape of figure 9.5 is actually not complete: more solution curves start emanating from the ones shown in the figure for values of μ smaller than 0.01. Due to the extra condition (9.1), we omitted the generation of these curves.

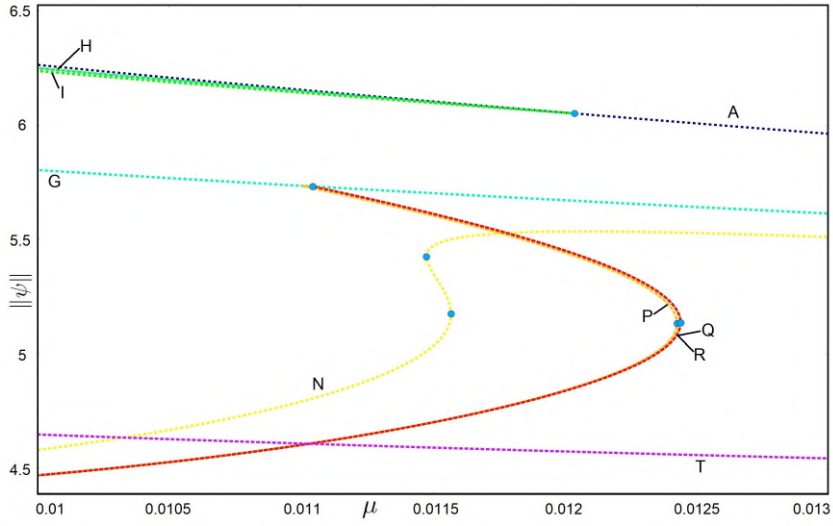
9.5 The Ginzburg-Landau equation

This section contains the connected solution landscapes for the Ginzburg-Landau equation (described in section 2.5), applied to multiple shapes of materials. We will only consider small-scale (mesoscopic) shapes, which are for example of interest when building nanoscale fluxonics devices to use in e.g. SQUIDs [65], RSFQ processors [41] and supercomputers [33, 54, 76]. The materials used in these devices are typically shaped as a triangle, square or disc. The methods discussed in previous chapters however allow for general shapes to be considered, which is indicated by some of the examples.

We restrict ourselves to extreme type-II superconductors, for which the Ginzburg-Landau equation decouples (see section 2.5.2). The materials are sub-



(a) Close-up for $0.01 \lesssim \mu \lesssim 0.05$, $2.9 \lesssim \|\psi\| \lesssim 6.7$



(b) Close-up for $0.01 \lesssim \mu \lesssim 0.013$, $4.4 \lesssim \|\psi\| \lesssim 6.5$

Figure 9.6: Several close-ups of the connected solution landscape of figure 9.5. Representative solutions for the curves are given in figure 9.7.

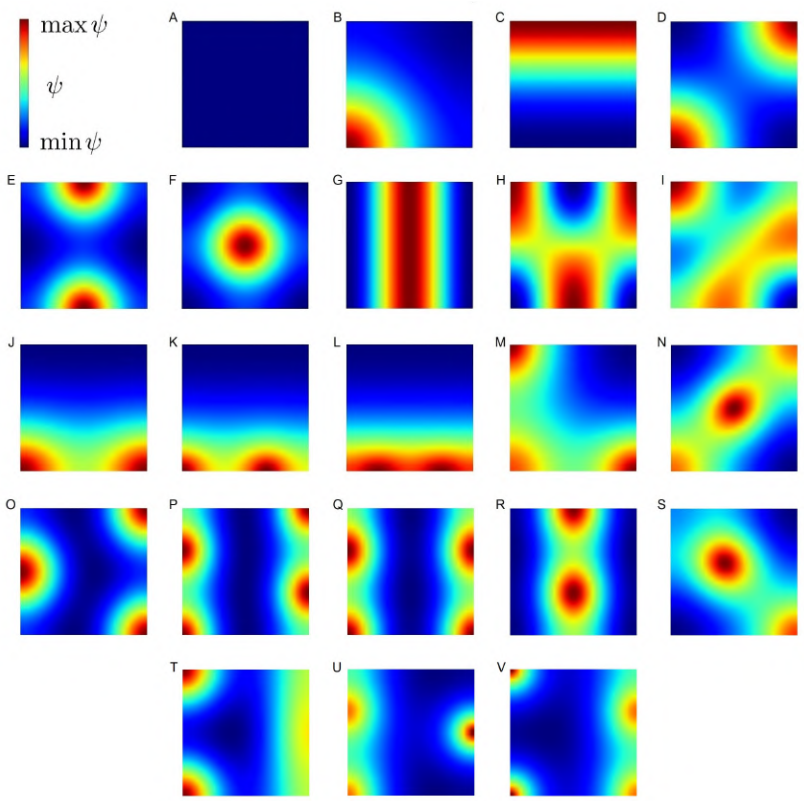


Figure 9.7: Representative solutions for the different curves of figures 9.5 and 9.6

ject to a homogeneous magnetic field, the dependence of the state on this fields strength (μ) is investigated. The PyNCT package is applied to the function $\mathcal{F}$, given by (2.20). The equation is presented in its dimensionless form. The magnetic vector potential $\mathbf{A}_0$ for $\mu = 1$, which is required to evaluate the discrete kinetic operator $K^{(h)}$ that appears in (2.20), will be described for each example (see section 2.5.2 for details on $\mathbf{A}_0$).

Note that solution landscapes will be plotted in terms of the expression

$$\mathcal{E}(\psi) = -|\Omega|^{-1} \int_\Omega |\psi|^4 d\Omega, \tag{9.2}$$

which is the part of the Gibbs energy (2.11) that depends on the order parameter ψ (see Schlömer [92] for more details).

Except for the irregular shaped material, the point

$$(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)}) = (1, 0)$$

will be used as the starting value. This point represents the homogeneous solution in absence of a magnetic field. The preconditioner given by (2.25) and inner product (2.21) are applied.

Most of the discussed shapes contain a discrete symmetry, described by either the cyclic or dihedral group. This induces an invariance on $\mathcal{F}$ under the actions of the corresponding symmetry group, as described by proposition 2.3. The function $\mathcal{F}$ is invariant under the actions of the continuous group S^1 as well, independent of the material's shape. The PyNCT package is able to deal with any complications caused by these symmetries (as discussed in previous chapters).

Extra conditions C_1 and C_2, defined by

$$C_1 : \mathbb{C}^n \times \mathbb{R} \to \{True, False\} : (\psi^{(k)}, \mu^{(k)}) \to \begin{cases} True & \mu^{(k)} < 0, \\ False & \text{otherwise}, \end{cases} \tag{9.3}$$

$$C_2 : \mathbb{C}^n \times \mathbb{R} \to \{True, False\} : (\psi^{(k)}, \mu^{(k)}) \to \begin{cases} True & \mathcal{E}(\psi^{(k)}) \approx 0, \\ False & \text{otherwise}, \end{cases} \tag{9.4}$$

are used in the application of PyNCT for all shapes except the irregular one. Condition C_1 asserts a positive magnetic field strength, condition C_2 stops pseudo-arclength continuation when the Gibbs energy part $\mathcal{E}(\psi)$, corresponding to the state ψ, approaches zero. In this case the curve connects to the trivial solution branch that consists of points $(\psi, \mu) = (0, a)$ for $a \in \mathbb{R}$. The conditions are applied after each pseudo-arclength continuation step, to the last generated point $(\psi^{(k)}, \mu^{(k)})$ (for a certain $k \in \mathbb{N}$) of the curve.

9.5.1 Square-shaped material

The first material we consider is shaped as a square, the appropriate symmetry group for the equation is the group $D_4 \times S^1$. We use $n = 20737$ discretization points. The size of the edges is denoted by d, scaled in units of the coherence length ξ (see section 2.5). We apply PyNCT for two different sizes. In both cases algorithm 6.5 (page 197) was used for approximating bifurcation points. Algorithm 7.6 (page 234) was used to construct tangent directions for branch points that induce a Jacobian with kernel dimension 2 (ignoring null vectors induced by the S^1 symmetry).

Side length $d = 3$

The connected solution landscape for a square with side length 3 was discussed in Schlömer [92]. We recreate the result by applying PyNCT, yielding figure 9.9. The magnetic vector potential $\mathbf{A}_0$ used in the discrete Ginzburg-Landau equation (2.20) is given by figure 9.8.

Two bifurcation points were found during the continuation, at $\mu \approx 1.64$ and $\mu \approx 1.17$. Both bifurcations are branch points that induce a Jacobian with a 2-dimensional kernel, application of algorithm 7.6 yielded the tangent directions to the emerging curves. The generated landscape in figure 9.9 shows 4 different solution curves, denoted as A, B, C and D. Representative solutions for these curves are given in figure 9.10.

The solution $(\psi, \mu) = (1, 0)$, which was used as a starting point for PyNCT, is part of solution curve A. Solutions for this curve are invariant under the

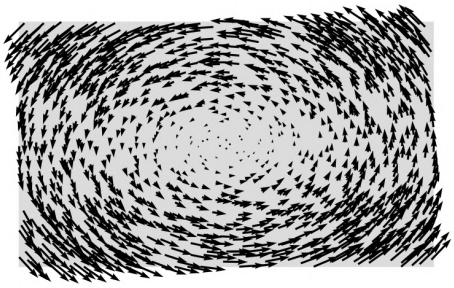


Figure 9.8: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.9. Vectors are scaled by 25%.

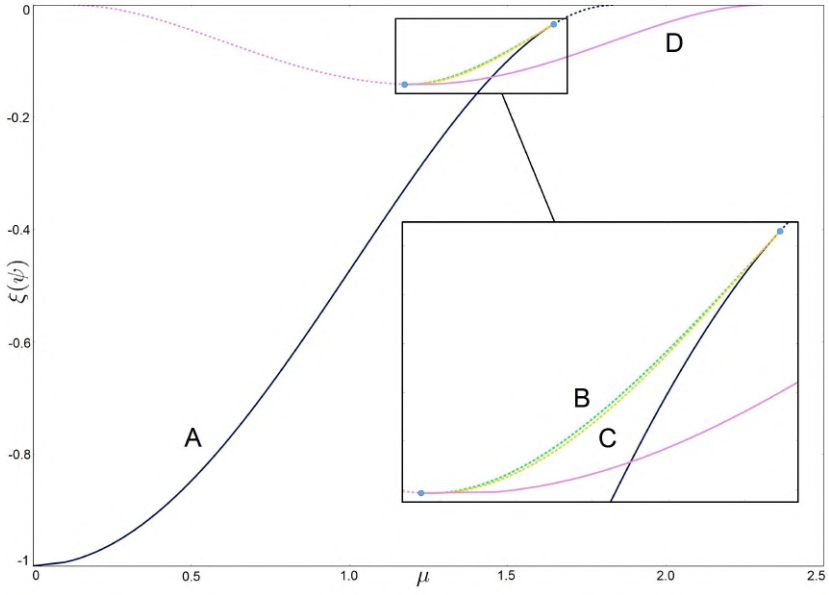


Figure 9.9: Recreated from Schlömer [92]. Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a square-shaped material with side length 3. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.8. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.10.

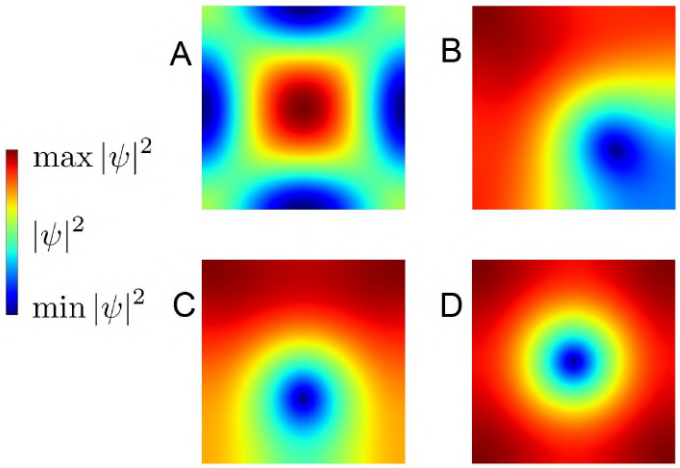


Figure 9.10: Recreated from Schlömer [92]. Representative solutions for the different curves of figure 9.9.

actions of the entire group D_4. For values of μ between 0 and approximately 1.64 (the value for the first bifurcation), curve A is stable. For non-zero field strength, the solutions consist of zones of low supercurrent density near the edges of the domain. At $\mu \approx 1.64$ branch A destabilizes. Further increase of μ along this curve shows how the branch connects to the trivial solution curve $\psi = 0$, $\mu \in \mathbb{R}$. This happens at $\mu \approx 1.84$.

Except for curve A, the solutions of curve D are also invariant under all actions of D_4. Its solutions show a single vortex in the centre of the material. It is stable for values of μ between approximately 1.17 (the value for the second bifurcation) and 2.30. When the fields strength is increased along solution branch D, four vortices start to appear near the edges of the domain. The curve connects to the trivial solution branch $\psi = 0$, $\mu \in \mathbb{R}$ at $\mu \approx 0.13$ and $\mu \approx 2.30$.

Curves A and D are connected through curves B and C. Starting from the bifurcation of branch A at $\mu \approx 1.64$ and decreasing the field strength, curve B shows how a single vortex forms at one of the corners and moves towards the centre of the material, reaching this destination at the bifurcation $\mu \approx 1.17$. Branch C shows the same behaviour, with the vortex appearing from one of the edges.

Side length $d = 5.5$

The connected solution landscape for a square-shaped material with edges sized $d = 3$ only contained 4 different solution curves. We now consider the same shape, this time with a side length $d = 5.5$. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.11. This problem was also treated in Schlömer [92], where a solution landscape with 13 different solution branches was constructed. Application of the PyNCT package yields the same landscape, with an extra

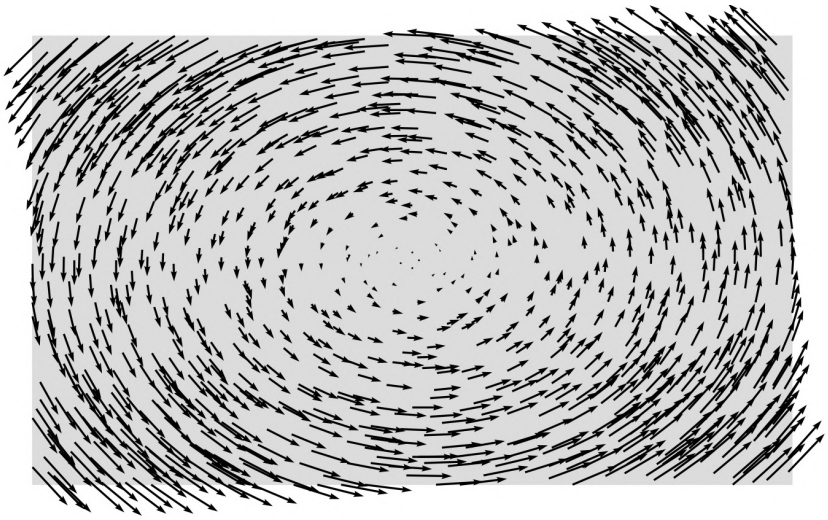


Figure 9.11: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.12. Vectors are scaled by 25%.

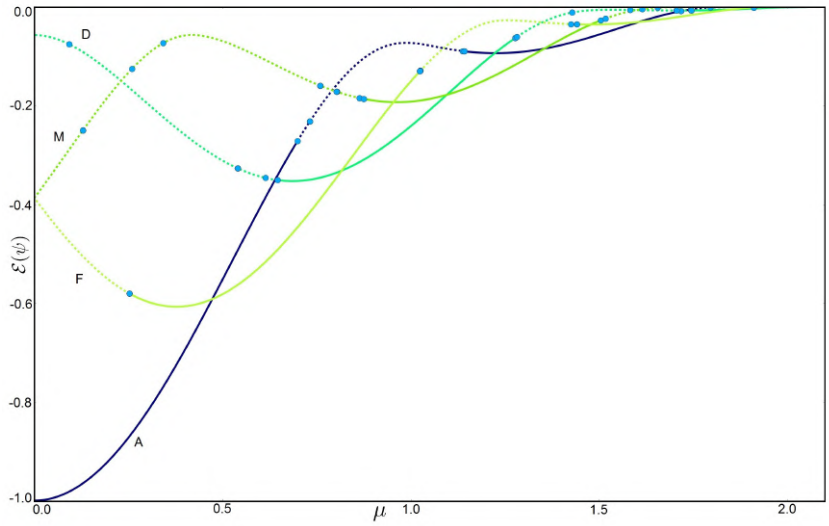
30 branches, resulting in a total of 43 solution curves. The four main branches (with full D_4 symmetry) are shown in figure 9.12. Several close-ups of the landscape containing other solution curves are given in figure 9.14. A schematic representation of the diagram can be found in figure 9.13. Finally, representative solutions for each curve are given in figure 9.15.

A total of 60 branch points were discovered during the continuation, their values are given in table 9.1. Several of these only contained a 1-dimensional kernel for which application of algorithm 7.4 (page 233) yielded the (single) new tangent direction. Other branch points corresponded to a kernel with dimension 2. For these points the tangent directions were constructed by application of algorithm 7.6 (based on the equivariant branching lemma, see page 234).

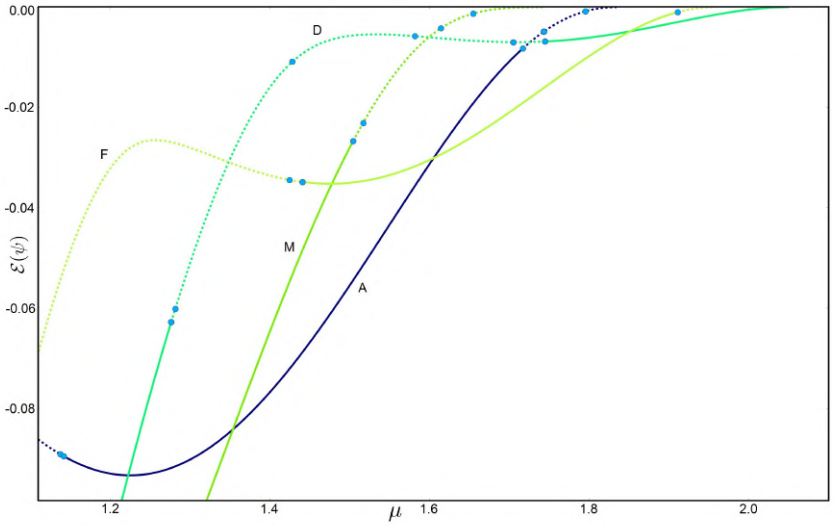
Note that except for the four main branches (A, D, F and M), solution curve B contains a stable part as well: it is stable for magnetic field strength values between approximately 0.6308 and 0.646098, for solutions that consist of two vortices very close to the center, on the horizontal or vertical axis. Curve B is the only non-main branch to contain a stable part, the stability of this region has been verified by the Crank-Nicolson time step method (algorithm 3.6 on page 47).

9.5.2 Triangular material

The second material is shaped as a regular triangle, the edge size is denoted by d. $n = 9409$ discretization points are used. The equation $\mathcal{F}$ is $D_3 \times S^1$ invariant for this problem. We again consider two different sizes, for both cases we applied algorithm 6.5 (page 197) for the approximation of bifurcation points.



(a) Complete view



(b) Close-up for $1.1 \lesssim \mu \lesssim 2.1$, $-0.1 \lesssim \mathcal{E}(\psi) \leq 0$

Figure 9.12: Main solution curves of the landscape for the extreme type-II Ginzburg-Landau equation applied to a square-shaped material with side length 5.5. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.11. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.15.

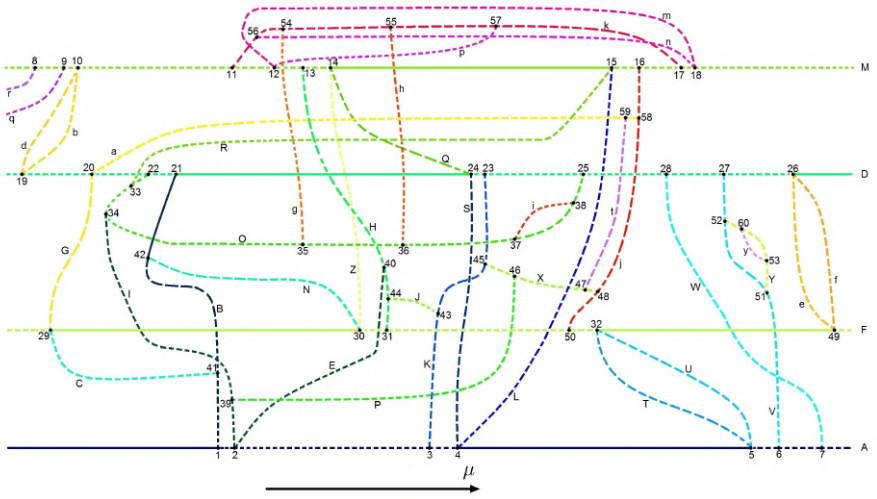


Figure 9.13: Schematic representation of the connected solution landscape (including branch points) for the extreme type-II Ginzburg-Landau equation applied to a square-shaped material with edge 5.5. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.11. Solid (dashed) lines represent stable (unstable) solutions. Representative solutions for each curve are given in figure 9.7, the μ values for the branch points are given in table 9.1.

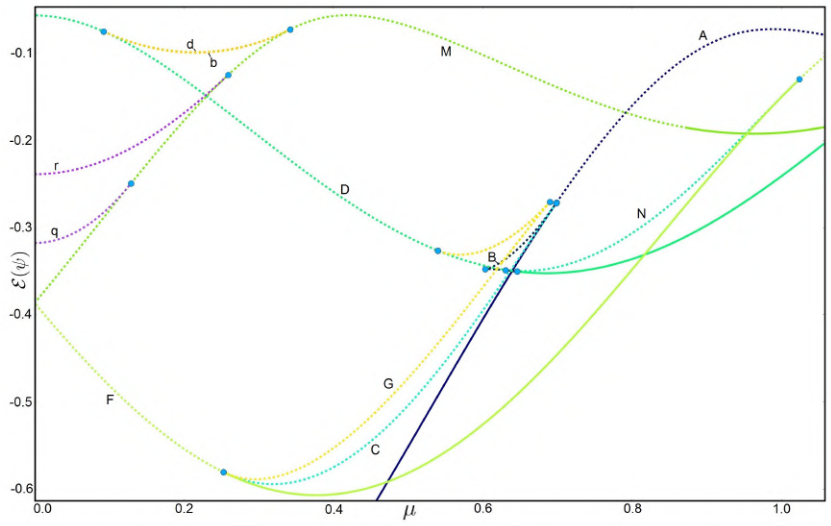
Side length $d = 4$

The size of the edges of the first triangular material equals $d = 4$. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.16. The connected solution landscape, constructed by application of the PyNCT package, is given by figure 9.17. Representative solutions for each curve are given by figure 9.18.

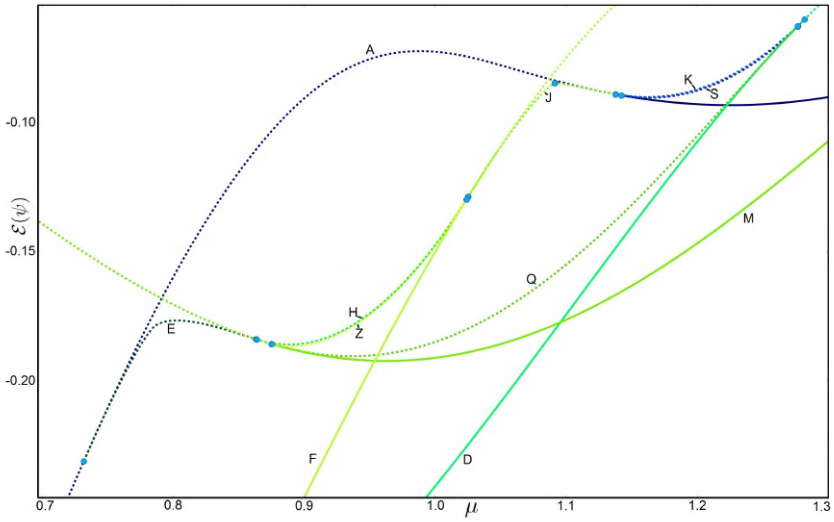
The PyNCT package found 4 bifurcation points during the continuation: 2 turning points and 2 branch points. Both branch points induced a partial Jacobian with kernel dimension 2. Application of algorithm 7.5 (page 233) yielded the tangent directions to the curves emerging from these points. Note that only the initial steps of this algorithm had to be executed: a reduced system of equations is found for $k = 2$ in algorithm 7.6.

As for the square-shaped material with side length $d = 3$, we find two solution curves with full D_3 symmetry. These are labelled A and C in figure 9.17. Curve A contains the starting point $(\psi, \mu) = (1, 0)$. Increasing the strength of the applied magnetic field from this solution, three zones of low supercurrent density form near the sides of the material. It connects to the trivial solution branch $\psi = 0$, $\mu \in \mathbb{R}$ at $\mu \approx 2.33$. The solution curve is stable for values of μ between 0 and approximately 2.202, the value for the first branch point.

The solutions of curve C contain a single vortex in the centre of the triangle. Increasing the magnetic field strength along this curve, three zones of low supercurrent density form near the edges. The curve is unstable for values

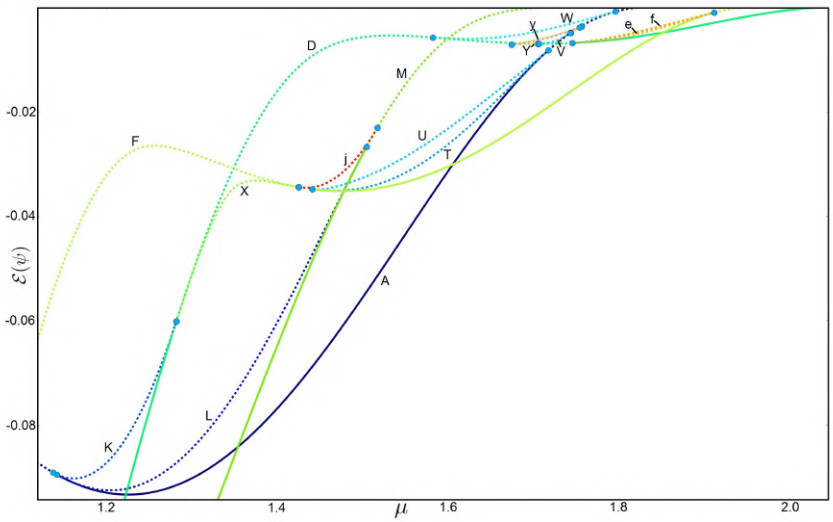


(a) Curves B, C, G, N, b, d, q and r

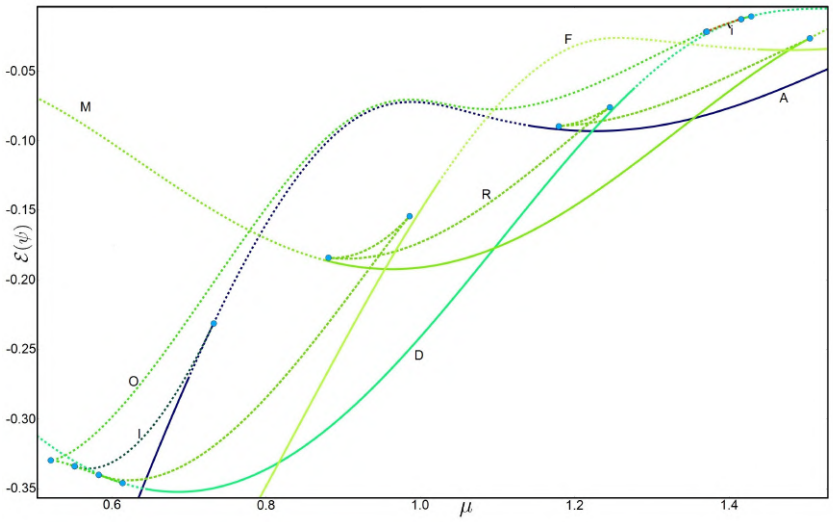


(b) Curves E, H, J, K, Q, S and Z

Figure 9.14: Several close-ups of the connected solution landscape of figure 9.12, including connective solution curves. Representative solutions for the curves are given in figure 9.15.

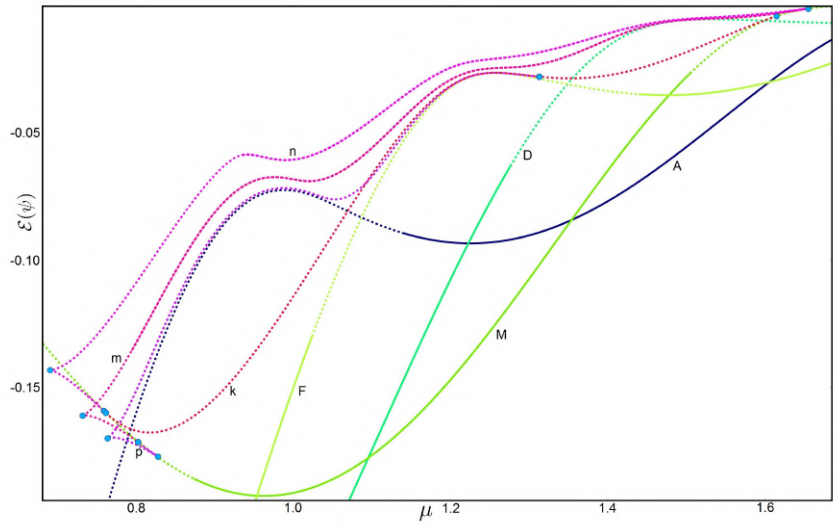


(c) Curves K, L, T, U, V, W, X, Y, e, f, j and y

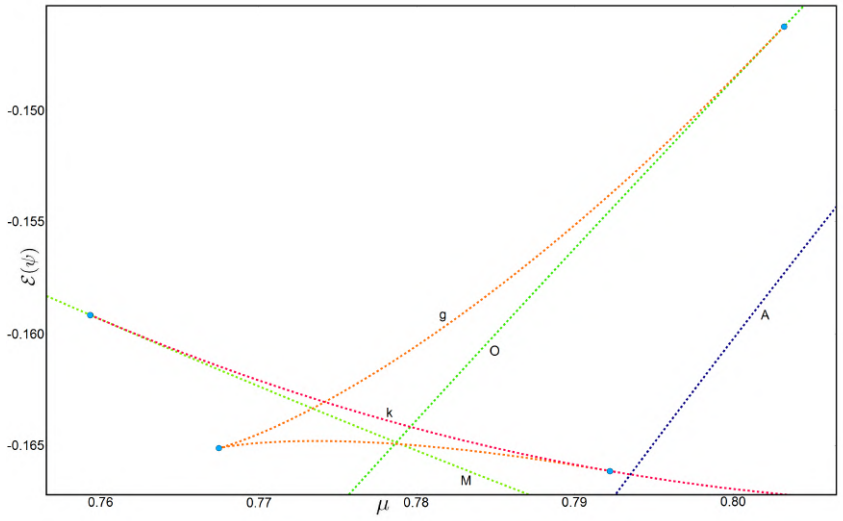


(d) Curves I, O, R and i

Figure 9.14: Several close-ups of the connected solution landscape of figure 9.12 (cont.).

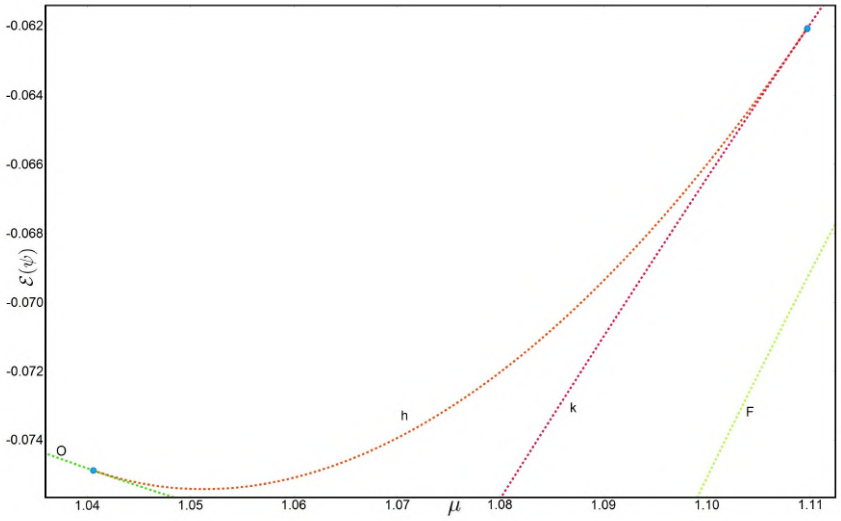


(e) Curves k, m, n and p

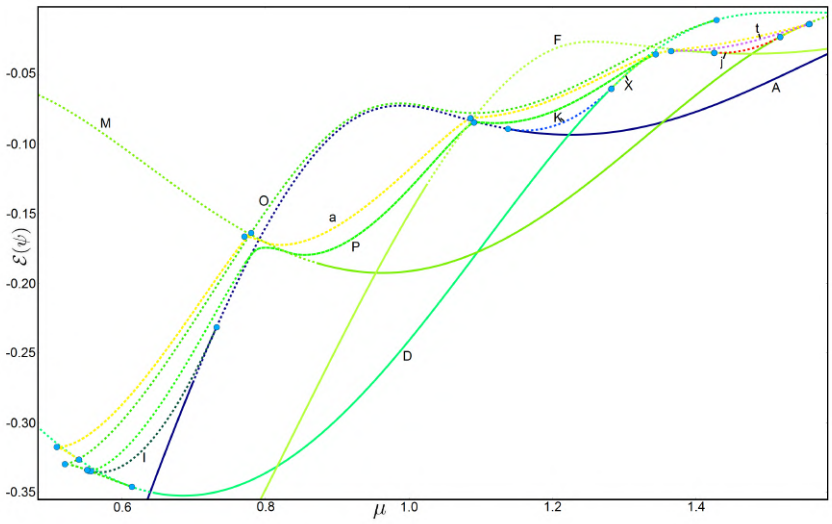


(f) Curve g

Figure 9.14: Several close-ups of the connected solution landscape of figure 9.12 (cont.).



(g) Curve h



(h) Curves I, K, O, P, X, a, j and t

Figure 9.14: Several close-ups of the connected solution landscape of figure 9.12 (cont.).

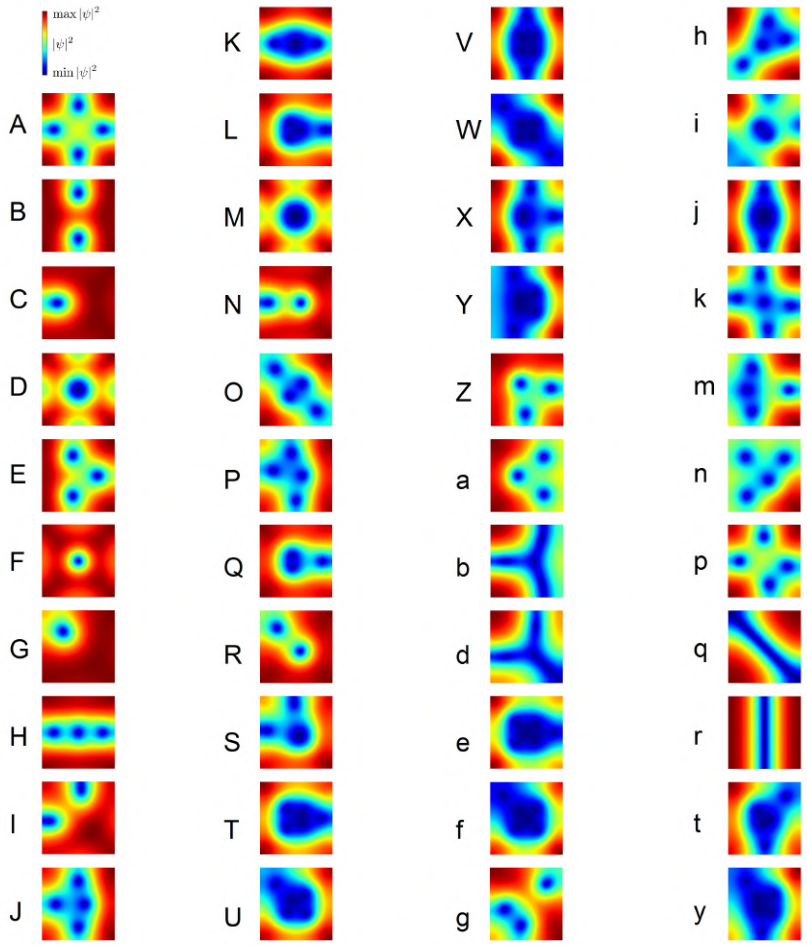


Figure 9.15: Representative solutions for the different curves of figures 9.12, 9.13 and 9.14

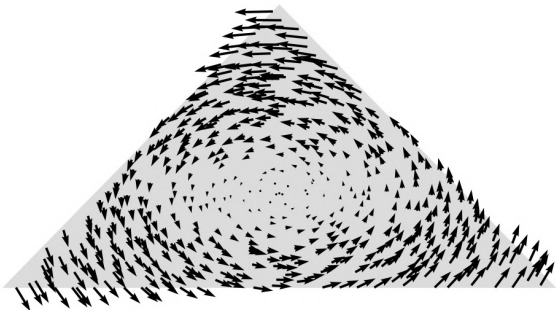


Figure 9.16: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.17. Vectors are scaled by 25%.

Table 9.1: Approximate values of the applied magnetic fields strength (μ) for the branch points displayed in figure 9.13.

Id.	μ value	Id.	μ value	Id.	μ value	Id.	μ value
1	0.6989	16	1.51736	31	1.02539	46	1.34396
2	0.7319	17	1.6149	32	1.4412	47	1.3655
3	1.13777	18	1.655	33	0.58275	48	1.425396
4	1.1423	19	0.09241	34	0.551370	49	1.9115
5	1.7176	20	0.54002	35	0.80322	50	1.425179
6	1.7437	21	0.646098	36	1.0406	51	1.7430
7	1.7959	22	0.61367	37	1.3712	52	1.7063
8	0.12905	23	1.281974	38	1.41573	53	1.75378
9	0.2591	24	1.27688	39	0.5566	54	0.7922136
10	0.34209	25	1.42875	40	0.8639	55	1.10976
11	0.7594	26	1.7453	41	0.6980	56	0.761674
12	0.8029	27	1.7055	42	0.6308	57	1.312923
13	0.8635	28	1.5824	43	1.13779	58	1.517199
14	0.875147	29	0.25285	44	1.02538	59	1.557099
15	1.5048	30	1.0240	45	1.28195	60	1.67462

$\mu \lesssim 1.835$, the value for the second branch point. It is stable between values $\mu \approx 1.835$ and $\mu \approx 2.75$. At this last value, and the value $\mu \approx 0.27$, the curve connects to the trivial solution branch $\psi = 0$, $\mu \in \mathbb{R}$.

Curves A and C are connected by a single curve B. To describe this curve, we start from the branch point of curve A at $\mu \approx 2.202$. Decreasing μ, a vortex moves from one side of the triangle to the opposite corner. It reaches the centre of the material at $\mu \approx 1.835$, where curves B and C connect in a branch point. Further decreasing μ along curve B, the vortex continues moving towards the opposite corner of the edge it appeared at. A turning point is encountered for $\mu \approx 1.825$. The movement towards the corner is continued when we increase μ from this value, encountering a second turning point at $\mu \approx 2.204$. A final decrease from this point shows how curve B reaches its own starting point, the bifurcation at $\mu \approx 2.202$.

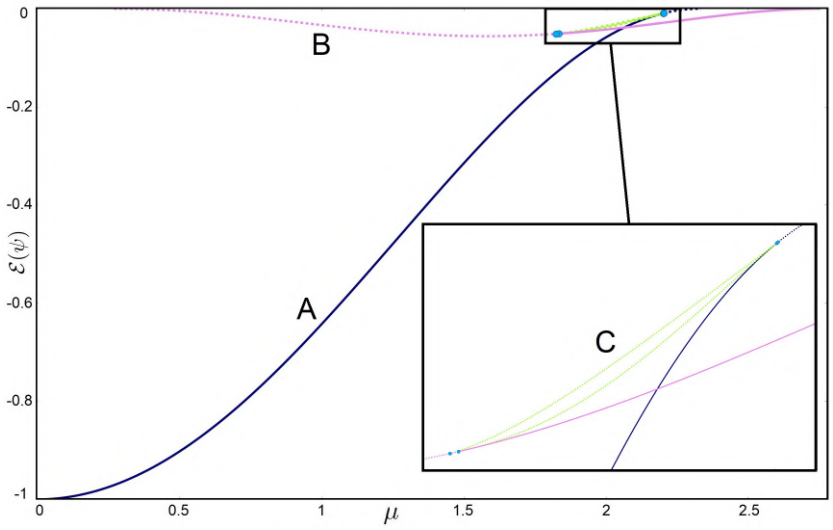


Figure 9.17: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a triangular material with side length 4. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.16. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.18.

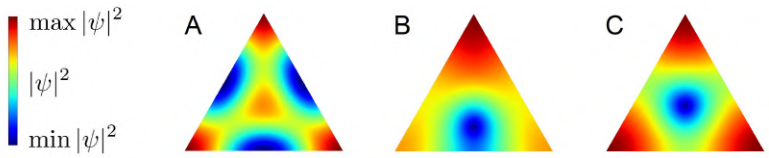


Figure 9.18: Representative solutions for the different curves of figure 9.17.

Side length $d = 6$

A second connected solution landscape was constructed for the triangular material, this time with edges sized $d = 6$. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.19. The landscape is given by figure 9.20, representative solutions by figure 9.21. A detailed description of the different solution curves is given in Wouters [110].

A total of 7 solution curves and 16 bifurcation points were found during the continuation: 7 turning points and 9 branch points. For all of these branch points, the corresponding Jacobian contained a kernel with dimension $n = 2$. The initial steps of algorithm 7.5 were again sufficient for the construction of tangent directions.

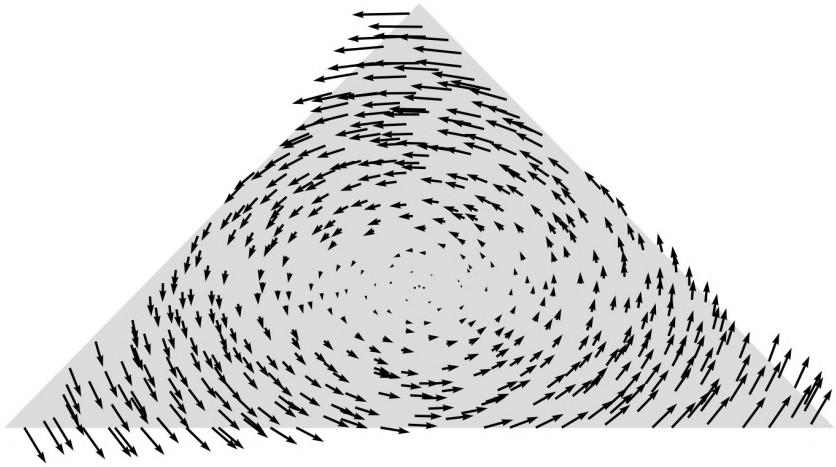


Figure 9.19: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.20. Vectors are scaled by 25%.

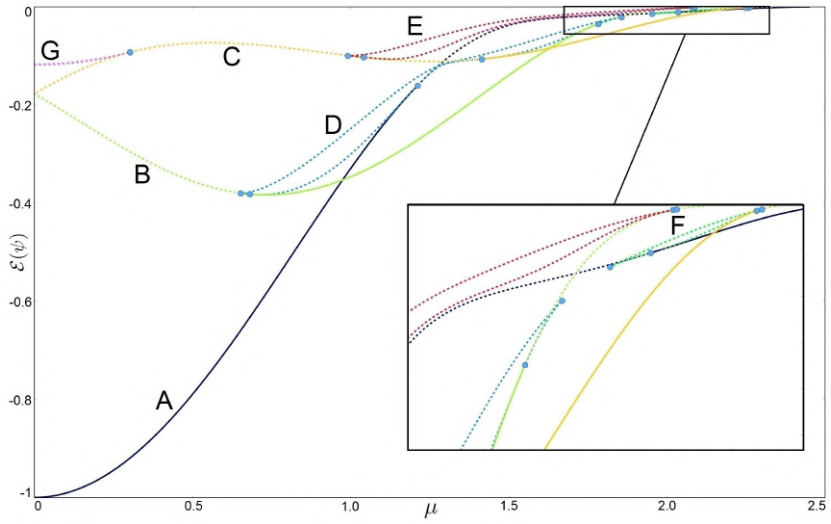


Figure 9.20: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a triangular material with side length 6. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.19. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.21.

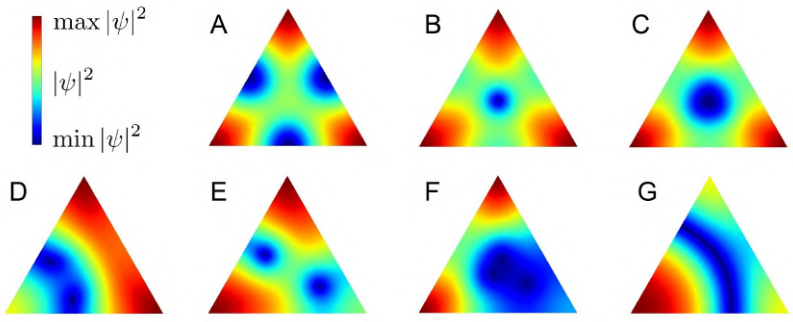


Figure 9.21: Representative solutions for the different curves of figure 9.20.

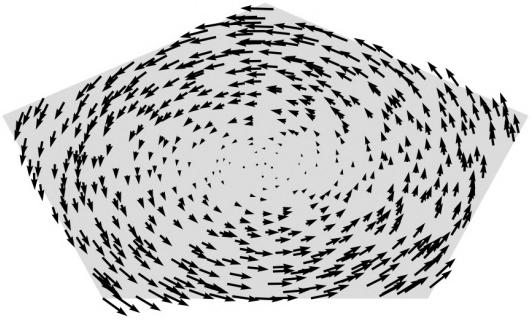


Figure 9.22: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.23. Vectors are scaled by 25%.

9.5.3 Pentagon-shaped material

The final regular shape we consider is that of a pentagon, inducing a $D_5 \times S^1$ invariance of the equation. We use $n = 10401$ discretization points, and again consider two different sizes of edge length (denoted by d). Contrary to the square-shaped and triangular materials, the split block Newton method (SBN, algorithm 5.4 on page 159) was used for pseudo-arclength continuation, instead of the standard method. Bifurcation points were approximated by a different algorithm as well, we used the Newton step length adaptation method (algorithm 6.6 on page 198) for this purpose. Tangent directions for branch points with kernel dimension 2 were constructed by application of algorithm 7.6 (page 234).

Side length $d = 2.35$

A pentagon with size length $d = 2.35$ is considered first. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.22. The connected solution landscape and representative solutions for each curve are given by figures 9.23 and 9.24.

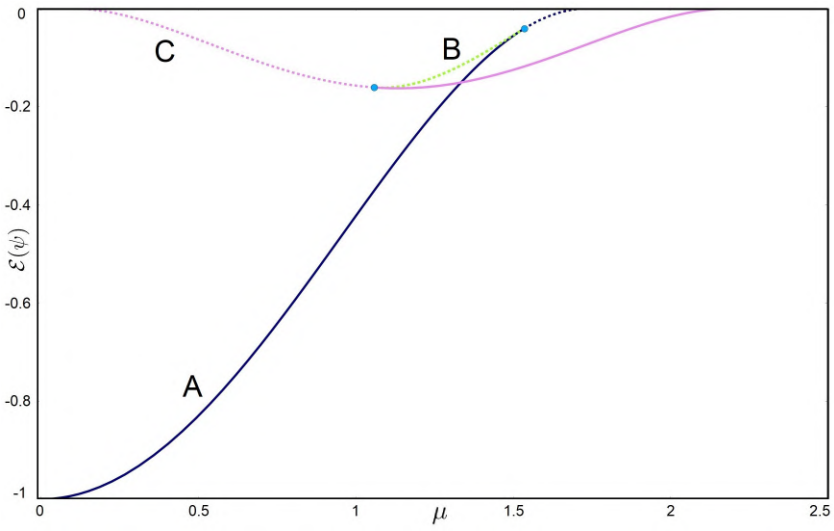


Figure 9.23: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a pentagon-shaped material with side length 2.35. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.22. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.24.

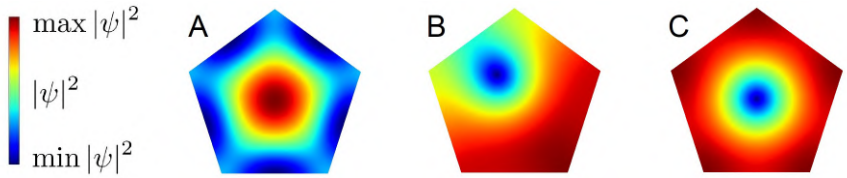


Figure 9.24: Representative solutions for the different curves of figure 9.23.

Three different solution curves were generated by the PyNCT package. Solutions of branches A and C are D_5-invariant, curve B connects these two branches through the branch points at $\mu \approx 1.54$ and $\mu \approx 1.06$. Both branch points induce a Jacobian with a 2-dimensional kernel. Solutions of curve A are stable for values of magnetic field strength up to $\mu \approx 1.54$. For curve C this is the case for values between $\mu \approx 1.06$ and $\mu \approx 2.19$.

Side length $d = 3.13$

Next we consider a material shaped as a pentagon with side length $d = 3.13$. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.25. Application of PyNCT yields the connected solution landscape and representative solutions given by respectively figures 9.26 and 9.28. Two close-ups of figure 9.26 are given in figure 9.27.

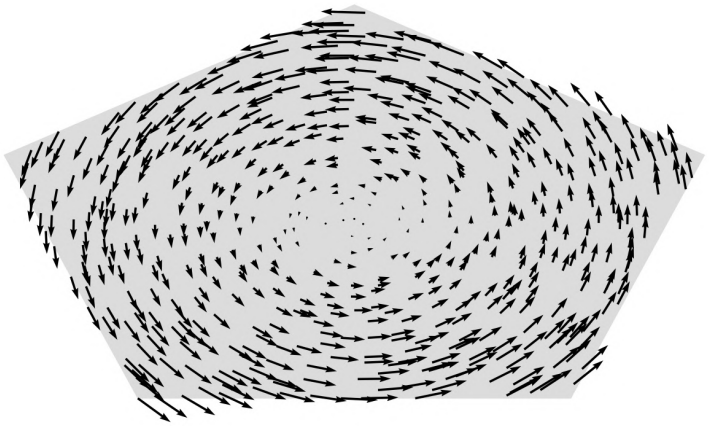


Figure 9.25: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.26. Vectors are scaled by 25%.

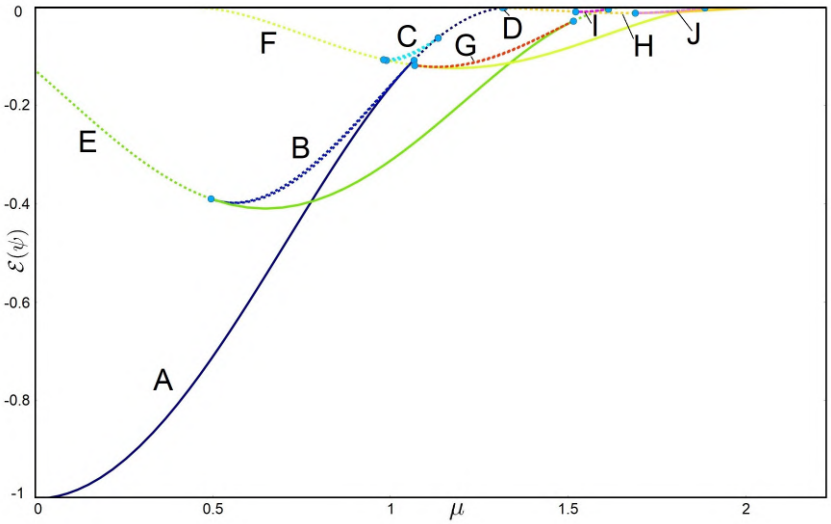
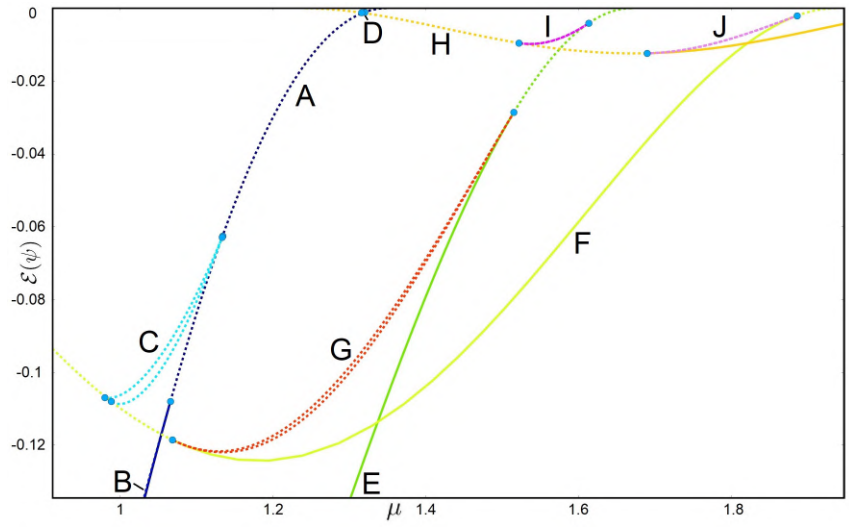
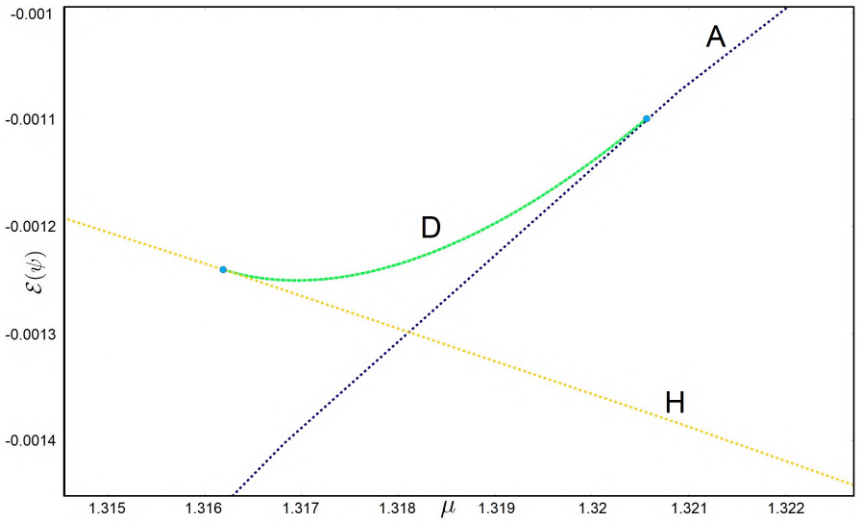


Figure 9.26: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a pentagon-shaped material with side length 3.13. The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.25. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.28.



(a) Close-up for $0.9 \lesssim \mu \lesssim 1.9$, $-0.13 \lesssim \mathcal{E}(\psi) \lesssim 0$



(b) Close-up for $1.314 \lesssim \mu \lesssim 1.323$, $-0.00145 \lesssim \mathcal{E}(\psi) \lesssim -0.001$

Figure 9.27: Several close-ups of the connected solution landscape of figure 9.26. Representative solutions for the curves are given in figure 9.28.

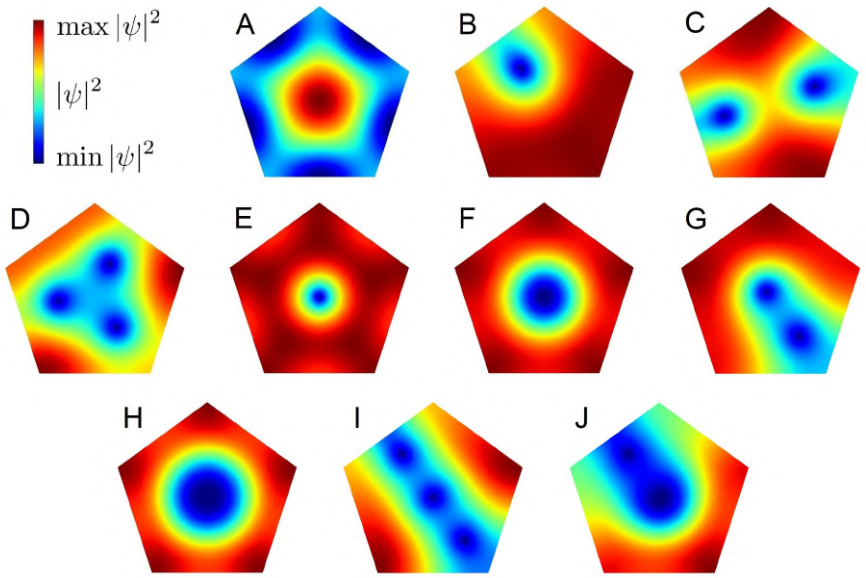


Figure 9.28: Representative solutions for the different curves of figures 9.26 and 9.27.

For this problem the PyNCT package generated a total of 10 solution curves and 14 bifurcations (2 turning and 12 branch points). The partial Jacobians that correspond to these branch points always contained a 2-dimensional kernel.

9.5.4 Star-shaped material

The next material is shaped as a four-pointed star. It only exhibits rotational symmetry, inducing an invariance of $C_4 \times S^1$ on the equation. The domain is chosen such that it fits into a square with edge 6: the distance between the center and outer corners is given by $\sqrt{10}$, the one between the center and inner corners by $\sqrt{2}$. The size of the short and long edges of the star itself are respectively given by 2 and $2\sqrt{2}$. The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.29. The connected solution landscape, and representative solutions, associated with this material are given by figures 9.30 and 9.31. Several close-ups of the landscape are given in figure 9.32.

19 solution curves were generated by the PyNCT package, interconnected through a total of 20 branch points. These points were approximated by application of algorithm 6.5 (page 197). Some of these corresponded to a Jacobian with a 1-dimensional kernel, for which algorithm 7.4 (page 233) was applied to construct tangent directions. Other branch points did correspond to a Jacobian with a 2-dimensional kernel, for these points algorithm 7.5 (page 233) was applied.

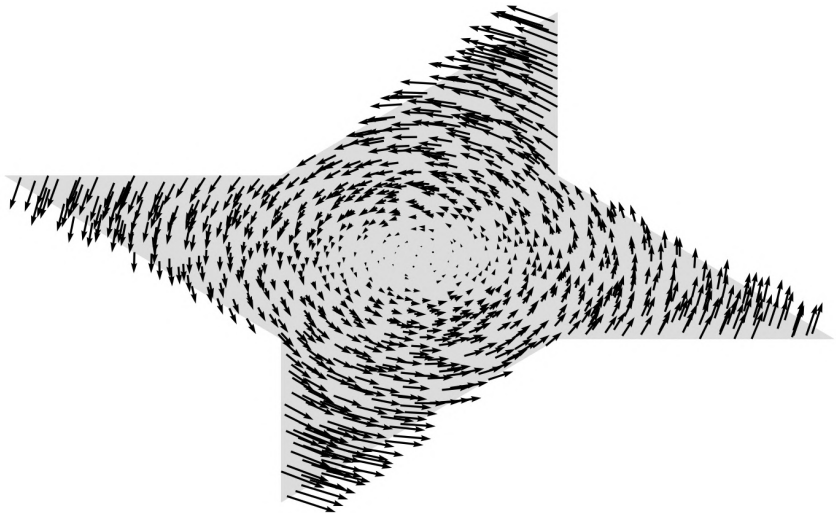


Figure 9.29: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.30. Vectors are scaled by 25%.

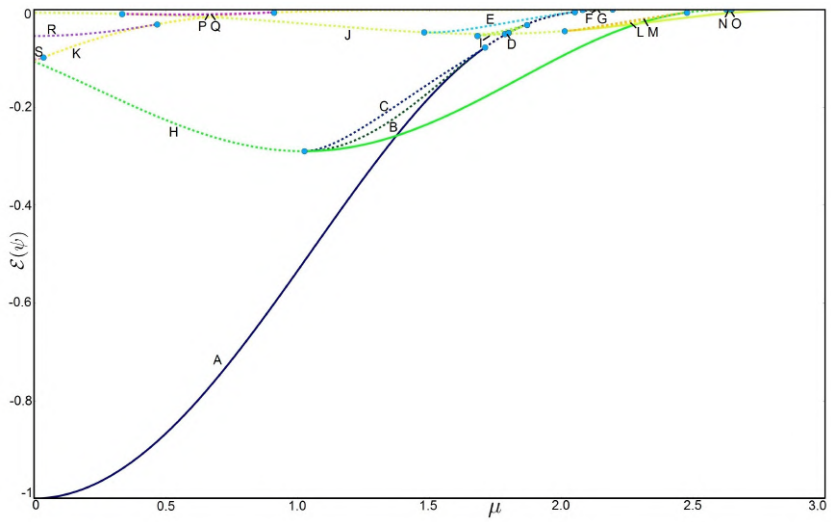
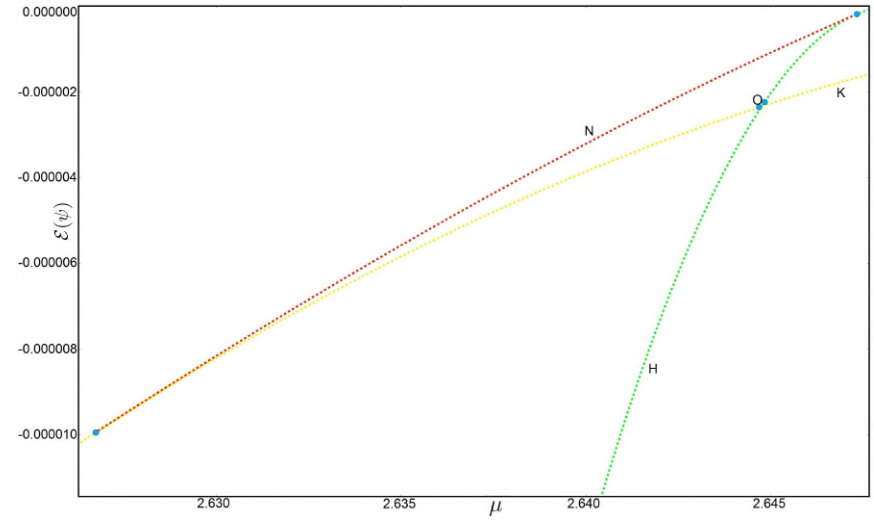
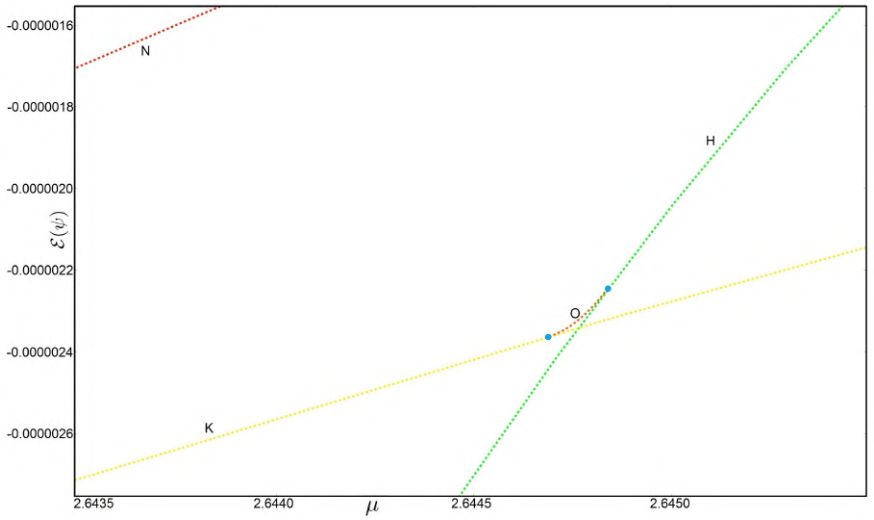


Figure 9.30: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to a C_4-symmetric, star-shaped material (as described in section 9.5.4). The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.29. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.32.

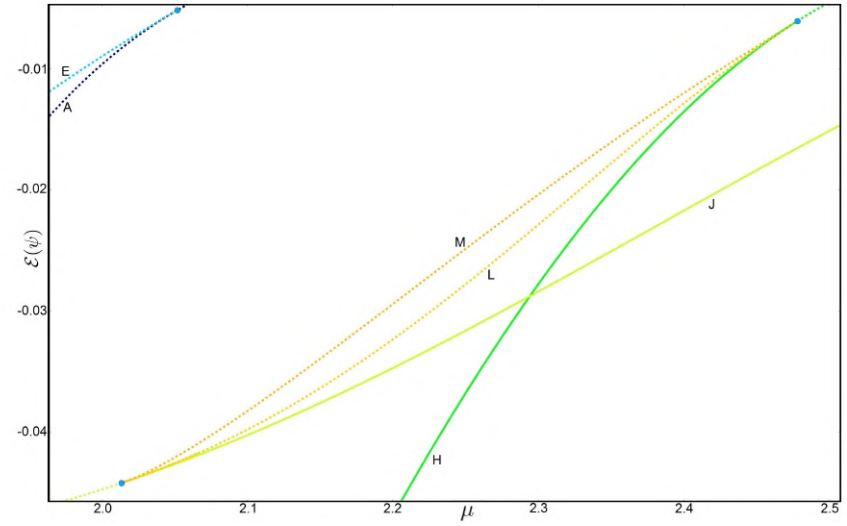


(a) Close-up around curves N and O

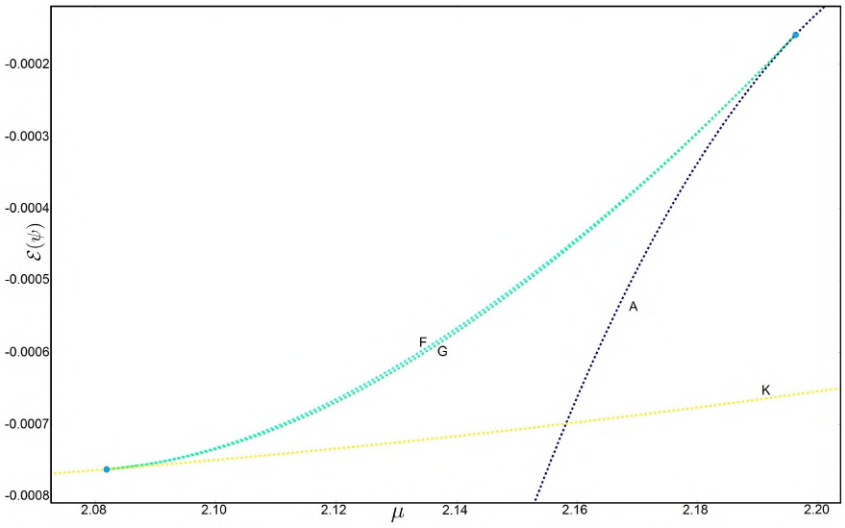


(b) Close-up around curve O

Figure 9.31: Several close-ups of the connected solution landscape of figure 9.30. Representative solutions for the curves are given in figure 9.32.

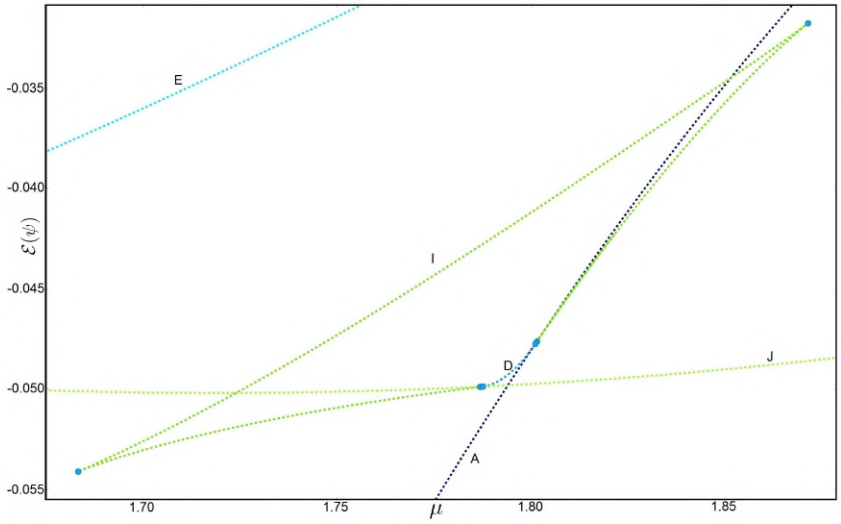


(c) Close-up around curves L and M

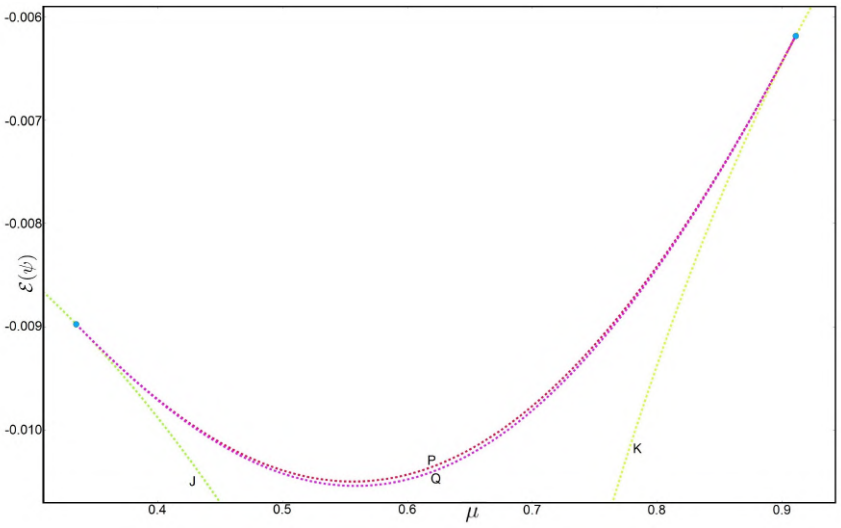


(d) Close-up around curves F and G

Figure 9.31: Several close-ups of the connected solution landscape of figure 9.30 (cont.).

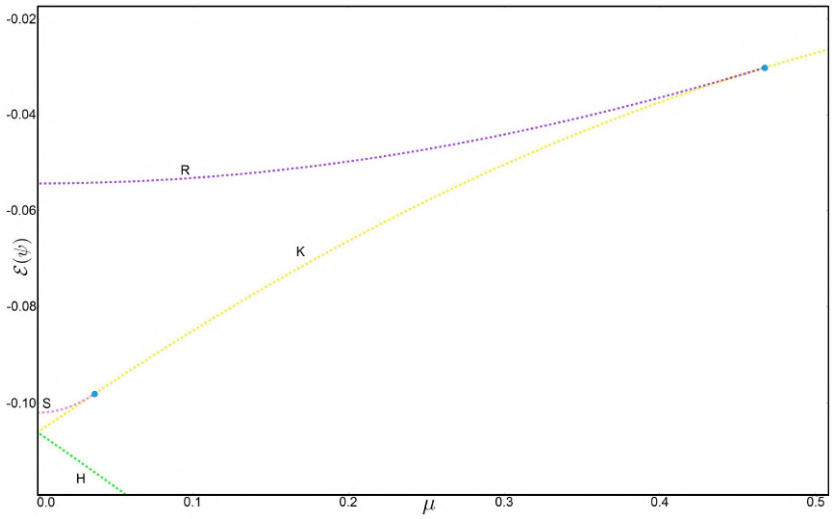


(e) Close-up around curves D and I



(f) Close-up around curves P and Q

Figure 9.31: Several close-ups of the connected solution landscape of figure 9.30 (cont.).



(g) Close-up around curves R and S

Figure 9.31: Several close-ups of the connected solution landscape of figure 9.30 (cont.).

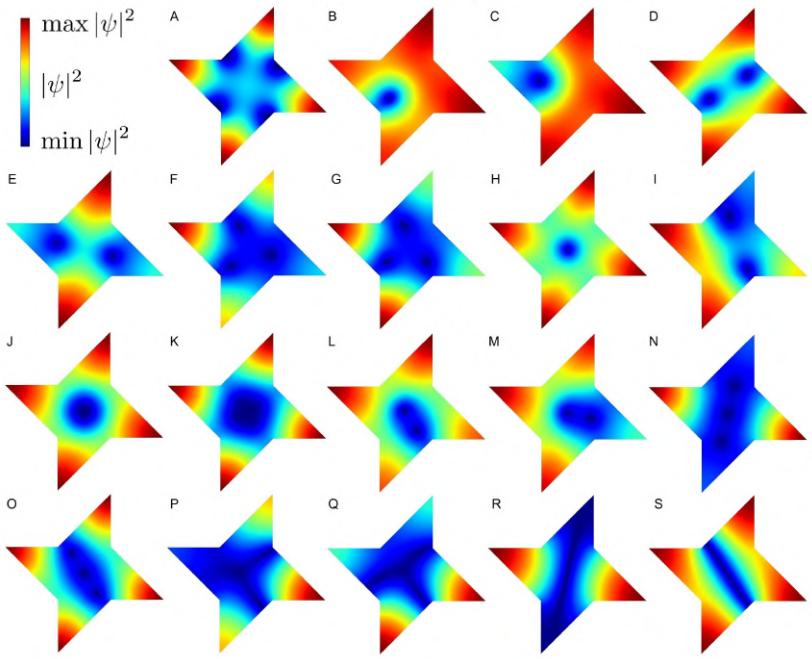


Figure 9.32: Representative solutions for the different curves of figures 9.30 and 9.31.

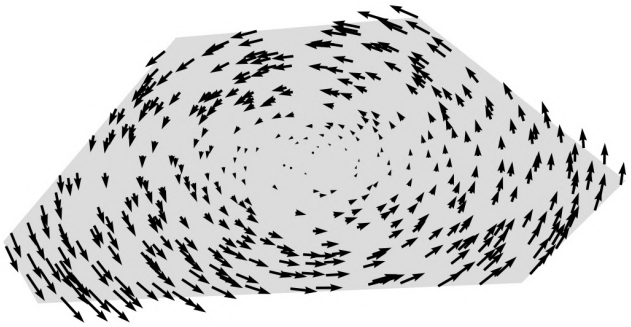


Figure 9.33: Magnetic vector potential $\mathbf{A}_0$ (given as a vector field) used in (2.20) for the construction of the connected solution landscape in figure 9.34. Vectors are scaled by 25%.

Note that it was not possible to apply algorithm 7.6 (based on the equivariant branching lemma, see page 234) to construct tangent directions for these points, since the material only exhibits rotational symmetry. Except for the 20 branch points, two turning points were also found, both belonging to curve I.

Four of the 19 generated curves (A, H, J and K) consist of solutions that are invariant under the actions of the entire group C_4. A detailed analysis of figure 9.30 is given in Wouters [110].

9.5.5 Material with an irregular shape

We finally consider a material with an irregular shape, not containing any symmetry. The shape is constructed by considering a two-dimensional grid and connecting the dots at $(-2.25, -1)$, $(-2, -1.75)$, $(1.5, -1.5)$, $(2.25, -0.25)$, $(0.75, 1.75)$ and $(-1, 1.5)$ (scaled in units of the coherence length ξ). The magnetic vector potential $\mathbf{A}_0$ is given by figure 9.33.

The Ginzburg-Landau equation is only invariant under the actions of S^1 for this problem. Bifurcation points are approximated by the Newton step length adaptation method (algoirthm 6.6 on page 198). Due to the absence of discrete symmetries, only branch points that induce a Jacobian with a 1-dimensional kernel appear in the connected solution landscape. It is sufficient to apply algorithm 7.4 (page 233) for the construction of tangent directions.

Contrary to the previous shapes, we use the point

$$(\tilde{\psi}^{(0)}, \tilde{\mu}^{(0)}) = (0, 0)$$

as a starting value for the PyNCT package. This point is part of the trivial solution curve $\psi = 0$, $\mu \in \mathbb{R}$. Computation of this curve was omitted in the previous examples by including the condition (9.4). We do not use conditions (9.3) and (9.4), instead

$$C_3 : \mathbb{C}^n \times \mathbb{R} \to \{True, False\} : (\psi^{(k)}, \mu^{(k)}) \to \begin{cases} True & \mu^{(k)} < 0 \text{ or } \mu^{(k)} > 3, \\ False & \text{otherwise,} \end{cases} \tag{9.5}$$

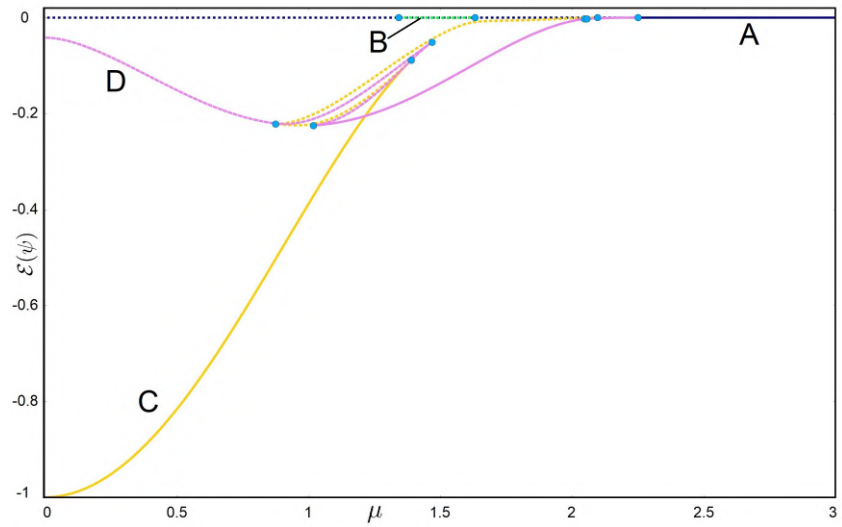
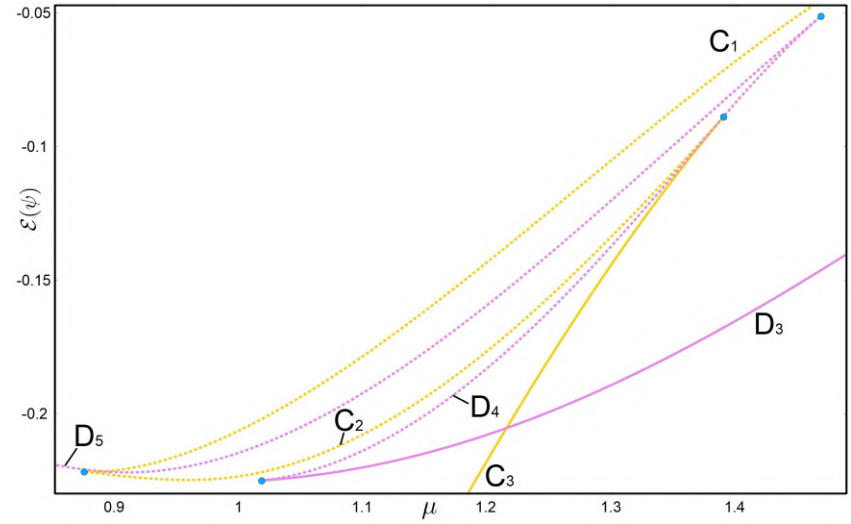


Figure 9.34: Connected solution landscape for the extreme type-II Ginzburg-Landau equation applied to an irregular material (as described in section 9.5.5). The material is subject to a homogeneous magnetic field with varying strength μ, the corresponding magnetic vector potential for $\mu = 1$ is described by figure 9.33. $\mathcal{E}$ represents the part (9.2) of the Gibbs energy. Solid (dashed) lines represent stable (unstable) solutions. Blue dots indicate bifurcation points. Representative solutions for the curves are given in figure 9.36.

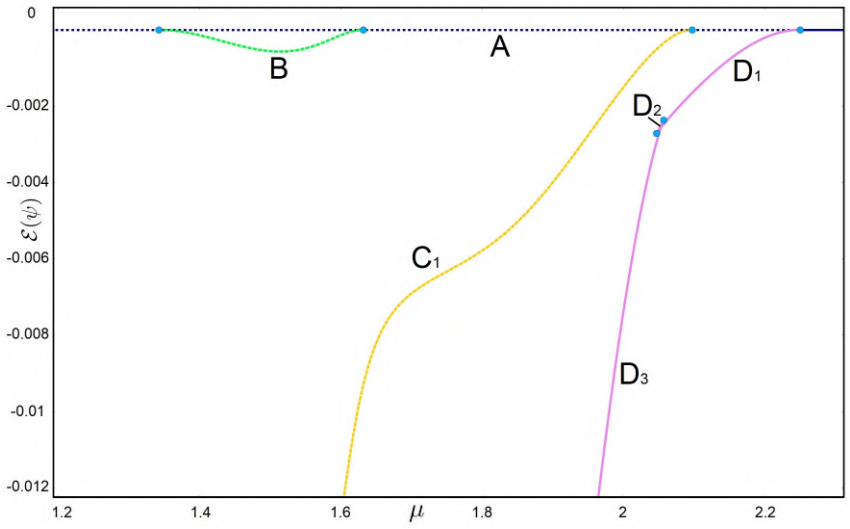
is used as an extra stopping criterion for pseudo-arclength continuation, it is applied after each step to the last generated point $(\psi^{(k)}, \mu^{(k)})$ (for a certain $k \in \mathbb{N}$) of the curve. This condition prevents the magnetic field strength from leaving the range $[0, 3]$.

Application of the PyNCT package to the problem yields the connected solution landscape in figure 9.34. Several close-ups of the landscape are provided by figure 9.35. Representative solutions for each of the generated curves are given in figure 9.36.

The generated solution landscape consists of 4 different curves, including the one (curve A) containing trivial solutions of the form $\psi = 0$, $\mu \in \mathbb{R}$. The other branches are connected to curve A, there are no other connections between them. A total of 4 branch points (all belonging to curve A) and 6 turning points were approximated.

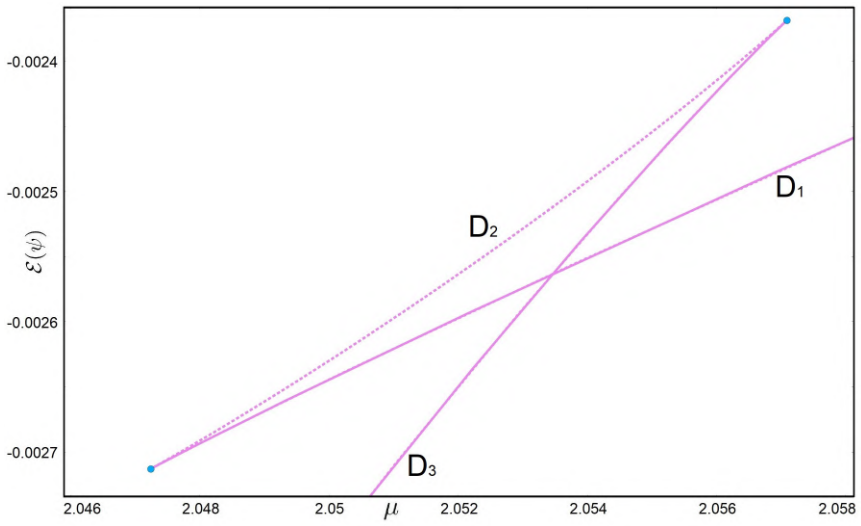


(a) Close-up for $0.88 \lesssim \mu \lesssim 1.5$, $-0.23 \lesssim \mathcal{E}(\psi) \lesssim -0.05$



(b) Close-up for $1.2 \lesssim \mu \lesssim 2.23$, $-0.012 \lesssim \mathcal{E}(\psi) \leq 0$

Figure 9.35: Several close-ups of the connected solution landscape of figure 9.34. Representative solutions for the curves are given in figure 9.36.



(c) Close-up for $2.046 \lesssim \mu \lesssim 2.058$, $-0.00272 \lesssim \mathcal{E}(\psi) \lesssim -0.00237$

Figure 9.35: Several close-ups of the connected solution landscape of figure 9.34 (cont.).

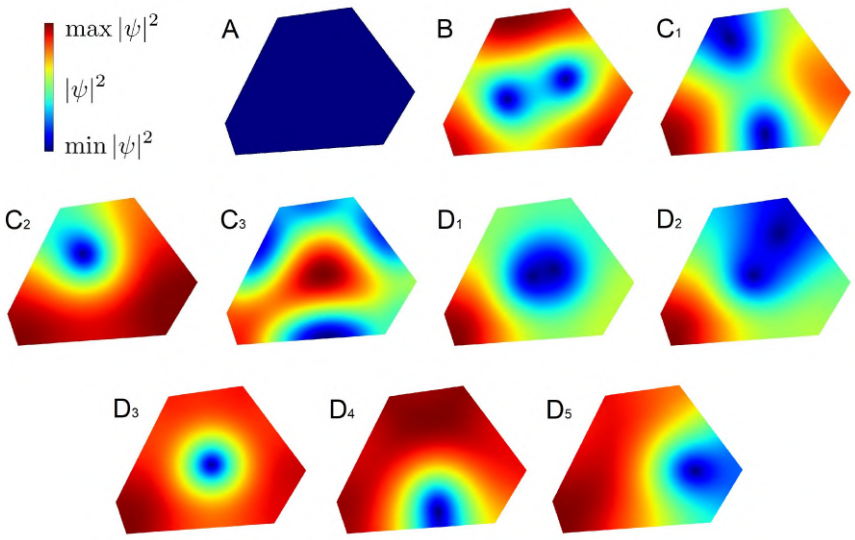


Figure 9.36: Representative solutions for the different curves of figures 9.34 and 9.35.

CHAPTER 10

Conclusions & outlook

"Just because something works doesn't mean it can't be improved."

– Shuri –
Black Panther

10.1 Conclusions

An essential element in the study of dynamical systems is to derive its equilibria: states of the system that remain constant over time. These equilibria often depend in an intricate way on physical parameters like the temperature or the strength of an applied magnetic field. Instead of solely deriving the equilibria of a given dynamical system, the thesis focussed on the problem of analysing the dependency of these steady states to the systems parameters. Mathematically this analysis is performed by constructing a connected solution landscape (bifurcation diagram) for the considered dynamical system.

The aim of the thesis was to analyse and develop robust numerical methods that generate a complete solution landscape of interconnected curves for dynamical systems described by a partial differential equation, where we limited ourselves to systems for which the equilibrium equation contains a Hermitian partial Jacobian.

Applying numerical continuation to a problem derived from a partial differential equation typically results in large, sparse, systems of equations that need to be analysed. Methods based on dense linear algebra are not practicable for this purpose, due to unacceptable computational and memory requirements. To analyse the systems, it is essential to use techniques from sparse linear algebra, like Krylov methods.

All of the numerical methods discussed in this thesis are applicable to such large systems. Either they are based on dense linear algebra, but are only applied to a reduced system of equations (e.g. the nonlinear conjugate gradients algorithm, see section 3.4), or they are based on sparse linear algebra (e.g. the split Newton method with mixed terms, see section 4.8). New numerical methods, developed in the thesis, that are based on sparse linear algebra never use the matrix or tensor form of operators. Instead only the matrix-vector (or tensor-vectors) product is required, exploiting the sparsity structure of the problems.

Automatic exploration techniques are an essential part of the numerical continuation algorithm. Given an initial (guess of a) solution, the goal is to generate the *complete* interconnected solution landscape, instead of just a single curve. Automatic exploration is realized by two steps. First, the bifurcation points of the dynamical system need to be identified during the approximation of its curves. However, the singularity of the Jacobian in these points typically gives rise to issues in the numerical algorithms that have this purpose.

Two different strategies for approximating bifurcations were discussed in the thesis. The first strategy consists of constructing an extended nonlinear system, which is then solved by the Newton method. Though algorithms based on this approach relatively yield fast results, it is unpredictable whether the method will converge due to the Jacobian's singularity.

A more robust strategy is given by the Newton step length adaptation algorithm. This method is based on the pseudo-arclength continuation algorithm, using approximate eigenvalues of the system's Jacobian to choose the used step length in each iteration. A crucial challenge for this method is implied by the singularity of the system's Jacobian at the searched bifurcation point. Due to this singularity, divergence is typically observed when solving the nonlinear systems of the pseudo-arclength continuation method by the standard Newton algorithm.

For this purpose a new class of Newton methods was derived in the thesis. One of these methods (the split block Newton method with mixed terms, described in section 5.8) is able to approximately solve the required systems in an efficient manner, even near and in points where the Jacobian is singular. The derived algorithms are entirely based on sparse linear algebra. They are applicable to large-scale systems, like the ones considered in the thesis. Compared to other methods with the same purpose, like the accelerated inexact Newton [42] and tensor method [4], our algorithm attains an improved accuracy.

The second step of automatic exploration is to construct for each bifurcation point, identified in the first step, the tangent directions of solution curves emerging from these points. If no symmetries are present in the underlying dynamical system, this problem is easily solved by applying the algebraic branching equation. For systems with symmetries typically bifurcations arise for which application of this equation is not sufficient.

We discussed two different strategies for constructing tangent directions, when the underlying system contains a two-dimensional (e.g. dihedral) symmetry. The first strategy is based on a Lyapunov-Schmidt reduction. Starting from an algorithm for the construction of tangent directions in Mei and Schwarzer [73], we derived an adjusted method to use for large-scale systems. Though effective, this adjusted algorithm requires multiple solves of a linear problem, linked to the underlying symmetry of the dynamical system.

We also proposed a second strategy, where prior knowledge on these underlying symmetries is exploited to omit solving additional linear problems. This alternative strategy strongly reduces the amount of computational work, but is not applicable to certain dynamical systems, depending on the kind of symmetry considered. Both strategies successfully construct tangent directions for the two-dimensional symmetry case.

As explained in the thesis, the two discussed steps (approximating bifurcation points and constructing tangent directions) are sufficient requisites for the execution of automatic exploration in the construction of connected solution landscapes. The results in chapter 9 emphasize the effectiveness of the derived algorithms.

An implementation of an algorithm that constructs a connected solution landscape for a dynamical system, given a single initial point, has been made in the Python package PyNCT. The package approximates both stable and unstable equilibria. Though not physically feasible, unstable equilibria contain important information on transitions between stable states of the system. This was indicated by example 1.1 in the introduction. Solution landscapes are helpful for the derivation of this information as well, as was explicated by example 1.2.

Contrary to other tools, *the PyNCT package is entirely based on sparse linear algebra*, allowing large-scale systems like the Ginzburg-Landau equations to be considered. Furthermore, prior knowledge on symmetries of the system can be provided. This knowledge is used to reduce computational work in some of the algorithms. As indicated by the results in chapter 9, automatic exploration techniques contained in the package make it possible to generate connected solution landscapes with a high amount (40+) of different curves. Users of PyNCT only need to provide a single state of the dynamical system, a connected solution landscape is then generated from this point. Further user interference is minimal, only consisting of the optional retuning of internal parameters (see section 8.4).

10.2 Outlook

Though we believe to have succeeded in the goal of deriving the necessary numerical tools, used to generate connected solution landscapes for large-scale dynamical systems, a lot of new questions and problems arose during our research. We give a brief sketch of possible future work, based on the results derived in this thesis.

- In chapters 4 and 5 we derived a new class of Newton-Krylov methods, applicable for nonlinear systems that contain an ill-conditioned Jacobian near the searched solution. Though numerical experiments were executed to show the efficiency of the methods, a theoretical analysis on when to expect convergence should still be performed.

- This new class of Newton-Krylov methods was especially derived to use for the approximation of bifurcation points. For this purpose the combination of the methods with block elimination was discussed in detail (see chapter 5). Nonlinear systems with an ill-conditioned Jacobian however occur in other situations as well. Examples include conservation laws in stiff systems of ordinary differential equations and unconstrained minimization problems arising from data fitting [94]. Application of the derived Newton methods to these problems might be an interesting topic for future research.

- The current implementation of PyNCT is focussed on robustness. Though based on sparse linear algebra and hence applicable to large-scale systems, many of the algorithms could be replaced by computationally more efficient counterparts. For example, solutions of linear problems with a self-adjoint operator could be approximated by the MINRES algorithm [77] instead of GMRES. MINRES requires less computations, and a lower memory storage cost. GMRES is on the other hand more robust.

- Other adjustments that should be made to the version of PyNCT concern parallelization. We already discussed this subject in section 8.5. A parallel implementation would be crucial when even larger-scale problems are considered, appearing for example in three-dimensional simulations of superconductors and other dynamical systems. Though the methods derived in the thesis are still valid for these kinds of problems, a parallel version of the code will be required to maintain a tolerable computational time. Some work on this problem is currently being done by linking the PyNCT package with PHIST [104], a Pipelined Hybrid-parallel Iterative Solver Toolkit.

- Currently a lot of different internal parameters (see section 8.4) need to be provided to generate a connected solution landscape, and optimally these should be retuned whenever a new curve is approximated. Further work is required to determine these in a robust manner. Although, for the examples presented in chapter 9, the desired tolerances were always achieved, they require a new tuning when applied to other dynamical systems. In the current version of PyNCT, this is required for different samples in the same dynamical system as well, depending on the sample's discretization level. Robust internal parameters, that work automatically for various discretizations, would be helpful in the future. The parameters provided by users of PyNCT should not be dependent on the discretization level of the underlying problem.

- Though the current version of PyNCT is able to generate connected solution landscapes, it does not contain any bifurcation tracking algorithms. These algorithms are used to determine how the location of a bifurcation is perturbed when a second physical parameter changes, and form an essential part of other numerical continuation software like AUTO [31], MatCont [30] and LOCA [90].

- The dynamical systems in the thesis were restricted to ones where the equilibrium equation contains a Hermitian Jacobian operator. Though a lot of mathematical models have this property, future work could be done to extend PyNCT for more general systems. More complicated bifurcation points, like the Hopf bifurcation [72, 95], need to be analysed for this extention. This requires further numerical techniques, like time step methods and algorithms that identify periodic orbits.

- Only materials with a two-dimensional shape were considered for the dynamical systems of chapter 2. The extension towards three-dimensional ones provides an interesting topic of future research. We expect that the

provided methods for detection and approximation of bifurcation points are still straightforward applicable. However, the construction of the tangent directions to emerging solution branches will require further efforts when the material contains a three-dimensional symmetry (e.g. a cube). For such materials we expect bifurcations to arise in which the Jacobian contains a three-dimensional kernel, rendering algorithm 7.6 useless.

We believe the analysis in sections 7.3 and 7.4 can be extended to yield an algorithm that successfully constructs the tangent directions for this case. Due to the increased amount of (approximate) null vectors, a similar algorithm to 7.6 is expected to require even more linear systems to be solved, and the form of the term (7.28) is expected to become more complex as well. We however believe that the method based on the equivariant branching lemma (see section 7.5), which strongly reduced the computational work of algorithm 7.6, can be extended as well.

- Finally, the methods presented in the thesis could be applied to more complicated superconducting systems. Some work has already been done to apply PyNCT to superconducting islands and to the multicomponent Ginzburg-Landau equations. Initial results for these applications are expected to be derived in the near future.

Bibliography

[1] A. Abrikosov. On the magnetic properties of superconductors of the second group. *Sov. Phys. JETP*, 5(6):1174–1182, 1957.

[2] G. Adomian. The diffusion–Brusselator equation. *Comput. Math. Appl.*, 29(5):1–3, 1995.

[3] E. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[4] B. Bader. Tensor–Krylov methods for solving large-scale systems of nonlinear equations. *SIAM J. Numer. Anal.*, 43(3):1321–1347, 2005.

[5] B. Bader and R. Schnabel. Curvilinear linesearch for tensor methods. *Siam J. Sci. Comput.*, 25(2):604–622, Nov 2003.

[6] B. Bader and R. Schnabel. On the performance of tensor methods for solving ill-conditioned problems. *SIAM J. Sci. Comput.*, 29(6):2329–2351, Jan 2007.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000.

[8] Z. Bai and R. Li. Stability and accuracy assessments. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 105–107. SIAM, 2000.

[9] G. Berdiyorov, A. Hernandez, and F. Peeters. Confinement effects on intermediate-state flux patterns in mesoscopic type–I superconductors. *Phys. Rev. Lett.*, 103, Dec 2009.

[10] W. Beyn et al. Numerical continuation, and computation of normal forms. In *Handbook of dynamical systems*, volume 2, pages 149–219. Elsevier, 2002.

[11] A. Bouaricha. Tensor-Krylov methods for large nonlinear equations. *Comput. Optim. Appl.*, 5(3):207–232, May 1996.

[12] A. Bouaricha and R. Schnabel. Algorithm 768: TENSOLVE: A software package for solving systems of nonlinear equations and nonlinear least–squares problems using tensor methods. *ACM Trans. Math. Softw.*, 23(2):174–195, Jun 1997.

[13] E. Boutyour, H. Zahrouni, M. Potier-Ferry, and M. Boudi. Bifurcation points and bifurcated branches by an asymptotic numerical method and Padé approximants. *Int. J. Numer. Meth. Eng.*, 60(12):1987–2012, 2004.

[14] S. Buhmiler, N. Krejić, and Z. Lužanin. Practical quasi-Newton algorithms for singular nonlinear systems. *Numer. Algorithms*, 55(4):481–502, Dec 2010.

[15] A. Champneys and B. Sandstede. Numerical computation of coherent structures. In *Numerical continuation methods for dynamical systems*, pages 331–358. Springer, 2007.

[16] T. Chan. Deflated decomposition of solutions of nearly singular systems. *SIAM J. Numer. Anal.*, 21(4):738–754, 1984.

[17] T. Chan. Deflation techniques and block–elimination algorithms for solving bordered singular systems. *SIAM J. Sci. Stat. Comput.*, 5(1):121–134, 1984.

[18] T. Chan and D. Resasco. Generalized deflated block–elimination. *SIAM J. Numer. Anal.*, 23(5):913–924, Oct 1986.

[19] S. Chandrasekhar. *An introduction to the study of stellar structure*, volume 2. Courier Corporation, 1958.

[20] S. Chapman, Q. Du, M. Gunzburger, and J. Peterson. Simplified Ginzburg–Landau models for superconductivity valid for high kappa and high fields. *Adv. Math. Sci. Appl*, 5(1):193–218, 1995.

[21] E. Charalampidis, P. Kevrekidis, and P. Farrell. Computing stationary solutions of the two–dimensional Gross–Pitaevskii equation with deflated continuation. *Commun. Nonlinear Sci. Numer Simul.*, 54:482–499, 2018.

[22] S. Cheng and N. Higham. A modified Cholesky algorithm based on a symmetric indefinite factorization. *SIAM J. Matrix Anal. Appl.*, 19(4):1097–1110, 1998.

[23] R. Córdoba et al. Magnetic field-induced dissipation-free state in superconducting nanostructures. *Nat. Commun.*, 4(1437), 2013.

[24] G. Cramer. *Introduction à l'analyse des lignes courbes algébriques.* chez les frères Cramer et C. Philibert, 1750.

[25] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat–conduction type. *Adv. Comput. Math.*, 6(1):207–226, Dec 1996.

[26] E. Dancer. The G–invariant implicit function theorem in infinite dimensions II. *Proc. Roy. Soc. Edinburgh*, 102:211–220, Jan 1986.

[27] M. Dellnitz and B. Werner. Computational methods for bifurcation problems with symmetries - with special attention to steady state and Hopf bifurcation points. *J. Comput. Appl. Math.*, 26(1):97–123, 1989.

[28] P. Deuflhard. A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting. *Numer. Math.*, 22(4):289–315, Aug 1974.

[29] P. Deuflhard. *Newton methods for nonlinear problems: Affine invariance and adaptive algorithms*. Springer, 2006.

[30] A. Dhooge, W. Govaerts, and Y. Kuznetsov. MATCONT: a Matlab package for numerical bifurcation analysis of ODEs. *ACM Trans. Math. Software*, 29(2):141–164, 2003.

[31] E. Doedel. Auto: A program for the automatic bifurcation analysis of autonomous systems. *Cong. Numer.*, 30:265–284, 1981.

[32] E. Doedel. Lecture notes on numerical analysis of nonlinear equations. In *Numerical Continuation Methods for Dynamical Systems: Path following and boundary value problems*, pages 1–49. Springer, 2007.

[33] M. Dorojevets and Z. Chen. Fast pipelined storage for high–performance energy–efficient computing with superconductor technology. In *12th International Conference Expo on Emerging Technologies for a Smarter World (CEWIT)*, pages 1–6, Oct 2015.

[34] D. Draelants, D. Avitabile, and W. Vanroose. Localized auxin peaks in concentration-based transport models of the shoot apical meristem. *J. R. Soc. Interface*, 12, May 2015.

[35] D. Draelants, J. Broeckhove, G. Beemster, and W. Vanroose. Numerical bifurcation analysis of pattern formation in a cell based auxin transport model. *J. Math. Biol.*, 67, Sep 2012.

[36] D. Draelants, P. Klosiewicz, J. Broeckhove, and W. Vanroose. Solving general auxin transport models with a numerical continuation toolbox in Python: PyNCT. *Lect. Notes Comput. Sci.*, 9271:211–225, 2015.

[37] Q. Du, M. Gunzburger, and J. Peterson. Analysis and approximation of the Ginzburg–Landau model of superconductivity. *SIAM Rev.*, 34(1):54–81, Mar 1992.

[38] L. Embon et al. Imaging of super–fast dynamics and flow instabilities of superconducting vortices. *Nat. Commun.*, 8(85), 2017.

[39] J. Fan and Y. Yuan. On the quadratic convergence of the Levenberg-Marquardt method without nonsingularity assumption. *Computing*, 74(1):23–39, Feb 2005.

[40] D. Feng and T. Pulliam. Tensor-GMRES method for large systems of nonlinear equations. *SIAM J. Optim.*, 7(3):757–779, Aug 1997.

[41] T. Filippov et al. 20 GHz operation of an asynchronous wave-pipelined RSFQ arithmetic–logic unit. *Physics Procedia*, 36:59–65, 2012.

[42] D. Fokkema, G. Sleijpen, and H. van der Vorst. Accelerated inexact Newton schemes for large systems of nonlinear equations. *SIAM J. Sci. Comput.*, 19(2):657–674, 1998.

[43] B. Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput.*, 51(184):699–706, 1988.

[44] K. Fossheim and A. Sudboe. *Superconductivity: Physics and Applications.* Wiley, 2004.

[45] A. Gaul, M. Gutknecht, J. Liesen, and R. Nabben. A framework for deflated and augmented Krylov subspace methods. *arXiv e-prints*, page arXiv:1206.1506, Jun 2012.

[46] A. Gaul and N. Schlömer. Preconditioned recycling Krylov subspace methods for self-adjoint problems. *Electron. Trans. Numer. Anal.*, 44:522–547, 2015.

[47] M. Golubitsky, D. Schaeffer, and I. Stewart. *Singularities and groups in bifurcation theory, volume II.* Springer-Verlag New York, 1985.

[48] M. Golubitsky and I. Stewart. *The symmetry perspective.* Birkhäuser, 2002.

[49] B. Goodman. Type II superconductors. *Rep. Prog. Phys.*, 29:445–483, 1966.

[50] W. Govaerts. Stable solvers and block elimination for bordered systems. *SIAM J. Matrix Anal. Appl.*, 12(3):469–483, Jul 1991.

[51] R. Groh, D. Avitabile, and A. Pirrera. Generalised path–following for well–behaved nonlinear structures. *Comput. Methods Appl. Mech. Eng.*, 331:394–426, 2018.

[52] G. Gundersen and T. Steihaug. On large–scale unconstrained optimization problems and higher order methods. *Optim. Methods Softw.*, 25(3):227–258, Jun 2010.

[53] J. Gutiérrez and M. Hernández. An acceleration of Newton's method: Super-Halley method. *Appl. Math. Comput.*, 117(2–3):223–239, Jan 2001.

[54] D. Holmes, A. Ripple, and M. Manheimer. Energy–efficient superconducting computing–power budgets and requirements. *IEEE Trans. Appl. Supercond.*, 23(3), Jun 2013.

[55] R. Hoyle. *Pattern formation.* Cambridge University Press, 2006.

[56] J. Hueso, E. Martínez, and J. Torregrosa. Modified Newton's method for systems of nonlinear equations with singular Jacobian. *J. Comput. Appl. Math.*, 224(1):77–83, 2009.

[57] J. Huitfeldt. Nonlinear eigenvalue problems - prediction of bifurcation points and branch switching. Technical report, Chalmers University of Technology and the University of Göteborg, 1991.

[58] J. Hull. *Options, futures, and other derivatives.* Pearson Prentice Hall, 6. ed., pearson internat. ed edition, 2006.

[59] D. Idczak. A global implicit function theorem and its applications to functional equations. *Discrete Cont. Dyn–B.*, 19, Jan 2014.

[60] J. Jacobsen and K. Schmitt. The Liouville–Bratu–Gelfand problem for radial operators. *J. Differ. Equations*, 184(1):283–298, 2002.

[61] H. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. In *Applications of bifurcation theory*, pages 359–384. Academic Press, 1977.

[62] H. Keller. *Lectures on numerical methods in bifurcation problems.* Springer–Verlag, 1986.

[63] C. Kelley. *Iterative Methods for Linear and Nonlinear equations.* SIAM, 1995.

[64] C. Kittel. *Introduction to Solid State Physics.* Wiley, 8th edition, 2004.

[65] R. Kleiner, D. Koelle, F. Ludwig, and J. Clarke. Superconducting quantum interference devices: State of the art and applications. *Proc. IEEE*, 92(10):1534–1548, 2004.

[66] R. Kouhia and M. Mikkola. Tracing the equilibrium path beyond compound critical points. *Int. J. Numer. Meth. Eng.*, 46(7):1049–1074, 1999.

[67] P. Lanzkron, D. Rose, and J. Wilkes. An analysis of approximate nonlinear elimination. *SIAM J. Sci. Comput.*, 17:538–559, 1993.

[68] R. Lehoucq and D. Sorensen. Implicitly restarted Arnoldi method. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 166–185. SIAM, 2000.

[69] D. Li, M. Fukushima, L. Qi, and N. Yamashita. Regularized Newton methods for convex minimization problems with singular solutions. *Comput. Optim. Appl.*, 28:131–147, 2004.

[70] P. Lynch. The origins of computer weather prediction and climate modeling. *J. Comput. Phys.*, 227(7):3431–3444, Mar 2008.

[71] S. MacLachlan and C. Oosterlee. Algebraic multigrid solvers for complex–valued matrices. *SIAM J. Sci. Comput.*, 30(3):1548–1571, 2008.

[72] Z. Mei. *Numerical bifurcation analysis for reaction–diffusion equations.* Springer, 2000.

[73] Z. Mei and A. Schwarzer. Scaling solution branches of one-parameter bifurcation problems. *J. Math. Anal. Appl.*, 204:102–123, 1996.

[74] J. Miller and J. Claycomb. Classical and high temperature superconductivity. In *Physics of Novel Materials: Proceedings of the 10th physics summer school*, pages 43–85. World Scientific Publishing Co. Pte. Ltd., 1999.

[75] G. Moore. The numerical treatment of non-trivial bifurcation points. *Numer. Funct. Anal. Optim.*, 2(6):441–472, 1980.

[76] A. Murphy, D. Averin, and A. Bezryadin. Nanoscale superconducting memory based on the kinetic inductance of asymmetric nanowire loops. *New J. Phys.*, 19, 2017.

[77] C. Paige and M. Saunders. Solutions of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[78] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer–Verlag, 1985.

[79] A. Ramm. A simple proof of the Fredholm alternative and a characterization of the Fredholm operators. *Am. Math. Mon.*, 108(9):855–860, 2001.

[80] L. Recke and D. Peterhof. Abstract forced symmetry breaking and forced frequency locking of modulated waves. *J. Differ. Equations*, 144(2):233–262, 1998.

[81] C. Reeves and R. Fletcher. Function minimization by conjugate gradients. *Comput. J.*, 7(2):149–154, Jan 1964.

[82] R. Riaza and P. Zufiria. Discretization of implicit ODEs for singular root–finding problems. *J. Comput. Appl. Math.*, 140:695–712, Mar 2002.

[83] O. Richardson. *The emission of electricity from hot bodies.* Longmans, Green and Company, 1921.

[84] J. Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry.* Springer Publishing Company, Incorporated, 1st edition, 2011.

[85] A. Ruhe. Lanczos method. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 56–66. SIAM, 2000.

[86] Y. Saad. Arnoldi method. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 161–166. SIAM, 2000.

[87] Y. Saad. An introduction to iterative projection methods. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 37–43. SIAM, 2000.

[88] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[89] P. Salas, L. Giraud, Y. Saad, and S. Moreau. Spectral recycling strategies for the solution of nonlinear eigenproblems in thermoacoustics. *Numer. Linear Algebra Appl.*, 22(6):1039–1058, 2015.

[90] A. Salinger et al. Loca 1.1 library of continuation algorithms: theory and implementation manual. Technical Report SAND2002-0396, Sandia National Laboratories, Albuquercue, NM, 2002.

[91] D. Sattinger. *Branching in the presence of symmetry.* Society for Industrial and Applied Mathematics Philadelphia, Pa, 1983.

[92] N. Schlömer, D. Avitabile, and W. Vanroose. Numerical bifurcation study of superconducting patterns on a square. *SIAM J. Appl. Dyn. Syst.*, 11:447–477, 2012.

[93] N. Schlömer and W. Vanroose. An optimal linear solver for the Jacobian system of the extreme type–II Ginzburg–Landau problem. *J. Comput. Phys.*, 234:560–572, 2013.

[94] R. Schnabel and P. Frank. Tensor methods for nonlinear equations. *SIAM J. Numer. Anal.*, 21(5):815–843, 1984.

[95] R. Seydel. *From equilibrium to chaos: practical bifurcation and stability analysis.* Springer, New York, NY, 1988.

[96] C. Shen, X. Chen, and Y. Liang. A regularized Newton method for degenerate unconstrained optimization problems. *Optim. Lett.*, 6:1913–1933, Dec 2012.

[97] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[98] G. Shroff and H. Keller. Stabilization of unstable procedures: the recursive projection method. *SIAM J. Numer. Anal.*, 30(4):1099–1120, 1993.

[99] G. Sleijpen and H. van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. and Appl.*, 17(2):401–425, Apr 1996.

[100] G. Sleijpen and H. van der Vorst. Jacobi–Davidson methods. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 88–105. SIAM, 2000.

[101] G. Sleijpen, H. van der Vorst, and E. Meijerink. Efficient expansion of subspaces in the Jacobi–Davidson method for standard and generalized eigenproblems. *Electron. Trans. Numer. Anal.*, 7:75–89, 1998.

[102] T. Steihaug and S. Suleiman. Rate of convergence of higher order methods. *Appl. Numer. Math.*, 67:230–242, 2013.

[103] G. Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, 1988.

[104] J. Thies et al. PHIST: a pipelined, hybrid-parallel iterative solver toolkit. *ACM Trans. Math. Software*, 2019.

[105] M. Tinkham. *Introduction to Superconductivity*. Dover Publications, 2004.

[106] E. Twizell, A. Gumel, and Q. Cao. A second-order scheme for the "Brusselator" reaction-diffusion system. *J. Math. Chem.*, 26:297–316, Apr 2000.

[107] H. Uecker, D. Wetzel, and J. Rademacher. pde2path - a Matlab package for continuation and bifurcation in 2D elliptic systems. *NM-TMA*, 7(1):58–106, 2014.

[108] H. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, 48:327–341, 1993.

[109] B. Werner and A. Spence. The computation of symmetry–breaking bifurcation points. *Siam J. Numer. Anal.*, 21:388–399, Apr 1984.

[110] M. Wouters and W. Vanroose. Automatic exploration techniques for the numerical bifurcation study of the Ginzburg–Landau equation. *arXiv e-prints*, page arXiv:1903.02377, 2019.

[111] N. Yamamoto. A note on solutions of nonlinear equations with singular Jacobian matrices. *J. Math. Anal. Appl.*, 123(1):152–160, 1987.

[112] N. Yamashita and M. Fukushima. On the rate of convergence of the Levenberg-Marquardt method. In G. Alefeld and X. Chen, editors, *Topics in Numerical Analysis*, pages 239–249, Vienna, 2001. Springer Vienna.

[113] D. Young, W. Huffman, R. Melvin, C. Hilmes, and F. Johnson. Nonlinear elimination in aerodynamic analysis and design optimization. In *Large–Scale PDE–Constrained Optimization*, pages 17–43. Springer Berlin Heidelberg, 2003.

Scientific résumé

Education

- **2015–2019:**
  PhD candidate – Mathematics, University of Antwerp, Belgium
  PhD thesis: *Automatic exploration techniques for the numerical continuation of large–scale nonlinear systems*
  Supervisor: Wim Vanroose

- **2013–2015:**
  Master of Science in Mathematics: Financial and Applied Mathematics, University of Antwerp, Belgium
  Master's thesis: Compound options and their applications
  Supervisor: Prof. dr Martine Van Wouwe

- **2010–2013:**
  Bachelor of Science in Mathematics, University of Antwerp, Belgium

- **2004–2010:**
  Secondary education Science–Mathematics, Sint–Victor Turnhout, Belgium

Teaching assistant

- Iterative methods for linear systems and eigenvalue problems
  Bachelor of Mathematics: 2015–2016

- Applications of differential equations
  Master of Mathematics: 2015–2017

- Applied linear algebra
  Bachelor of Biochemistry and Biotechnology: 2018–2019
  Bachelor of Chemistry: 2018–2019

Publications

- [**Submitted**] Michiel Wouters, Wim Vanroose: *Automatic exploration techniques of numerical bifurcation diagrams illustrated by the Ginzburg–Landau equation* (submitted 6/3/2019).

Conference talks & poster presentations

- **Mathematics Research Day 2019:** Presentation: Michiel Wouters and Wim Vanroose, *Numerieke continuatie methoden voor de analyse van fysische parameters in dynamische systemen.* May 16th, 2019, Antwerp, Belgium.
- **PMAA 2018:** 10th International Workshop on Parallel Matrix Algorithms and Applications. Presentation: Michiel Wouters and Wim Vanroose, *Deflated Newton–Krylov methods applied to bifurcation problems.* June 27–29, 2018, Zurich, Switzerland.
- **SciCADE 2017:** International Conference on Scientific Computation and Differential Equations. Presentation: Michiel Wouters and Wim Vanroose, *Numerical continuation methods for the Ginzburg–Landau equation of superconductivity.* September 11–15, 2017, Bath, UK.
- **Analysis & Geometry Seminar:** Presentation: Michiel Wouters and Wim Vanroose, *Newton's method and numerical continuation - application to the Ginzburg–Landau model of superconductivity.* February 28th, 2017, Antwerp, Belgium.
- **Faculty of Science Research Day 2017:** Presentation: Michiel Wouters and Wim Vanroose, *Automatic exploration of the solution landscape of mathematical models with numerical continuation.* January 13th, 2017, Antwerp, Belgium.
- **WSC 2016:** 41st Woudschoten Conference of the Dutch–Flemish Scientific Computing Society. Poster: Michiel Wouters and Wim Vanroose, *The Ginzburg–Landau equation of superconductivity: automatic exploration of the solution landscape.* October 5–7, 2016, Zeist, The Netherlands.

Further conferences & workshops

- **WSC Spring 2018:** Spring meeting of the Dutch–Flemish Scientific Computing Society. June 1st, 2018, Eindhoven, The Netherlands.
- **WSC 2017:** 42nd Woudschoten Conference of the Dutch–Flemish Scientific Computing Society. October 4–6, 2017, Zeist, The Netherlands.
- **WSC Spring 2017:** Spring meeting of the Dutch–Flemish Scientific Computing Society. May 19th, 2017, Antwerp, Belgium.
- **WSC Spring 2016:** Spring meeting of the Dutch–Flemish Scientific Computing Society. May 13th, 2016, Utrecht, The Netherlands.

- **WSC 2015:** 40th Woudschoten Conference of the Dutch–Flemish Scientific Computing Society. October 7–9, 2015, Zeist, The Netherlands.

"And now for something completely different."
– The Announcer –
Monty Python's Flying Circus