Numerical Bifurcation Analysis of Periodic Solutions of Partial Differential Equations

Kurt Lust

Department of Computer Science — K.U.Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract

There is a growing interest in the study of periodic phenomena in large-scale nonlinear dynamical systems. Often the high-dimensional system has only low-dimensional dynamics. We propose an approach that exploits this property in order to compute branches of periodic solutions of such systems efficiently. We call our approach the Newton–Picard method.

We first study a family of techniques based on the single shooting method. The linearized single shooting system is projected on the low-dimensional subspace of unstable and weakly stable modes and its orthogonal complement. In both subspaces, different numerical techniques are used. In its simplest form, our method is equivalent to time integration in the high-dimensional subspace followed by Newton-based single shooting in the low-dimensional subspace. The convergence of several variants of the method is thoroughly analyzed and we exploit the analysis to construct variants that are nearly as robust as Newton-based single shooting with a direct linear system solver. As a side-effect of the construction of the projectors we obtain good estimates for the dominant stability-determining Floquet multipliers. Hence the method is particularly suited for bifurcation analysis.

The approach is extended to multiple shooting. The algorithm is shown to be almost as efficient as the single-shooting based methods. However, multiple shooting is a much more robust method. We pay special attention to the accurate computation of the Floquet multipliers for highly unstable limit cycles. We show that the Newton–Picard idea can also be applied to extended systems for the direct computation of bifurcation points using both single- and multiple shooting based techniques.

Preface

Many physical phenomena are described by partial differential equations (PDEs). In the past, many systems in engineering applications were modeled using rather low-dimensional systems of ODEs. However, as ever higher performance is desired, these models can no longer satisfy the needs of the engineer and partial differential equations need to be used to model the system's behavior with the required accuracy. Classical numerical techniques are not sufficient to analyze these models and new techniques need to be developed. Nowadays, the computer power is available to study not only the steady states of large systems, but also the periodic regimes provided efficient algorithms are used. In this thesis, we propose an approach that allows to analyze the bifurcation behavior of periodic solutions of many large systems.

This work would have been impossible without the support of many people. First of all, I wish to thank my supervisor prof. Roose. He initiated this research, created the working environment and brought me into contact with many specialists in the field. I also wish to thank prof. Spence and dr. Champneys, who helped me start with this research. Prof. Roose and prof. Spence also carefully read the text and made several suggestions to improve the text.

Many other people have also supported this work. In particular, I wish to thank Koen Engelborghs, Tatyana Luzyanina, Karl Meerbergen, prof. R. Mattheij, dr. A. Khibnik, prof. Y. Kevrekidis and prof. H. Keller. Koen Engelborghs implemented some of the extensions for delay differential equations (DDEs) and the algorithm for the computation of period doubling points and also provided some test results. Discussions with him triggered some of the improvements made to our early algorithms. Tatyana Luzyanina pointed out that my approach was also suited to compute periodic solutions of DDEs. I learned a lot about iterative methods from the discussions with Karl Meerbergen during the daily walk to the student restaurant. Prof. Mattheij made some time to explain me the basics of his multiple shooting algorithm. This allowed me to finally find the right way to derive an efficient multiple shooting technique. Dr. Khibnik taught me the basics of bifurcation analysis and also learned me the basic principles of the continuation strategy used in his package Locbif. I had several short but instructive discussions with prof. Kevrekidis on the cost of various methods to compute periodic solutions of large systems. He also made several suggestions for further research and gave me some information on other techniques. Prof. Keller gave me the opportunity to visit Caltech in May 1995 and to discuss with him the techniques which I derived.

I enjoyed my stay at the department of computer science. I wish to thank everybody who helped to create the unique atmosphere at the department which allows one to pursue his research successfully. In particular I wish to thank Margot Peeters and Denise

Brams for all the administrative work they did for me.

I am very grateful to my parents and family for the continuing support, especially in the weeks before my departure to the U.S.A. and during the preparation of my defense. They helped me arrange a lot of things while I was busy writing this text.

I also wish to thank prof. S. Vandewalle for very carefully reading this text and pointing out many small errors.

Finally, I also wish to thank the IMA at the University of Minnesota for giving me the opportunity to finish this text and further pursuing my research interests.

Kurt Lust
Heverlee, December 1997

Numerieke bifurcatieanalyse van periodieke oplossingen van partiële differentiaalvergelijkingen

Samenvatting

Inhoudsopgave

1. Inleiding

In dit eindwerk bestuderen we algoritmen voor de berekening en bifurcatieanalyse van periodieke oplossingen van sommige niet-lineaire autonome partiële differentiaalvergelijkingen (PDVen) met parameters. In vele gevallen zijn er slechts weinig zwak-stabiele en onstabiele eigenmodes, bijvoorbeeld in reactie-diffusiesystemen of systemen beschreven door de Navier-Stokes vergelijkingen voor onsamendrukbare vloeistoffen. Onze techniek probeert dit uit te buiten. De doelstelling van dit eindwerk is om een familie van technieken gebaseerd op enkelvoudig- en meervoudig schieten te ontwikkelen om takken van periodieke oplossingen in functie van een parameter te berekenen. Verder willen we ook informatie over de stabiliteit berekenen en detecteren wanneer het aantal onstabiele eigenmodes verandert.

Theoretische en numerieke technieken voor bifurcatieanalyse en voortzettingsmethodes hebben een snelle evolutie gekend sinds het verschijnen van het beroemde artikel van Keller [63] over numerieke oplossingstechnieken voor bifurcatieproblemen en niet-lineaire eigenwaardenproblemen. Er zijn tegenwoordig heel wat pakketten beschikbaar voor de analyse van kleine systemen beschreven door gewone differentiaalvergelijkingen (GDVen) of discrete afbeeldingen. Voorbeelden zijn AUTO [27, 28, 29, 31], Locbif [67, 68] en CONTENT [72]. De eerste versie van AUTO dateert uit 1986. Onlangs werd dit pakket uitgebreid met routines voor de analyse van homoclinische bifurcaties. Locbif kan evenwichtsoplossingen en periodieke oplossingen analyseren van stelsels van autonome GDVen, oplossingen van discrete afbeeldingen en periodieke oplossingen van niet-autonome stelsels van GDVen. CONTENT is een betrekkelijk nieuw pakket dat nog volop verder ontwikkeld wordt. Andere voorbeelden zijn Pitcon [97] en Bifpack [105]. Deze pakketten zijn allemaal geoptimiseerd voor kleine problemen. Hoewel het mogelijk is om stelsels van PDVen om te vormen tot grote stelsels van GDVen door ruimtediscretisatietechnieken zoals (pseudo-)spectrale technieken, eindige differenties of eindige elementen, is het moeilijk dergelijke problemen te analyseren met bovengenoemde pakketten. De reken- en geheugenkost van deze pakketten groeit sterk met toenemende dimensie van het probleem. Bovendien kunnen de pakketten de ijle structuur van de matrices die optreden tijdens de berekeningen niet uitbuiten.

Er is slechts weinig software beschikbaar voor de voortzetting en bifurcatieanalyse van evenwichtsoplossingen van PDVen. Een vrij bekend pakket is PLTMG [8, 9]. Dit pakket kan echter enkel scalaire elliptische PDVen analyseren en keerpunten, transkritische- en stemvorkbifurcatiepunten berekenen. Het pakket is niet in staat de stabiliteit van de berekende oplossingen te analyseren en kan geen andere types van bifurcatiepunten (zoals Hopf bifurcatiepunten) detecteren en berekenen. In [86, 87] wordt een techniek beschreven om takken van evenwichtsoplossingen van PDVen te berekenen en de stabiliteit te analyseren door de berekening van de meest rechtse eigenwaarden. Er is reeds veel werk gebeurd op het gebied van Krylov technieken en domeindecompositiemethodes om evenwichtsoplossingen van PDVen te berekenen en op het gebied van technieken voor het berekenen van de meest rechtse eigenwaarden van een matrix (zie bijvoorbeeld [81] voor een overzicht). Op het gebied van periodieke oplossingen is nog niet zoveel werk gebeurd. In de voorbije jaren werden studies van periodieke oplossingen van PDVen voorgesteld op verschillende conferenties, maat meestal werden de resultaten bekomen door dynamische simulatie of bleven ze beperkt tot heel kleine problemen met slechts weinig vrijheidsgraden.

Verschillende technieken proberen het feit uit te buiten dat dissipatieve systemen meestal een laagdimensionale attractor hebben. De theorie van de inertiële meervoudigheden ("inertial manifolds") beweert dat de attractor vaak ingebed kan worden in een laag-dimensionaal en zacht verlopende meervoudigheid die alle andere trajecten exponentieel snel aantrekt. De lange-termijn dynamica kan dan beschreven worden door een laag-dimensionaal stelsel van GDVen op de inertiële meervoudigheid. Dit idee wordt gebruikt in numerieke technieken door het gebruik van een benaderende inertiële meervoudigheid. In oudere artikels worden niet-lineaire Galerkin methodes gebaseerd op (pseudo-)spectrale discretisatietechnieken gebruikt (bijvoorbeeld in [12]). Tegenwoordig worden ook methodes gebaseerd op andere discretisatietechnieken onderzocht. Vaak is de winst bereikt met dergelijke technieken eerder beperkt. Meestal is het slechts mogelijk het aantal vrijheidsgraden per ruimtedimensie te halveren. Een andere techniek is gebaseerd op een statistische analyse van het tijdsgedrag van het systeem door middel van de Karhunen-Loève ontbinding [7, 10, 23, 46] (ook gekend als de methode van de empirische eigenfuncties of de eigenlijke orthogonale ontbinding). In deze techniek worden de belangrijkste modes uit het systeem berekend uit de covariantiematrix of -operator. De ontbinding is gebaseerd op een statistische analyse van de attractor voor een op voorhand vastgelegde waarde van de parameter. Extrapollatie van de resultaten naar andere parameterwaarden is gevaarlijk en kan leiden tot onbetrouwbare resultaten. In [23] wordt een vrij succesvol en een mislukt experiment beschreven waarbij een gedeelte van het bifurcatiediagram gereconstrueerd wordt via de analyse van het laag-dimensionale model opgesteld voor een welbepaalde parameterwaarde.

Beide bovengenoemde technieken hebben een aantal belangrijke nadelen. De resultaten kunnen heel onbetrouwbaar zijn en er is niet altijd een duidelijke winst in rekentijd. Verder vereisen deze technieken aanpassingen aan de ruimtediscretisatietechnieken. Het is vaak heel moeilijk of zelfs onmogelijk om dergelijke aanpassingen door te voeren. Het ontwikkelen van een goede code voor de simulatie van het tijdsverloop van oplossingen van PDVen is vaak heel duur en tijdrovend. Er is bijgevolg een duidelijke behoefte aan technieken die nauwkeurige en betrouwbare resultaten produceren binnen een aanvaardbare rekentijd en die niet teveel aanpassingen vereisen aan bestaande codes voor simulatie. De techniek die wij voorstellen in dit proefschrift voldoet aan deze vereisten.

In dit proefschrift willen we aantonen dat een goede combinatie van goedkope iteratieve technieken zoals Picard iteratie en de Newton–Raphson methode met een directe methode voor het oplossen van lineaire stelsels leidt tot heel efficiënte algoritmen voor de berekening en bifurcatieanalyse van periodieke oplossingen van PDVen. Dit idee werd voor het eerst gebruikt door Jarausch en Mackens [56, 57, 58] voor de analyse van evenwichtsoplossingen van grote symmetrische problemen. Zij doopten hun techniek de "gecondenseerde Newton–ondersteunende Picard" techniek. De recursieve projectiemethode van Shroff en Keller [107] breidt deze aanpak uit tot niet-symmetrische problemen. In dit proefschrift zullen we eerst een aantal technieken gebaseerd op enkelvoudig schieten voorstellen. Hierbij proberen we in zekere zin tijdsintegratie te combineren met een Newton-gebaseerde methode voor enkelvoudig schieten. De afleiding van onze techniek verschilt in een aantal punten van de afleiding in [107]. Dit leidt tot een familie van technieken. De recursieve projectiemethode van Shroff en Keller is een lid van die familie. Sommige andere leden van de familie presteren beter wanneer de systeemmatrix niet-normaal is (wat het geval is bij de berekening van periodieke oplossingen). Onze techniek is vooral interessant voor de berekening van takken van periodieke oplossingen. Ze levert informatie op over de dominante Floquet vermenigvuldigers zonder een grote extra kost. Een meer algebraïsche kijk op de methode laat toe om onze techniek heel robuust te maken. We tonen in het proefschrift ook aan hoe de methode uitgebreid kan worden tot meervoudig schieten en tot de berekening van bifurcatiepunten. We passen onze methodes ook aan om periodieke oplossingen te berekenen van gewone differentiaalvergelijkingen met één of meerdere discrete vertraginstermen. We geven aan hoe de techniek veralgemeend kan worden tot collocatiemethodes.

2. Technieken gebaseerd op enkelvoudig schieten

2.1. Enkelvoudig schieten op de klassieke manier

Het startpunt voor al onze technieken is het grote stelsel van gewone differentiaalvergelijkingen

$$\frac{dx}{dt} = f(x,\gamma), \quad f : \mathbb{R}^N \times \mathbb{R} \mapsto \mathbb{R}^N \tag{1}$$

bekomen na de ruimtediscretisatie van het stelsel PDVen. N is groot en γ stelt de parameter voor die we wensen te laten variëren.

$$\varphi(x(0), T, \gamma), \quad \varphi : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^N$$

beduidt het resultaat van een tijdsintegratie van (1) over een tijdsinterval van lengte T startende vanuit $x(0)$. Een periodieke oplossing met periode T kan gevonden worden als een oplossing van het niet-lineaire stelsel van vergelijkingen

$$\begin{cases} r(x(0), T; \gamma) = \varphi(x(0), T; \gamma) - x(0) = 0, \\ s(x(0), T; \gamma) = 0. \end{cases} \tag{2}$$

De laatste vergelijking van dit stelsel is de fasevoorwaarde. Deze is nodig omdat ieder punt van de periodieke oplossing het punt op tijdstip 0 kan zijn voor een autonoom systeem. Een goede fasevoorwaarde moet de limietcyclus transversaal snijden. Een voorbeeld van zo'n voorwaarde is

$$f(x(0)_p, \gamma_p)^T (x(0) - x(0)_p) = 0$$

waarbij $x(0)_p$ een punt dicht bij de limietcyclus is.

In de techniek van het enkelvoudig schieten worden $x(0)$ en T rechtstreeks berekend uit het stelsel (2). Wanneer we een tak van periodieke oplossingen wensen te berekenen, laten we γ variëren en voegen we een extra vergelijking toe. In dit proefschrift gebruiken we een vergelijking die een pseudo-booglengte langsheen de berekende tak uitdrukt. Dit leidt tot het stelsel

$$\begin{cases} r(x(0), T, \gamma) = \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) = 0, \\ n(x(0), T, \gamma; \eta) = 0. \end{cases} \tag{3}$$

Hierin is η een artificiële parameter die de pseudo-booglengte uitdrukt. Voorbeelden van dergelijke voorwaarden zijn

$$\begin{aligned} n_1(x(0),T,\gamma;\eta) &= \theta_x \left\|x(0;\eta)-x(0;\eta_0)\right\|_2^2 + \theta_T \left(T(\eta)-T(\eta_0)\right)^2 + \\ &\quad \theta_\gamma \left(\gamma(\eta)-\gamma(\eta_0)\right)^2 - (\eta-\eta_0)^2 = 0, \end{aligned} \tag{4}$$

waarbij $(x(0;\eta_0),T(\eta_0),\gamma(\eta_0))$ een punt is van een reeds berekende periodieke oplossing op de tak, of

$$n_2(x(0),T,\gamma;\eta) = \theta_x \left(x(0)-x_p(0)\right)^T x_s(0) + \theta_T \left(T-T_p\right) T_s + \theta_\gamma \left(\gamma-\gamma_p\right)\gamma_s = 0, \tag{5}$$

waarbij $(x_p(0),T_p,\gamma_p)$ de startwaarde gegenereerd door de predictor in de voortzettingsmethode is en $(x_s(0),T_s(0),\gamma_s(0))$ de genormaliseerde richting gebruikt door de predictor. Eventueel kunnen ook fasevoorwaarden en pseudo-booglengtevoorwaarden gedefinieerd op basis van de ganse limietcyclus en niet op basis van één punt gebruikt worden zoals de voorwaarden gebruikt in het pakket AUTO. Het is echter moeilijk om deze voorwaarden efficient te implementeren in methodes gebaseerd op schieten.

Het niet-lineaire stelsel (3) kan opgelost worden gebruik makende van de Newton–Raphson techniek. Hierbij wordt herhaaldelijk het gelineariseerde stelsel

$$\begin{bmatrix} M-I & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \\ c_n^T & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r(x(0),T,\gamma) \\ s(x(0),T,\gamma) \\ n(x(0),T,\gamma;\eta) \end{bmatrix} \tag{6}$$

opgelost. In de oplossing $(x(0)^*,T^*,\gamma^*)$ van (3) is M de "monodromy matrix" of systeemmatrix. Deze matrix heeft voor een autonoom stelsel steeds een eigenwaarde 1. De bijhorende eigenvector is

$$b_T^* = f(x(0)^*,\gamma^*),$$

de raaklijn aan de limietcyclus in het punt $x(0)^*$. De andere eigenwaarden geven informatie over de stabiliteit van de limietcyclus. De eigenwaarden van de systeemmatrix worden de Floquet vermenigvuldigers of karakteristieke vermenigvuldigers genoemd. Wanneer alle Floquet vermenigvuldigers behalve de triviale (de eigenwaarde 1) binnen de eenheidscirkel rond de oorsprong in het complexe vlak liggen, is de limietcyclus stabiel. Merk op dat de systeemmatrix een volle $N \times N$ matrix is. De berekening van M vereist het oplossen van N beginwaardeproblemen van dimensie N. Verder vereist het opslaan van M veel geheugen en is het duur om stelsels met de volle matrix M op te lossen. Bijgevolg wensen we het berekenen van M te vermijden.

2.2. Afleiding van de Newton–Picard methode

Zoals reeds vroeger vermeld, heeft M slechts weinig grote eigenwaarden(d.i. eigenwaarden dicht bij of buiten de eenheidscirkel in het complexe vlak) in vele praktische problemen. Bovendien zijn deze eigenwaarden nagenoeg onafhankelijk van de gebruikte discretisatietechniek en het aantal vrijheidsgraden in de ruimtediscretisatie. Dit is logisch vermits deze eigenmodes het gedrag van het systeem op middellange en lange termijn bepalen en diegene zijn die fysisch waargenomen worden, en bijgevolg goed benaderd moeten worden door iedere behoorlijke discretisatietechniek. Dit wordt geïllustreerd in Tabel 1. Deze

15 punten	31 punten	63 punten	127 punten
$1{,}1684 \pm 0{,}4576i$	$1{,}2219 \pm 0{,}4387i$	$1{,}2367 \pm 0{,}4360i$	$1{,}2367 \pm 0{,}4361i$
$1{,}1958 \pm 1{,}1631i$	$1{,}1780 \pm 0{,}1716i$	$1{,}1710 \pm 0{,}1692i$	$1{,}1710 \pm 0{,}1692i$
1,0000	1,0000	1,0000	1,0000
$0{,}6840 \pm 0{,}1320i$	$0{,}7065 \pm 0{,}1114i$	$0{,}7131 \pm 0{,}1003i$	$0{,}7131 \pm 0{,}1003i$
$0{,}4257 \pm 0{,}1341i$	$0{,}4026 \pm 0{,}1389i$	$0{,}3951 \pm 0{,}1393i$	$0{,}3951 \pm 0{,}1393i$
0,3425	0,3720	0,3823	0,3823
$0{,}1986 \pm 0{,}1641i$	$0{,}1673 \pm 0{,}1622i$	$0{,}1593 \pm 0{,}1611i$	$0{,}1593 \pm 0{,}1611i$
$0{,}0866 \pm 0{,}1280i$	$0{,}0537 \pm 0{,}1112i$	$0{,}0464 \pm 0{,}1064i$	$0{,}0464 \pm 0{,}1064i$

Tabel 1: Dominante Floquet vermenigvuldigers voor het Brusselatormodel; één van de periodieke oplossingen voor $L = 1{,}9$.

tabel geeft de meest dominante eigenwaarden van M voor een periodieke oplossing van het ééndimensionale Brusselatormodel

$$\left\{\begin{array}{rcl} \dfrac{\partial X}{\partial t} &=& \dfrac{D_X}{L^2}\dfrac{\partial^2 X}{\partial z^2} + X^2Y - (B+1)X + A, \\[2ex] \dfrac{\partial Y}{\partial t} &=& \dfrac{D_Y}{L^2}\dfrac{\partial^2 Y}{\partial z^2} - X^2Y + BX, \end{array}\right. \tag{7}$$

met Dirichlet randvoorwaarden

$$\left\{\begin{array}{l} X(t, z=0) = X(t, z=1) = A, \\ Y(t, z=0) = Y(t, z=1) = \dfrac{B}{A}, \end{array}\right. \tag{8}$$

waarbij $L = 1.9$, $A = 2$, $B = 5.45$, $D_X = 0.008$ en $D_Y = 0.004$. Het stelsel werd gediscretiseerd met tweede-orde centrale differenties. We gebruikten 15, 31, 63 en 127 discretisatiepunten. Merk op dat er nog nauwelijks veranderingen optreden wanneer we overgaan van 63 op 127 discretisatiepunten. In de problemen beschreven in [7] en [120] treedt een gelijkaardig spectrum op. Bijgevolg is het redelijk te vertrekken van de volgende veronderstelling

Veronderstelling 1 *Veronderstel dat $y^* = (x(0)^*, T^*, \gamma^*)$ een geïsoleerde oplossing is van (3). Beschouw een kleine omgeving $\mathcal{B}$ van y^*. Stel $M(y) = \frac{\partial \varphi}{\partial x(0)}(y)$ voor $y \in \mathcal{B}$. We noemen de eigenwaarden van $M(y)$ μ_i, $i = 1, \ldots, N$. Veronderstel dat voor alle $y \in \mathcal{B}$ er precies p eigenwaarden buiten de schijf*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{9}$$

liggen en dat geen enkele eigenwaarde modulus ρ heeft. Dit houdt in dat voor alle $y \in \mathcal{B}$

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

Deze veronderstelling is strenger dan de veronderstelling in [107]. In y^* is M^* de systeemmatrix. Onze methode is efficient wanneer $p \ll N$. Stel dat de kolomvectoren van $V_p \in \mathbb{R}^{N\times p}$ een basis vormen voor de deelruimte $\mathcal{U}$ van $\mathbb{R}^N$ van de veralgemeende eigenvectoren van $M(x(0),T,\gamma)$ overeenkomende met de eigenwaarden μ_i, $i=1,\ldots,N$ en stel dat de kolomvectoren van $V_q \in \mathbb{R}^{N\times(N-p)} = \mathbb{R}^{N\times q}$ een basis vormen voor $\mathcal{U}^\perp$, het orthogonaal complement van $\mathcal{U}$ in $\mathbb{R}^N$. Meestal is $\mathcal{U}^\perp$ geen invariante deelruimte van $M(x(0),T)$. V_p en V_q kunnen berekend worden via de reële Schur ontbinding van $M(x(0),T)$. Tijdens de berekening van de periodieke oplossing kunnen sommige eigenwaarden in en uit de schijf C_ρ bewegen en onze code kan deze situatie opvangen. Naarmate de iteratie convergeert kan de veronderstelling 1 in de praktijk waargenomen worden. Bijgevolg is deze veronderstelling niet onrealistisch.

De orthogonale projectoren

$$\begin{aligned} P &:= V_p V_p^T \text{ en} \\ Q &:= V_q V_q^T = I_N - V_p V_p^T \end{aligned} \tag{10}$$

beelden $\mathbb{R}^N$ af op $\mathcal{U}$ en $\mathcal{U}^\perp$. Elke vector $x(0)\in\mathbb{R}^N$ heeft een unieke ontbinding in termen van de basissen V_p en V_q. We kunnen stellen

$$x(0) = V_p\bar{p} + V_q\bar{q} = p + q, \quad p = V_p\bar{p} = Px(0), \quad q = V_q\bar{q} = Qx(0) \tag{11}$$

waarbij $\bar{p}\in\mathbb{R}^p$ en $\bar{q}\in\mathbb{R}^{N-p}$. Wanneer we deze betrekking invullen in (6) en de eerste N vergelijkingen langs links vermenigvuldigen met $[V_p \;\; V_q]^T$ verkrijgen we het stelsel

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & V_q^T b_T & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}. \tag{12}$$

Hierbij gebruikten wij het feit dat $\mathcal{U}$ een invariante deelruimte is van de matrix M en bijgevolg $V_q^T M V_p = 0$, tenminste wanneer we aannemen dat de basis nauwkeurig berekend werd. Op de limietcyclus is $b_T^* = b_T(x(0)^*, T^*, \gamma^*)$ een eigenvector van $M(x(0)^*, T^*, \gamma^*)$ voor de triviale Floquet vermenigvuldiger 1. Deze vector ligt in de deelruimte $\mathcal{U}$ en bijgevolg is $V_q^T b_T^* = 0$. Dit is niet meer het geval wannneer de basis niet exact berekend wordt of wanneer het startpunt van de iteratiestap niet op de limietcyclus ligt. Gewoonlijk is de term $V_q^T b_T$ echter heel klein en kan deze term verwaarloosd worden. We houden echter rekening met deze term in de afleiding van de methode.

Voor het oplossen van het stelsel (12) doen we beroep op een techniek gelijkaardig aan de techniek gebruikt in [83]. Uit de eerste $N-p$ vergelijkingen van (12) kan men gemakkelijk afleiden dat

$$\Delta\bar{q} = -\left(V_q^T M V_q - I_q\right)^{-1}\left(V_q^T r + \Delta T\, V_q^T b_T + \Delta\gamma\, V_q^T b_\gamma\right).$$

Bijgevolg kan $\Delta\bar{q}$ ontbonden worden in 3 componenten volgens

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta T\,\Delta\bar{q}_T + \Delta\gamma\,\Delta\bar{q}_\gamma \tag{13}$$

waarbij de bijdragen $\Delta\bar{q}_r$, $\Delta\bar{q}_T$ en $\Delta\bar{q}_\gamma$ de oplossing zijn van het stelsel

$$\left[V_q^T M V_q - I_q\right]\left[\begin{array}{ccc} \Delta\bar{q}_r & \Delta\bar{q}_T & \Delta\bar{q}_\gamma \end{array}\right] = -\left[\begin{array}{ccc} V_q^T r & V_q^T b_T & V_q^T b_\gamma \end{array}\right]. \tag{14}$$

De andere onbekenden $\Delta\bar{p}$, ΔT en $\Delta\gamma$ worden berekend uit het stelsel

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T\left(b_T + M V_q^T \Delta\bar{q}_T\right) & V_p^T(b_\gamma + M V_q^T \Delta\bar{q}_\gamma) \\ c_s^T V_p & d_{s,T} + c_s^T V_q \Delta\bar{q}_T & d_{s,\gamma} + c_s^T V_q \Delta\bar{q}_\gamma \\ c_n^T V_p & d_{n,T} + c_n^T V_q \Delta\bar{q}_T & d_{n,\gamma} + c_n^T V_q \Delta\bar{q}_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T(r + M V_q \Delta\bar{q}_r) \\ s + c_s^T V_q \Delta\bar{q}_r \\ n + c_n^T V_q \Delta\bar{q}_r \end{bmatrix} \tag{15}$$

waarna $\Delta\bar{q}$ kan worden gevormd via de betrekking (13). Men kan aantonen dat de matrix in het stelsel (15) regulier is als en alleen als de matrix in het stelsel (12) en bijgevolg (6) regulier is wanneer tenminste wordt verondersteld dat de basis nauwkeurig berekend werd en het stelsel (14) nauwkeurig opgelost werd. Het stelsel (15) is slechts een $(p+2)\times(p+2)$ stelsel en kan opgelost worden met behulp van Gauss eliminatie met pivotering. Het stelsel (14) is echter een stelsel van dimensie $N-p$. Het is duidelijk dat we de constructie van de basis V_q en de matrix $V_q^T M V_q$ willen vermijden. We lossen dit stelsel op met het Picard schema

$$\begin{aligned} &\Delta\bar{q}_*^{[0]} = 0 \text{ (of een andere startwaarde)}, \\ &\Delta\bar{q}_*^{[i]} = V_q^T M V_q \Delta\bar{q}_*^{[i-1]} + V_q^T r_*, \ i = 1, \cdots, \text{ tot convergentie.} \end{aligned} \tag{16}$$

Hierin stelt r_* de residu r of een van de vectoren b_T of b_γ voor en is $\Delta\bar{q}_*$ de bijhorende onbekende. Vermits voor de opbouw van het stelsel (15) en de vector $\Delta x(0)$ enkel de termen $\Delta q_* = V_q \Delta\bar{q}_*$ nodig zijn, kunnen we echter meteen de N-dimensionale vectoren Δq_* berekenen met behulp van het Picard schema

$$\begin{aligned} &\Delta q_*^{[0]} = 0 \text{ (of een andere startwaarde)}, \\ &\Delta q_*^{[i]} = \left(I - V_p V_p^T\right)\left(M \Delta q_*^{[i-1]} + r_*\right), \ i = 1, \cdots, \text{ tot convergentie} \end{aligned} \tag{17}$$

Dit schema wordt bekomen door iedere vergelijking in (16) langs links te vermenigvuldigen met V_q. Merk op dat de basis V_q niet meer nodig is in (17).

De basis V_p kan berekend worden door middel van orthogonale deelruimteïteratie met projectie en deflatie. Dit algoritme vereist enkel matrix–vectorprodukten met de matrix M. Bovendien kunnen we een aantal matrix–vectorprodukten recupereren voor de opbouw van het laagdimensionale stelsel (15).

De matrix–vectorprodukten Mv kunnen berekend worden met behulp van numerieke differentiatie, bijvoorbeeld met de formule

$$Mv = \frac{\varphi(x(0) + \varepsilon v, T, \gamma) - \varphi(x(0), T, \gamma)}{\varepsilon}, \tag{18}$$

of door het integreren van de variationele vergelijking

$$\frac{dv(t)}{dt} = \left.\frac{df(x,\gamma)}{dx}\right|_{(\varphi(x(0),t,\gamma),\gamma)} v(t), \quad v(0) = v, \tag{19}$$

over het interval $[0, T]$. In dit geval is $v(T) = Mv$. In het eerste geval is er één bijkomende tijdsintegratie nodig van het niet-lineaire probleem (1), in het tweede geval moet er een

N-dimensionaal lineair beginwaardeprobleem geïntegreerd worden wanneer we tenminste veronderstellen dat $\varphi(x(0), t, \gamma)$ bijgehouden werd voor alle waarden van t. In het proefschrift bespreken we hoe het verband tussen de variationele vergelijking (19) en het niet-lineaire probleem (1) uitgebuit kan worden tijdens de tijdsintegratie.

De Newton–Picard methode kan ook gebruikt worden voor de berekening van een periodieke oplossing voor een vooraf bepaalde waarden van γ. In dit geval lossen we (2) op. De termen komende van de pseudo-booglengtevergelijking en de afgeleiden naar de parameter γ vallen dan weg in de bovenstaande afleiding.

Laten we eerst een aantal eenvoudige varianten van de methode bekijken.

2.3. Een aantal eenvoudige varianten

In deel 2.2 stelden we de meest krachtige varianten van onze techniek voor. We kunnen een aantal eenvoudige varianten afleiden door de term $V_q^T b_T$ in (12) te verwaarlozen, wat overeen komt met $\Delta\bar{q}_T = 0$ in (13) en (15) en door in (17) een vooraf bepaald aantal Picard stappen te nemen, d.w.z.

$$\begin{aligned} &\Delta q_*^{[0]} = 0, \\ &\Delta q_*^{[i]} = \left(I - V_p V_p^T\right)\left(M \Delta q_*^{[i-1]} + r_*\right), \quad i = 1, \cdots, l, \\ &\Delta q_* = \Delta q^{[l]}. \end{aligned} \tag{20}$$

Merk hierbij op dat

$$\Delta q_* = \Delta q_*^{[l]} = V_q \left(\sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i \right) V_q^T r_*.$$

- Beschouwen we eerst de berekening voor een vooraf bepaalde parameterwaarde γ. In dit geval is

  $$\Delta q = \Delta q^{[l]} = V_q \left(\sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i \right) V_q^T r \tag{21}$$

  en $\Delta\bar{p}$ en ΔT kunnen berekend worden als de oplossing van het stelsel

  $$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T (r + M\Delta q) \\ s + c_s^T \Delta q \end{bmatrix}. \tag{22}$$

  Het rechterlid van dit stelsel op hogere orde termen na equivalent is met

  $$\begin{bmatrix} V_p^T r(x(0) + \Delta q, T, \gamma) \\ s(x(0) + \Delta q, T, \gamma) \end{bmatrix},$$

  m.a.w. het rechterlid van het laagdimensionaal stelsel werd berekend in het punt $(x(0) + \Delta q, T, \gamma)$ bekomen na het benaderend oplossen van het Q-stelsel met (21). Men kan aantonen dat de formule (21) op hogere orde termen na overeenkomt met l tijdsintegraties rond de limietcyclus geprojecteerd in de deelruimte $\mathcal{U}^\perp$. Bovendien heeft het P-stelsel (22) precies dezelfde vorm als het originele gelineariseerde stelsel voor enkelvoudig schieten (d.i. (6) zonder de rand voor de pseudo-booglengtevoorwaarde en de afgeleiden naar de parameter γ). Daarom kunnen we zeggen dat de methode eigenlijk tijdsintegratie in de hoogdimensionale deelruimte $\mathcal{U}^\perp$ combineert met een Newton-gebaseerde techniek voor enkelvoudig schieten in de laagdimensionale deelruimte $\mathcal{U}$. Het schema komt ook overeen met de berekening van de exacte oplossing van het stelsel

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1}\left(V_q^T M V_q\right)^i\right)^{-1} & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \end{bmatrix}, \quad (23)$$

d.i. we benaderen de matrix

$$\begin{bmatrix} V_q^T M V_q - I_q & V_q^T M V_p & V_q^T b_T \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix}$$

door de bovendriehoeksmatrix in (23). Deze interpretatie vormt de basis van een theoretische analyse van de asymptotische convergentie in het proefschrift.
De combinatie van (21) gevolgd door (22) gelijkt op een Gauss-Seidel techniek. Daarom en vermits we l Picard iteratiestappen gebruiken duiden we deze methode aan met de code $NPGS(l)$, wat staat voor Newton–Picard Gauss-Seidel met l Picard stappen.

- We kunnen ook de termen $V_p^T M \Delta q$ en $c_s^T \Delta q$ in het rechterlid van (22) weglaten, d.w.z. we berekenen $\Delta\bar{p}$ en ΔT uit het stelsel

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \end{bmatrix}. \quad (24)$$

De berekening van $\Delta\bar{q}$ enerzijds en $\Delta\bar{p}$ en ΔT anderzijds gebeurt onafhankelijk van elkaar. Dit schema gelijkt op een Jacobi-aanpak. De combinatie van (21) en (24) wordt daarom aangeduid met de code $NPJ(l)$, voor Newton–Picard Jacobi met l Picard stappen. Dit schema met $l = 1$ komt overeen met de recursieve projectiemethode voorgesteld in [107]. In vergelijking met de $NPGS(l)$ techniek sparen we in deze techniek één matrix–vectorprodukt uit in iedere iteratiestap. Merk echter op dat de term $V_p^T M \Delta q$ heel groot kan zijn wanneer M een niet-normale matrix is. Het verwaarlozen van deze term kan het convergentiegedrag sterk beïnvloeden.

- Een eerste voortzettingssvariante wordt gevonden door Δq_r en Δq_γ te berekenen volgens

$$\begin{bmatrix} \Delta q_r & \Delta q_\gamma \end{bmatrix} = \begin{bmatrix} \Delta q_r^{[l]} & \Delta q_\gamma^{[l]} \end{bmatrix} = V_q \left(\sum_{i=0}^{l-1}\left(V_q^T M V_q\right)^i\right) V_q^T \begin{bmatrix} r & b_\gamma \end{bmatrix}$$

en $\Delta\bar{p}$, ΔT en $\Delta\gamma$ uit

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T (b_\gamma + M\Delta q_\gamma) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} + c_s^T \Delta q_\gamma \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} + c_n^T \Delta q_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T (r + M\Delta q_r) \\ s + c_s^T \Delta q_r \\ n + c_n^T \Delta q_r \end{bmatrix}. \quad (25)$$

We duiden deze variante aan met de code $CNP(l)$, voor Continueringsvariante van de Newton–Picard methode met l Picard iteratiestappen. Merk op dat we de afkortingen eigenlijk baseerden op het engels, vandaar continuering i.p.v. voortzetting. We verkozen echter de namen van de schema's niet te wijzigen voor deze nederlandse samenvatting. Bemerk ook dat we in de $CNP(l)$ methode twee sequenties van l Picard stappen moeten uitvoeren. Bijgevolg is een $CNP(l)$ iteratiestap heel wat duurder dan een stap met de $NPGS(l)$ of $NPJ(l)$ methode.

- We kunnen opnieuw de termen in Δq_* verwaarlozen in (25) en $\Delta\bar{p}$, ΔT en $\Delta\gamma$ berekenen uit

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \\ n \end{bmatrix}. \tag{26}$$

In dit geval hangt het P-stelsel niet meer af van de termen Δq_r en Δq_γ. Het stelsel (26) kan dus eerst opgelost worden. Hierna is $\Delta\gamma$ gekend. Bijgevolg moet Δq niet meer worden berekend uit de twee componenten Δq_r en Δq_γ, maar kunnen de Picard iteratiestappen voor beide termen worden samengevoegd en kan Δq worden berekend als

$$\Delta q = \Delta q^{[l]} = V_q \left(\sum_{i=0}^{l-1} \left(V_q^T M V_q \right)^i \right) V_q^T \left(r + \Delta\gamma\, b_\gamma \right).$$

Hierbij is slechts 1 sequentie van l Picard stappen nodig. Deze techniek is dus heel wat goedkoper per iteratiestap dan de $CNP(l)$ techniek. Voor $l = 1$ komt deze methode overeen met de continueringsvariante van de recursieve projectiemethode in [107]. Daarom duiden we deze methode aan met de code $CRP(l)$, voor Continueringsvariante van de Recursieve Projectiemethode met l Picard iteratiestappen.

In het proefschrift tonen we aan dat de $NPGS(l)$ en $NPJ(l)$ varianten asymptotisch lineair convergeren wanneer we veronderstellen dat de basis V_p exact berekend wordt. De asymptotische lineaire convergentiefactor is

$$\left|\mu_{p+1}\right|^l < \rho^l,$$

d.w.z. dat het convergentiegedrag bepaald wordt door het Picard gedeelte van het schema. Het bewijs kan echter niet veralgemeend worden tot de continueringsvarianten $CNP(l)$ en $CRP(l)$. Merk op dat de waarde van μ_{p+1} nagenoeg onafhankelijk is van de discretisatie. Bijgevolg is het aantal tijdsintegraties (of matrix–vectorprodukten) dat nodig is om een vooraf bepaalde verbetering van de residu te verkrijgen, nagenoeg onafhankelijk van de discretisatie en het aantal vrijheidsgraden N. Dit maakt de methode bijzonder aantrekkelijk wanneer N groot is en p redelijk klein. De toename van de rekenkost van één stap wanneer N wordt verhoogd hang volledig af van de toename van de rekenkost voor een tijdsintegratie.

2.4. Implementatieaspecten

We hebben reeds kort aangehaald hoe matrix–vectorprodukten met de systeemmatrix berekend kunnen worden zonder eerst expliciet de systeemmatrix op te bouwen (zie (18) en (19)). Eenmaal de basis V_p gekend is, kunnen we ook eenvoudig het Picard iteratieschema (17) implementeren. Net als in [56, 57, 58] en [107] gebruiken we orthogonale deelruimteïteratie om de basis V_p te berekenen. Om wille van de hoge kost van de matrix–vectorprodukten met de systeemmatrix verkozen we echter de meest gesofisticeerde variante van het algoritme te gebruiken, orthogonale deelruimteïteratie met projectie en deflatie. We gebruikten een variante van algoritme 5.4 uit [101] (met $iter = 1$), licht herwerkt om ieder overbodig matrix–vectorprodukt te vermijden. In het proefschrif ontwikkelen we een strategie om op een robuuste manier de grootte p van de basis te bepalen. Daartoe gebruiken we p_e extra vectoren in de basis. De extra kost door de grotere basis blijft beperkt omdat door gebruik van projectie de meest dominante eigenvectoren sneller convergeren. Als initiële basis gebruiken we de laatste basis voor de vorige periodieke oplossing op de tak. We brengen wel een kleine willekeurige verstoring aan zodat nieuwe vectoren gemakkelijker in de basis kunnen komen. Voor de eerste periodieke oplossing op de tak kunnen we een startbasis afleiden uit de eigenvectoren voor de meest rechtse eigenwaarden van $\partial f(x,\gamma)/\partial x$ in het Hopf bifurcatiepunt of een aantal willekeurig gekozen vectoren gebruiken. Om de nauwkeurigheid van een dominante subset van de basis te testen, gebruiken we een convergentiecriterium voorgesteld in [110]. Dit criterium vereist geen extra matrix–vectorprodukten. In het proefschrift tonen we dat dit criterium

$$\max_{\substack{\|\bar{p}_k\|_2=1\\ \bar{p}_k\in\mathbb{R}^k}} \left\| \left(I - V_k V_k^T\right) M V_k \bar{p}_k \right\|_2$$

contoleert.

Tot slot van deze korte discussie over implementatieaspecten stellen we een uitgewerkt algoritme voor het NPGS(l) schema voor, met inbegrip van het deelruimteïteratiealgoritme.

Algoritme 1 *NPGS(l) en orthogonale deelruimteïteratie met projectie en deflatie voor de berekening van periodieke oplossingen.*

```
Invoer:
    Startwaarde $x^{(0)}(0)$, $T^{(0)}$.
    Startbasis $V_e = \begin{bmatrix} v_1 & \cdots & v_{p+p_e} \end{bmatrix}$
    Functie voor de berekening van $M(x(0),T)\,v$.
Uitvoer:
    GEFAALD of $x^*(0)$, $T^*$, eindbasis $V_e$.
Begin
    $\nu \leftarrow 0$;
    $p_{eff} \leftarrow 0$;
    $W \leftarrow V_e$;
    Herhaal
        Bereken $\varphi \leftarrow \varphi(x(0),T)$; $r \leftarrow \varphi - x(0)$;
        /* Berekening van de basis. */
        Voor $i = 1$ tot $\nu_{sub}$ doe
```

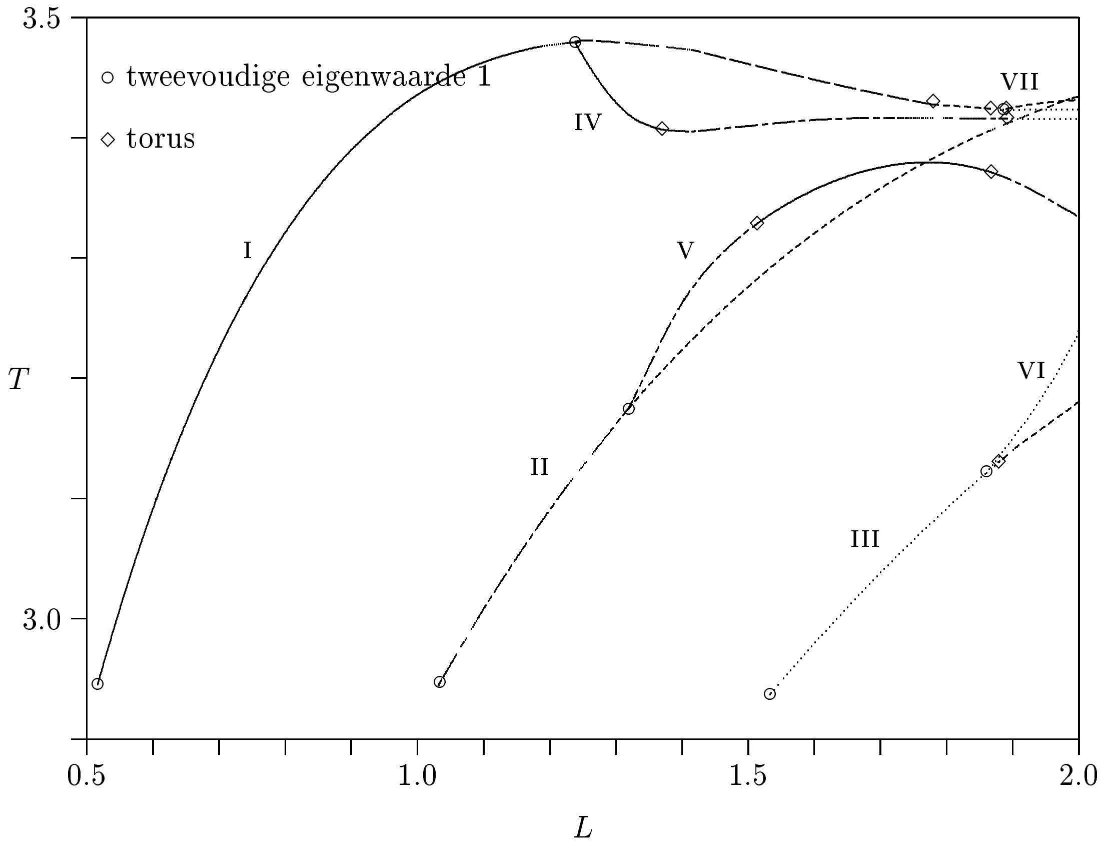
```
            $V_e[p_{eff}+1 : p+p_e] \leftarrow W[p_{eff}+1 : p+p_e]$
            Orthonormaliseer de kolomvectoren $p_{eff}+1$ t.e.m. $p+p_e$ van $V_e$ ten opzichte van alle vorige vectoren zodat $V_e$ een orthogonale matrix wordt.
            Bereken $W[p_{eff}+1 : p+p_e] = MV_e[p_{eff}+1 : p+p_e]$.
            Bereken $U = V_e^T M V_e = V_e^T W$
            Bereken de Schur ontbinding $UY = YS$ van $U$, orden de Schur vectoren volgens dalende modulus van de bijhorende eigenwaarde.
            $V_e \leftarrow V_e Y$, $W \leftarrow WY$.
            Bepaal het aantal nauwkeurige vectoren $p_{eff}$ in $V_e$.
        Einde
        /* Picarditeraties. */
        $\Delta q \leftarrow \left(I - V_e V_e^T\right) r$
        Voor $i = 2$ tot $l$ doe
            $\Delta q \leftarrow \left(I - V_e V_e^T\right) (M\,\Delta q + r)$
        Einde
        /* Newton correctie. */
        Bereken de oplossing van
            $$\begin{bmatrix} S - I_{p+p_e} & V_e^T f(\varphi) \\ c_s(x(0),T)^T & d_{s,T}(x(0),T) \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_e^T(r + M(x(0),T)\,\Delta q) \\ s(x(0),T) + c_s(x(0),T)^T \Delta q \end{bmatrix}.$$
        /* Pas het punt $(x(0),T)$ aan. */
        $x(0) \leftarrow x(0) + \Delta q + V_e \Delta\bar{p}$
        $T \leftarrow T + \Delta T$
        $\nu \leftarrow \nu + 1$
        Beslis of $p_{eff}$ opnieuw op 0 gezet moet worden.
    tot $(\nu \geq \nu_{\max})$ of convergentie
    Indien ( geen convergentie ) dan verlaat de routine en geef de waarde GEFAALD terug.
    anders $x^*(0) \leftarrow x(0)$; $T^* \leftarrow T$.
Einde
```

De andere schema's kunnen op een gelijkaardige manier geïmplementeerd worden.

2.5. Enkele testvoorbeelden

Om de performantie van de methodes geïntroduceerd in deel 2.3 te bestuderen, beschouwen we de volgende twee testvoorbeelden:

Brusselator model Er bestaan vele varianten van het Brusselator model. Wij bestuderen een variante gedefinieerd op het eenheidsinterval en met twee onbekende concentraties. Dit model wordt gegeven door de vergelijkingen (7)–(8). De parameter L is onze voortzettingsparameter. De andere parameters hebben de waarden $A = 2$,



Figuur 1: Bifurcatiediagram van de periodieke oplossingen van het gediscretiseerde Brusselatormodel ($h = \frac{1}{32}$), periode T in functie van de lengte L van de reactor. De romeinse cijfers duiden het nummer van de tak aan in de tabellen in de tekst. Tweevoudige Floquet vermenigvuldigers 1 en torus bifurcatiepunten worden respectievelijk met $\circ$ en $\diamond$ aangeduid. Op de berekende takken zijn geen periodeverdubbelingspunten.

$B = 5{,}45$, $D_X = 0{,}008$ en $D_Y = 0{,}004$. Op de tak van evenwichtsoplossingen $X = A = 2$, $Y = B/A = 2{,}725$ treden Hopf bifurcaties op voor de parameterwaarden

$$L_k^H = k\pi\sqrt{\frac{D_X + D_Y}{B - A^2 - 1}} = k\,0{,}5130.$$

We gebruiken een $O(h^2)$ centrale eindige differentie discretisatie voor de ruimtecoordinaat met roosterafstand $h = 1/32$, resulterend in een stelsel van 62 GDVen. Voor de tijdsintegratie gebruikten we een routine uit het pakket LSODE [95]. Het berekende bifurcatiediagram voor de periodieke oplossingen is gegeven in Figuur 1. De positie van de bifurcatiepunten werd ook op een fijner rooster gecontroleerd.

Olmstead model Het tweede model is een eenvoudig model voor een vloeistofstroming met geheugen [90, 47]. Dit model is ook een ééndimensionaal model. Het oorspronkelijke model is een integraal-differentiaalvergelijking. We gebruiken de variante met de Jeffreys kern in het integraalgedeelte. Deze kan herschreven worden in een stelsel van twee partiële differentiaalvergelijkingen

$$\begin{aligned} \frac{\partial u}{\partial t} &= \delta\frac{\partial^2 u}{\partial x^2} + (1-\delta)\frac{\partial^2 v}{\partial x^2} + Ru - u^3, \\ \lambda\frac{\partial v}{\partial t} &= u - v, \end{aligned} \tag{27}$$

met Dirichlet randvoorwaarden

$$\begin{aligned} u(t, x=0) = u(t, x=\pi) = 0, \\ v(t, x=0) = u(t, x=\pi) = 0. \end{aligned} \tag{28}$$

We voeren een voortzetting uit in het Rayleigh getal R. De waarden voor λ en δ zijn respectievelijk $\lambda = 2$ en $\delta = 0{,}1$. Het bifurcatiediagram voor evenwichtsoplossingen heeft een tak voor $u = v = 0$. Op deze tak hebben we Hopf bifurcatiepunten voor

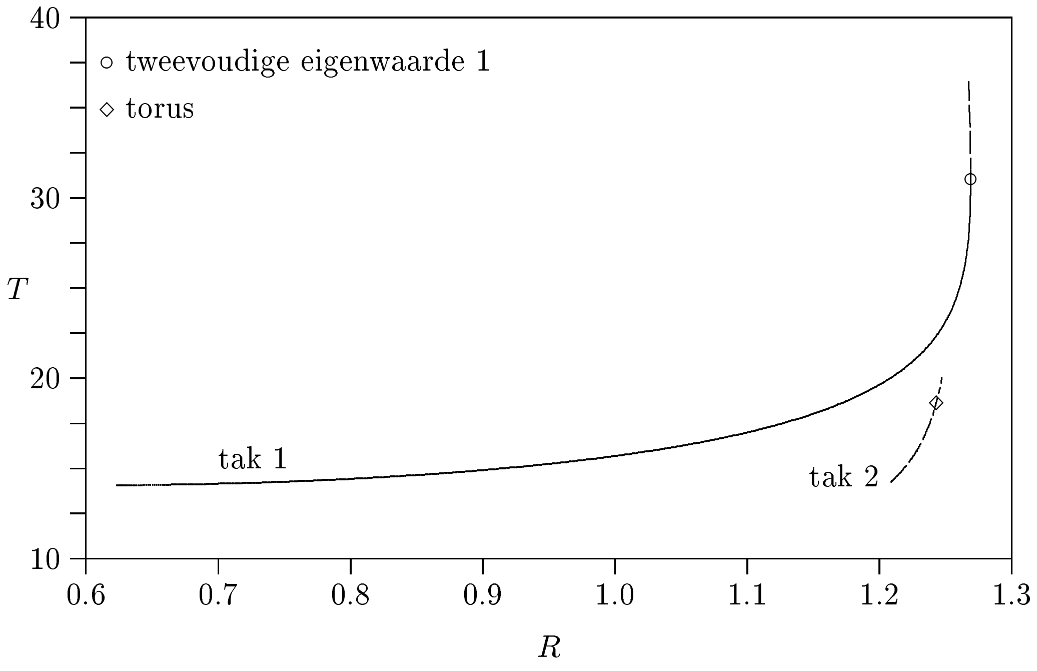
$$R = \frac{1}{\lambda} + k^2\delta = 0{,}5 + k^2 0{,}1,\ k = 1, 2, \ldots$$

en stemvorkbifurcatiepunten voor

$$R = k^2,\ k = 1, 2, \ldots.$$

De takken van evenwichtsoplossingen die voortkomen uit de stemvorkbifurcatiepunten hebben ook een aantal Hopf bifurcatiepunten. We berekenden delen van de takken van periodieke oplossingen die ontspringen vanuit de triviale tak voor $R = 0{,}6$ en vanuit de tweede Hopf bifurcatie op de tak die ontspringt in het eerste stemvorkbifurcatiepunt. Deze takken worden afgebeeld in Figuur 2. We gebruikten opnieuw een $O(h^2)$ centrale differentieschema voor de ruimtediscretisatie, ditmaal met $h = 1/41$, resulterend in een stelsel van 80 GDVen. Voor de tijdsdiscretisatie gebruikten we de trapeziumregel.

Testresultaten We berekenden de takken I t.e.m. IV van Figuur 1 en een gedeelte van de twee takken in Figuur 2 met de $NPJ(l)$, $NPGS(l)$, $CRP(l)$ en $CNP(l)$ schema's. Ter vergelijking berekende we ook een oplossing met behulp van de Newton methode. In het laatste geval probeerden we ook om de jacobiaanmatrix vast te houden gedurende een aantal iteraties. In alle gevallen probeerden we een aantal waarden voor de verschillende parameters in de algoritmen uit (waaronder l); telkens vermelden we enkel het beste resultaat. Voor de berekening van de kostprijs brachten we ieder matrix–vectorprodukt in rekening als één oplossing van een beginwaardeprobleem. De resultaten voor de kostprijs in termen van het aantal op te lossen beginwaardeproblemen wordt gegeven in Tabel 2. Merk op dat het $NPGS(l)$ schema veel beter presteert dan het $NPJ(l)$ schema. Dit laatste schema is ongeveer 50% duurder. Het verschil treedt echter alleen op voor het Brusselator testmodel en vooral in het begin van de iteraties. Bij het Gauss-Seidel schema kunnen we een beduidend grotere staplengte nemen. Het attractiegebied van de oplossing is dus duidelijk groter voor het Gauss-Seidel schema. Het $CNP(l)$ schema valt duurder uit dan het $CRP(l)$ schema. Het is echter robuuster, wat niet af te lezen valt in Tabel 2 vermits deze tabel enkel het beste bekomen resultaat vermeld. Daarom is in de praktijk het $CNP(l)$ schema toch te verkiezen boven het $CRP(l)$ schema. Het is echter duidelijk dat het beter is de voortzettingsvarianten met pseudo-booglengtevoortzetting



Figuur 2: Twee (onvolledige) takken van periodieke oplossingen voor het Olmstead model, de periode T in functie van het Rayleigh getal R.

slechts te gebruiken wanneer een voorzettingsmethode in functie van de parameter γ faalt, bijvoorbeeld in de buurt van een keerpunt.

In Tabel 3 vergelijken we het aantal stappen genomen in de voortzettingsmethode. Hier blijkt duidelijk dat het $NPGS(l)$ schema een grotere staplengte toelaat dan het $NPJ(l)$ schema.

We voerden testen uit voor $l = 1$, $l = 2$ en $l = 3$. Tabel 4 geeft de waarde van l aan die overeenkomt met de resultaten in Tabel 2. Merk op dat het beste resultaat bij het Olmstead model steeds bekomen werd voor $l = 1$. De theoretische convergentiesnelheid voor de gekozen parameterwaarden is vrij hoog en wordt in de praktijk niet bereikt. Het overgangsgedrag van de Newtonmethode domineert de convergentiesnelheid. Dit komt omdat het spectrum van de systeemmatrix voor het Olmstead model heel sterk rond 0 geclusterd is.

2.6. Een praktische convergentieanalyse

Laten we terugkeren naar het algemene schema besproken in deel 2.2. Veronderstel dat de verschillende onbekenden Δq_* bekomen worden door het benaderend oplossen van het stelsel (14) en dat $\Delta\bar{p}$, ΔT en $\Delta\gamma$ berekend worden als de oplossing van het stelsel (15). Veronderstellen we verder dat de berekening van de residu $r(x(0), T, \gamma)$ en van matrix–vectorprodukten met de systeemmatrix met een voldoende hoge nauwkeurigheid gebeurt. Stel

$$\left[\; Qr_r \quad Qr_T \quad Qr_\gamma \;\right] := \left[\; Qr \quad Qb_T \quad Qb_\gamma \;\right] + (QMQ - I)\left[\; \Delta q_r \quad \Delta q_T \quad \Delta q_\gamma \;\right], \qquad (29)$$

Schema	Brusselator model				Olmstead model		Totaal
Tak	I	II	III	IV	1	2	
Begin: $L/R =$	5.55	1.04	1.55	1.30	0.623	1.209	
$T =$	3.017	2.948	2.947	3.433	14.06	14.27	
Einde: $L/R =$	2.0	2.0	2.0	2.0	1.267	1.24	
$T =$	3.424	3.436	3.197	3.415	27.5	18.0	
$NPJ(l)$	1045	1611	959	1463	864	385	6327
$NPGS(l)$	723	721	682	950	832	412	4320
$CRP(l)$	905	852	746	919	1238	539	5199
$CNP(l)$	987	932	753	998	1408	662	5740
Gemiddelde	915	1029	785	1083	1086	500	5397
Chord-Newton	2053	1607	1128	1874	9229	3791	19682

Tabel 2: Aantal op te lossen beginwaardeproblemen (beste resultaat).

Schema	Brusselator model				Olmstead model		Totaal
Tak	I	II	III	IV	1	2	
$NPJ(l)$	16	21	11	18	16	7	89
$NPGS(l)$	11	9	7	12	15	7	61
$CRP(l)$	13	9	7	11	23	8	71
$CNP(l)$	13	9	7	11	23	9	72
Gemiddelde	13	12	8	13	19	8	73
Chord-Newton	9	7	5	8	22	9	60

Tabel 3: Aantal gegenereerde punten in de voortzettingsmethode voor het beste resultaat.

d.w.z. Qr_r, Qr_T en Qr_γ zijn de N-dimensionale residus van het stelsel (14). We kunnen dan aantonen dat

$$\begin{aligned} &Qr(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = \\ &\qquad Qr_r + \Delta T\, Qr_T + \Delta\gamma\, Qr_\gamma + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \tag{30}$$

en

$$\left\{ \begin{array}{rcl} Pr(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ s(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ n(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{array} \right. \tag{31}$$

Deze formules geven het verband tussen de residu na een Newton–Picard stap (Δq, Δp, ΔT, $\Delta\gamma$) in functie van de hogereordetermen, een term voor de nauwkeurigheid van de basis en de residus van de lineaire Q-stelsels. Merk op dat we in de afleiding nergens veronderstelden dat ieder van de Δq_*'s berekend werd door middel van een aantal Picard iteratiestappen. Gelijkaardige formules kunnen ook worden afgeleid wanneer de parameterwaarde γ vastgehouden wordt. Het verwaarlozen van de termen $V_p^T M V_q$, $c_s^T V_q$ en $c_n^T V_q$ in (12) introduceert termen $PM\Delta q$, $c_s^T\Delta q$ en $c_n^T\Delta q$ in (31). Het verwaarlozen van de term $V_q^T b_T$ in (12) komt overeen met $Qr_T = Qb_T$ in (30). Wanneer l Picard stappen

Schema	Brusselator model				Olmstead model		Gemiddelde
Tak	I	II	III	IV	1	2	
$NPJ(l)$	2	1	1	1	1	1	1.17
$NPGS(l)$	2	2	3	1	1	1	1.67
$CRP(l)$	1	1	3	2	1	1	1.50
$CNP(l)$	1	1	2	1	1	1	1.17

Tabel 4: Waarde voor l voor het resultaat in de vorige tabellen.

gebruikt worden voor het benaderend oplossen van een van de onbekende vectoren $\Delta\bar{q}_*$ in (14), krijgen we voor de overeenkomstige term Qr_*

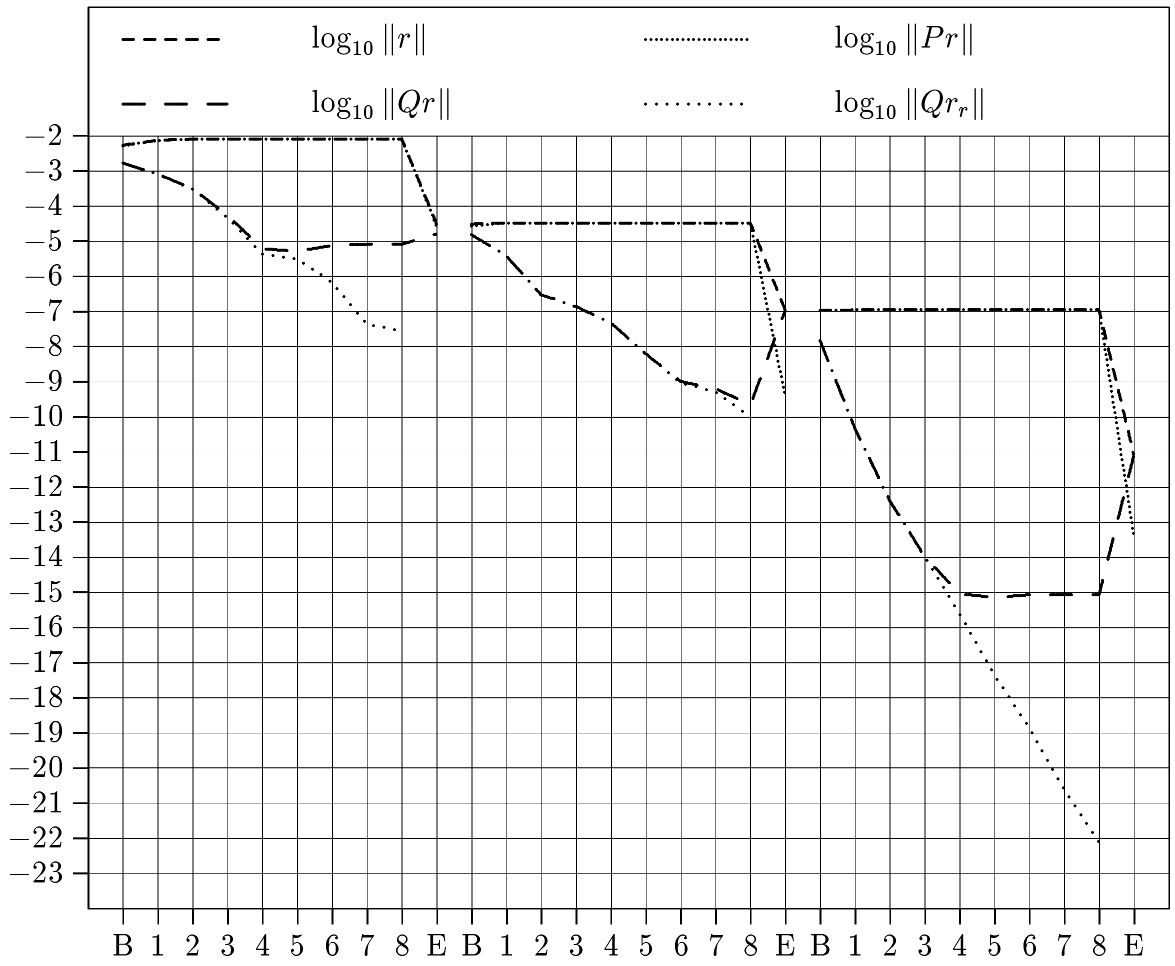
$$Qr_* = V_q \left(V_q^T M V_q\right)^l V_q^T r_*.$$

Convergentiegedrag van de verschillende varianten. De formules (30) en (31) laten toe om het convergentiegedrag van de varianten uit deel 2.3 te bestuderen. In het proefschrift maken we een grondige analyse van de invloed van elk van de termen op het convergentiegedrag. We komen tot de volgende conclusies:

1. Het verwaarlozen van de koppelterm $V_p^T M V_q$ kan ervoor zorgen dat het onmogelijk is op het einde van de Newton–Picard step een P-projectie van de residu te krijgen die gevoelig lager is dan de Q-projectie van de residu in het begin van de stap. Met andere woorden, voor een dergelijk schema is de P-projectie van de residu voortdurend hoger dan de Q-projectie. Dit houdt ook in dat de basis met een vrij hoge nauwkeurigheid berekend moet worden, vermits we een vrij grote norm voor $\Delta\bar{p}$ kunnen verwachten.

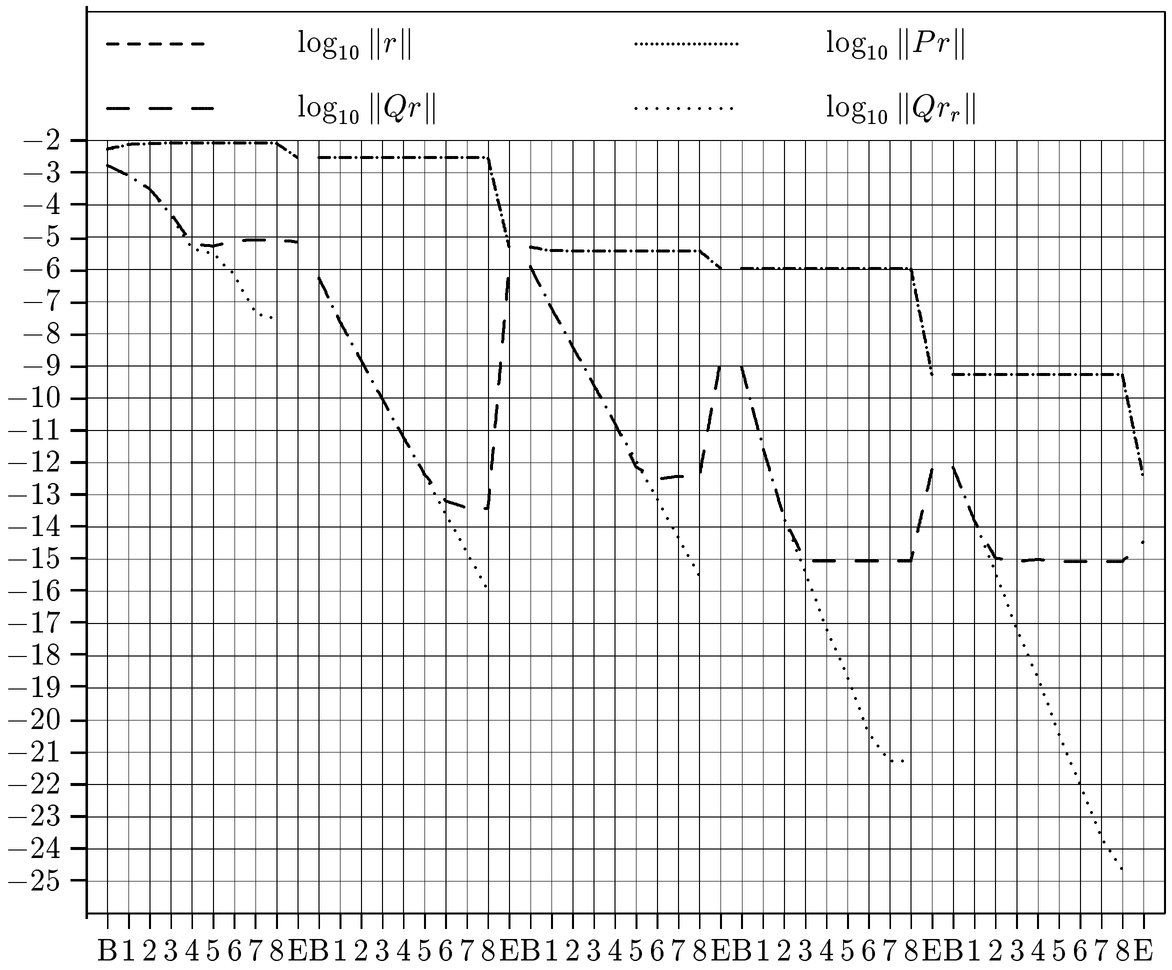
2. Voor een vooraf bepaalde waarde van l kunnen we zowel bij het $CRP(l)$ als bij het $CNP(l)$ schema convergentieproblemen krijgen wanneer het Picard schema initieel divergeert voor het rechterlid Qb_γ.

Ter illustratie presenteren we convergentiegrafieken voor het $NPGS(8)$ en $NPJ(8)$ schema in respectievelijk Figuur 3 en Figuur 4. We voerden acht Picard stappen uit omdat bij een hoge convergentiesnelheid de verschillen tussen de schema's het best zichtbaar zijn. In beide gevallen werd dezelfde periodieke oplossing berekend—een stabiele limietcyclus op tak I voor het Brusselatormodel (7–8) voor $L = 0{,}9158$—en werd ongeveer dezelfde startwaarde gebruikt. De letter B duidt het begin van de Newton–Picard stap aan, de cijfers 1 t.e.m. 8 komen overeen met de 8 Picard iteratiestappen en de letter E duidt het einde van de Newton–Picard stap aan. We hebben de Picard correctie uitgevoerd op de periodieke oplossing na iedere Picard stap en de residu $r(x(0), T, \gamma)$ opnieuw berekend zodat duidelijk wordt wanneer hogere orde termen een rol spelen. De gebruikte waarde voor ρ varieerde van stap tot stap. We stelden een maximale waarde voor ρ voorop tijdens de berekening, maar gebruikten alle voldoende nauwkeurige vectoren in de basis. De verandering van de basisgrootte is verantwoordelijk voor de veranderende convergentiesnelheid van het Picard schema en voor de sprongen in de projecties van de



Figuur 3: Convergentieverloop voor het $NPGS(8)$ schema. De grootte van de basis voor de eerste Newton-Picard stap was 2 en voor de volgende stappen respectievelijk 3 en 6.

residu tussen het einde van een Newton–Picard stap en het begin van de volgende stap. In het begin van de iteraties vallen de curves voor $\|Qr\|$ en $\|Qr_r\|$ samen, maar na een aantal stappen breekt de convergentie van $\|Qr\|$ af. Dit komt door de hogere orde termen in (30). Merk ook op dat tijdens de eerste Picard stappen $\|Pr\|$ verandert wegens de koppelterm $PM\Delta q$. In Figuur 3 merken we dat tijdens het Newton gedeelte van de tweede en derde Newton–Picard stap de projectie van $\|Qr\|$ toeneemt. Tijdens de tweede stap is dat te wijten aan de term Qb_T en tijdens de laatste stap aan de term $QM\Delta p$. We merken dat in dit geval tijdens de eerste iteratiestap het convergentiegedrag bepaald wordt door de hogere orde termen, tijdens de tweede convergentiestap door de term Qb_T en tijdens de derde stap door de term $QM\Delta p$, dat terwijl de theoretische analyse voorspelde dat de convergentie van het Picard schema de convergentie van de Newton–Picard methode bepaalt! In Figuur 4, voor het $NPJ(8)$ schema, krijgen we een enigzins ander beeld. Het convergentieverloop is heel grillig. Tijdens de eerste stap neemt de P-projectie van de residu nauwelijks af door de grote term $PM\Delta q$ die verwaarloosd wordt in dit schema. Bij de aanvang van de tweede Newton–Picard stap is de Q-projectie van de residu heel wat kleiner dan de P-projectie, zodat Δq en dus $PM\Delta q$ relatief klein zijn. Nu neemt de norm van de Q-projectie echter zelfs een beetje toe tijdens de Newton–Picard stap door



Figuur 4: Convergentieverloop voor het $NPJ(8)$ schema. De grootte van de basis voor de eerste Newton-Picard stap was 2 en voor de volgende stappen respectievelijk 5, 5, 6 en 6.

de invloed van de termen $\Delta T\, Qb_T$ en $QM\Delta p$! De derde stap toont een gelijkaardig beeld als de eerste stap. In de vierde en de vijfde stap is de basis zo nauwkeurig dat beide projecties van de residu behoorlijk afnemen.

Robuustere technieken door controle van de convergentie We kunnen de relaties (30) en (31) ook gebruiken om een heel robuust schema op te bouwen indien we het convergentiegedrag proberen te controleren. Op het einde van de iteratiestap is ieder van de termen in (30) en (31) immers eenvoudig te berekenen. De berekening van $r(x(0) + \Delta p + \Delta q, T + \Delta T, \gamma + \Delta\gamma)$ vereist een tijdsintegratie, maar deze tijdsintegratie is toch vereist aan het begin van de volgende Newton–Picard stap of om te controleren of het schema geconvergeerd is. De termen Qr_r, Qr_T en Qr_γ kunnen eenvoudig berekend worden tijdens de Picard iteratie. Het matrix–vectorprodukt dat hiervoor nodig is, is ook nodig voor de opbouw van het stelsel (15). We kennen ook MV_p uit de deelruimteïteratiestappen, en bijgevolg ook $QM\Delta p$. We kunnen dus eenvoudig de termen van hogere orde berekenen. Vijf factoren kunnen de convergentie van de Newton–Picard schema's beïnvloeden:

- De hogere orde termen die verwaarloosd worden door de Newton linearisatie beperken de verbetering in de residu die we kunnen bekomen in één Newton–Picard step. Wanneer de hogere orde termen zo groot zijn dat het Newton schema niet zou convergeren, kunnen we die controleren door demping.

- De nauwkeurigheid waarmee de verschillende onbekende vectoren in het stelsel (14) berekend worden, heeft ook een invloed. Deze termen kunnen we controleren door het aantal Picard iteratiestappen te laten variëren en de residu's Qr_* te testen. Dit vergt geen extra rekenwerk vermits de producten $M\Delta q_*$ toch nodig zijn voor de constructie van het P-stelsel (15). Merk op dat de afleiding ook toelaat om andere technieken te gebruiken voor het oplossen van (14), bijvoorbeeld de GMRES methode.

- De onnauwkeurigheid van de basis beperkt de mogelijke afname van de Q-projectie van de residu. Deze term kunnen we controleren door het aantal deelruimteiteratiestappen te laten variëren. Merk op dat een onnauwkeurige basis ook kan leiden tot een verlies van de contractiviteit van het Picard schema.

- Het onnauwkeurig oplossen van het P-stelsel (15) introduceert extra termen in (31). Vermits dit stelsel echter een heel klein stelsel is, kunnen we ons goede en dure technieken veroorloven voor de oplossing van dit stelsel. Heel goede resultaten werden bekomen met het gebruik van een kleinste-kwadratentechniekgebaseerd op de singuliere waardenontbinding i.p.v. Gauss eliminatie met partiële pivotering.

- Om de analyse te kunnen gebruiken, hebben we aangenomen dat de matrix-vectorprodukten met de systeemmatrix en de bereking van $r(x(0), T, \gamma)$ in de verschillende punten met voldoende nauwkeurigheid gebeurt. Het is nodig om goede tijdsintegratietechnieken te gebruiken.

In het proefschrift stellen we voor om de relaties (30) en (31) op het einde van de Newton–Picard stap te gebruiken om te controleren of de convergentie verliep zoals gewenst en in het begin van de Newton–Picard stap om convergentiecriteria vast te leggen voor de verschillende oplossingen $\Delta\bar{q}_*$ in (30) en voor de deelruimteïteraties. Dit laatste gebeurt als volgt: we kennen natuurlijk de residu $r(x(0), T, \gamma)$ in het begin van de Newton–Picard stap. We proberen aan de hand van resultaten in de vorige Newton–Picard stappen de hogere orde termen te schatten. Dan beslissen we welke verbetering van de residu we willen bekomen. Merk op dat we hier kunnen opteren voor lineaire convergentie, maar even goed kwadratische convergentie van de Newton–Picard stappen kunnen bekomen door te eisen dat de residu op het einde van de stap van dezelfde grootteorde is als de hogereordetermen. We proberen ook een schatting te maken voor Δp, ΔT en $\Delta\gamma$. Aan de hand van die schatting kunnen we dan convergentiecriteria vastleggen voor de residu's Qr_* en voor de nauwkeurigheid van de basis.

Merk op dat we bij het afleiden van het Newton–Picard schema niet veronderstelden dat we een Picard iteratieschema zouden gebruiken voor het oplossen van (14) of Gauss eliminatie voor het oplossen van (15). GMRES is een goed alternatief voor de Picard iteraties. Immers, een l-staps Picard iteratieschema zoekt een welbepaalde oplossing Δq_* in $\mathcal{K}_l(QMQ, Qr_*)$, de l-dimensionale Krylov deelruimte voor de matrix QMQ opgebouwd op basis van de vector Qr_*. Het GMRES schema zal de Δq_* in die deelruimte selecteren die de laagste norm voor de residu Qr_* oplevert, en presteert dus zeker even goed als het Picard schema. Het is soms ook interessant om Gauss eliminatie te vervangen door een kleinste-kwadratenmethode gebaseerd op de singuliere waardenontbinding. In dit laatste geval kunnen we zonder problemen de pseudo-booglengteconditie weglaten en een ondergedetermineerd stelsel oplossen. In principe kunnen we ook de fasevoorwaarde weglaten, maar dat leverde soms problemen op tijdens onze testen.

2.7. Gewone differentiaalvergelijkingen met vertraginstermen

In samenwerking met K. Engelborghs (K.U.Leuven, departement computerwetenschappen) en T. Luzyanina (Institute for Mathematical Problems in Biology, RAS, Pushchino, Rusland) hebben we de Newton–Picard techniek uitgebreid tot de berekening van periodieke oplossingen van stelsels van gewone differentiaalvergelijkingen met vertraginstermen. Dit zijn vergelijkingen van de vorm

$$\frac{dx(t)}{dt} = f(x(t), x(t-\tau_1), \ldots, x(t-\tau_k), \gamma) \tag{32}$$

waarbij $\tau_1, \ldots, \tau_k$ de k discrete en eindige vertragingen zijn.

$$f : \mathbb{R}^n \times \mathbb{R}^n \times \cdots \times \mathbb{R}^n \times \mathbb{R} = \mathbb{R}^{(k+1)\times n+1} \mapsto \mathbb{R}^n$$

is een vectorfunctie. Merk op dat n eventueel 1 kan zijn. Onze techniek zal ook in dat geval nog efficient zijn. We duiden de grootste vertraging aan met de letter τ, d.w.z.

$$\tau = \max_i \tau_i.$$

Om een traject van (32) uniek vast te leggen moet een beginwaarde over het inteval $[-\tau, 0]$ gespecifieerd worden. We duiden die beginvoorwaarde aan met de Griekse letter ϕ, d.w.z.

$$\phi : [-\tau, 0] \mapsto \mathbb{R}^n \tag{33}$$

is een continue n-dimensionale functie gedefinieerd op het interval $[-\tau, 0]$. We duiden met $\varphi(\phi, t, \gamma)$ de oplossing op het tijdstip t van (32) met de beginfunctie ϕ aan.

$$x_T(\phi, \gamma) : [-\tau, 0] \mapsto \mathbb{R}^n : x_T(\phi, \gamma)(\theta) = \varphi(\phi, T + \theta, \gamma)$$

duidt een segment van lengte τ aan van de oplossing van (32) verschoven van tijd T naar 0. $x_T(\phi, \gamma)$ is een functie over een interval van lengte τ en geparameteriseerd door de beginfunctie ϕ en de scalaire systeemparameter γ.

We kunnen een periodieke oplossing voor een gegeven parameterwaarde γ berekenen als de oplossing van het stelsel

$$\left\{ \begin{array}{rcl} r(\phi, T) & = & x_T(\phi) - \phi = 0, \\ s(\phi, T) & = & 0. \end{array} \right. \tag{34}$$

Merk hierbij op dat ϕ een functie is. Wanneer we dit stelsel met de Newtonmethode oplossen, moeten we herhaaldelijk lineaire stelsels van de vorm

$$\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} \\ c_s^{T(\nu)} & d_{s,T}^{(\nu)} \end{bmatrix} \begin{bmatrix} \Delta\phi^{(\nu)} \\ \Delta T^{(\nu)} \end{bmatrix} = - \begin{bmatrix} r(\phi^{(\nu)}, T^{(\nu)}) \\ s(\phi^{(\nu)}, T^{(\nu)}) \end{bmatrix} \tag{35}$$

met

$$\begin{bmatrix} M^{(\nu)} & b_T^{(\nu)} \\ c_s^{T^{(\nu)}} & d_{s,T}^{(\nu)} \end{bmatrix} = \left.\frac{\partial(r,s)}{\partial(\phi,T)}\right|_{(\phi^{(\nu)},T^{(\nu)})}$$

oplossen. Wanneer (ϕ, T) een periodieke oplossing bepaalt, is M de monodromyoperator. Het spectrum van deze operator is compact en geclusterd rond 0 (zie [50]). Om de periodieke oplossing te berekenen, discretiseren we de beginfunctie ϕ. Hiertoe kiezen we een tijdsrooster met m roosterpunten en vervangen we ϕ door een nm-dimensionale vector die de n-dimensionale functie ϕ voorstelt in de m roosterpunten. $x_T(\phi)$ duidt nu de oplossing op het interval $[T-\tau, T]$ aan, gebruik makende van hetzelfde tijdsrooster als ϕ maar opgeschoven naar $[T-\tau, T]$. Het stelsel (34) wordt dan een $(nm+1)$-dimensionaal stelsel. Het gelineariseerde stelsel heeft delzelfde vorm als (35), maar is een $(nm+1) \times (nm+1)$-dimensionaal stelsel. De matrix M heeft dezelfde kenmerken als bij partiële differentiaalvergelijkingen. De grotere eigenwaarden zijn ongeveer onafhankelijk van de gebruikte tijdsdiscretisatie, terwijl eigenwaarden heel dicht bij 0 toegevoegd worden wanneer we m verhogen. Bijgevolg kunnen we opnieuw de Newton–Picard techniek toepassen. De uitwerking van de techniek is eenvoudig en gebeurt in het proefschrift. We presenteren ook een aantal testvoorbeelden in het proefschrift.

2.8. Vergelijking met ander werk

Onze methode bouwt voort op werk van Jarausch en Mackens [56, 57, 58] en Shroff and Keller [107]. Er werden reeds verschillende andere uitbreidingen gemaakt op deze technieken. We vermelden kort het werk van Burrage, Erhel, Pohl en Williams [13, 14, 36, 118, 119] en later werk van Jarausch [54, 55].

Werk van Jarausch en Mackens.

Jarausch en Mackens ontwikkelden de "gecondenseerde Newton–ondersteunende Picard" methode in het begin van de jaren '80 om oplossingen te berekenen van het stelsel

$$Ax = f(x, \gamma).$$

Hierbij is A een symmetrisch positief definiete matrix en is de jacobiaan $\partial f/\partial x$ van f een symmetrische matrix. Zij veronderstellen ook dat er een efficient schema voor handen is om stelsels $Ax = b$ op te lossen. Vermits zowel A als $\partial f/\partial x$ symmetrisch zijn, kunnen ze een opsplitsing van het stelsel bekomen in twee stukken die volledig onafhankelijk van elkaar zijn (indien de parameter γ tenminste constant gehouden wordt). Zij beschrijven ook een techniek om het convergentiegedrag te controleren en te beslissen of een basisaanpassing, een Q-stap of een P-stap aangewezen is. Door de volledige ontkoppeling van het hoogdimensionale Q-stelsel en het laagdimensionale P-stelsel verschilt hun techniek aanzienlijk van de onze. Ze hebben hun techniek ook uitgebreid voor pseudo-booglengtevoortzetting, gecombineerd met demping en uitgebreid tot de berekening van een aantal types bifurcatiepunten waarbij ze een uitgebreid systeem op basis van het P-stelsel gebruiken.

Het werk van Shroff en Keller.

De hoofddoelstelling van het werk van Shroff en Keller is het stabiliseren van een rekenschema

$$x^{(\nu+1)} = f(x^{(\nu)}, \gamma) \tag{36}$$

en het eventueel versnellen van de convergentie. In hun werk gaan ze ervan uit dat er vaak een code beschikbaar is om een bepaald probleem op te lossen, maar dat die code faalt in bepaalde parameterintervallen. Ze willen die code ook in die intervallen kunnen gebruiken met zo weinig mogelijk wijzigingen aan de code zelf. Zij doopten hun techniek de “recursieve projectiemethode”. Onze doelstelling daarentegen was een goede maar te dure techniek—Newton met een directe methode voor het oplossen van de lineaire stelsels—goedkoper te maken, en dit leidt tot een aantal belangrijke verschillen. Hun voornaamste toepassing is de berekening van evenwichtsoplossingen van partiële differentiaalvergelijkingen. In later werk hebben Keller en Von Sosen de methode ook uitgebreid tot de berekening van evenwichtsoplossingen van stelsels differentiaalvergelijkingen met algebraische beperkingen [115, 64].

Hun schema voor een vaste waarde van de parameter komt sterk overeen met onze $NPJ(1)$ methode, en hun schema voor pseudo-booglengtevoortzetting is equivalent met onze $CRP(1)$ methode. Vermits hun werk vertrekt van een ander uitgangspunt dan het onze, zijn er wel een aantal verschillen in de techniek om de basis aan te passen. Onze strategie is volledig op robuustheid afgestemd, terwijl de strategie gebruikt door Shroff en Keller eerder op een maximale performantie afgestemd is. Onze techniek breidt de techniek van Shroff en Keller ook uit op een belangrijk aantal punten:

- Wij ontwikkelden schema’s waar meedere Picard iteratiestappen gecombineerd worden met één Newton stap.

- Wij erkenden ook de belangrijke invloed van de koppeling tussen het Q- en het P-stelsel (de term $V_p^T M V_q$ in (12)) voor problemen met een niet-normale matrix en ontwikkelden schema’s die rekening houden met die term.

- Onze techniek voor voortzettingstechnieken heeft als voordeel dat er meer informatie over bifurcatiepunten bewaard blijft in het P-stelsel. Wij kunnen een transkritisch- of stemvorkbifurcatiepunt detecteren in het P-stelsel wanneer we onze meest geavanceerde varianten gebruiken en alle vectoren $\Delta\bar{q}_*$ in (14) voldoende nauwkeurig berekenen. Shroff en Keller kunnen dit enkel indien $V_p^T M V_p$ een normale matrix is.

- Wij ontwikkelden ook een techniek om de convergentie van onze meer geavanceerde schema’s te controleren, wat leidt tot heel robuuste methodes. In onze meer geavanceerde technieken kunnen we ook eenvoudig een directe methode combineren met de GMRES techniek.

Merk op dat het niet triviaal is om een efficiënte techniek te bekomen voor evenwichtsoplossingen van partiële differentiaalvergelijkingen. Eenvoudige Picarditeratieschema’s zoals expliciete tijdsintegratieschema’s hebben vaak veel eigenwaarden dicht bij de eenheidscirkel. Volledig impliciete tijdsintegratieschema’s hebben weinig zin, vermits dan in iedere tijdstap een niet-lineair stelsel opgelost moet worden dat heel gelijkaardig is aan het probleem dat we proberen op te lossen met de recursieve projectiemethode. Er is zeker nog nood aan verder onderzoek op dat gebied.

Het werk van Burrage, Erhel, Pohl en Williams.

In [13] leiden Burrage, Erhel en Pohl een techniek af gebaseerd op de recursieve projectiemethode van Shroff en Keller om de convergentie van een traditioneel Jacobi- of Gauss-Seideliteratieschema en andere gelijkaardige iteratieschema's voor de oplossing van stelsels van lineaire vergelijkingen te versnellen. Naast de variante van Shroff en Keller stellen ze nog twee varianten voor: in de ene variant gebeurt er eerst een Q-stap en wordt de nieuwe waarde voor q gebruikt in de P-stap (equivalent met onze Gauss-Seidel variante), de tweede variante doet eerst een P-stap en gebruikt daarna de nieuwe p in de Q-stap. Ze bestuderen ook theoretisch de invloed van een onnauwkeurige basis op het convergentiegedrag van het schema. Merk op dat de auteurs enkel het asymptotisch convergentiegedrag voor een lineair stelsel bespreken. Zij merken nauwelijks verschil tussen de convergentie van beide nieuwe varianten. In onze context, waar we niet-lineaire stelsels oplossen en na iedere Newton–Picard stap een nieuwe linearisatie doorvoeren, heeft de variante waar eerst het P-stelsel opgelost wordt weinig zin. In die variante wordt immers de kleine koppelterm $V_q^T M V_p$ in rekening gebracht terwijl de veel grotere term $V_p^T M V_q$ verwaarloosd wordt, en dit heeft een grote invloed op het convergentiegedrag in de eerste stappen. De methode wordt gebruikt in [14, 118, 119] om een veralgemeend kruisvalidatieprobleem op te lossen.

Een verschillend en slechts vaag verwant idee wordt voorgesteld in [36]. In dit artikel bestuderen de auteurs een probleemtransformatie (engels: "preconditioner") voor de "flexible GMRES" methode [102]. Dit is een GMRES methode met herstart waarbij na iedere herstart een nieuwe transformatie van het probleem gebruikt wordt. De bedoeling is om spectrale informatie verzameld tijdens de GMRES stappen te gebruiken om het probleem te transformeren naar een probleem waarvoor de GMRES methode sneller convergeert.

Later werk van Jarausch. Jarausch heeft de idee om twee numerieke technieken te combineren nog verder onderzocht.

In [54] beschrijft hij twee opsplitsingen om gediscretiseerde parabolische partiële differentiaalvergelijkingen te integreren. De technieken zijn ook geschikt voor niet-symmetrische problemen. De ene opsplitsing steunt op orthogonale projectoren en is gelijkaardig aan de opsplitsing die wij gebruiken. In de tweede techniek gebruikt hij schuine projectoren gebaseerd op eigenruimten. Deze techniek leidt tot een volledige opsplitsing, maar vereist wel kennis van een aantal linkse (veralgemeende) eigenvectoren.

In [55] beschrijft hij een techniek voor de berekening van evenwichtsoplossingen en voor de berekening van periodieke oplossingen via enkelvoudig schieten. Hier ontwikkelt hij een ontkoppeling gebaseerd op orthogonale projectoren die toch een volledige ontkoppeling oplevert. Dit doet hij door gebruik te maken van ruimten overspannen door singuliere vectoren en een rotatie. De aanpak heeft echter een aantal nadelen ten opzichte van de onze.

- De berekening van de benodigde singuliere vectoren en de rotatie is moeilijk.
- Zijn techniek voor de berekening van periodieke oplossingen gebruikt een impliciet gedefinieerde fasevoorwaarde en pseudo-booglengteconditie. Onze experimenten met een kleinste-kwadratentechniek voor het oplossen van het P-stelsel hebben geleerd dat een dergelijke fasevoorwaarde soms problemen geeft.
- In onze techniek verkrijgen we de dominante Floquet vermenigvuldigers als een bijprodukt van de opsplitising. Dit is niet het geval in [55]. De singuliere waardendecompositie die gebruikt wordt, is gebaseerd op de matrix met de extra kolommen voor de afgeleiden naar de periode T en de parameter γ. Het is ook niet duidelijk of informatie over periodeverdubbelingspunten en torusbifurcatiepunten bewaard blijft in het laagdimensionale stelsel in [55].
- Zoals we verderop zullen zien, kan onze aanpak eenvoudig veralgemeend worden tot meervoudig schieten en de berekening van allerhande bifurcatiepunten.

De aanpak van [55] heeft echter ook een aantal voordelen.

- Door de aard van de opsplitsing kan Jarausch monotone convergentie van het iteratieve schema voor het hoogdimensionale stelsel verzekeren. In onze techniek kunnen we tijdens de eerste stappen een groei van de residu krijgen.
- De volledige ontkoppeling maakt het iets gemakkelijker om strategieën te ontwikkelen om de convergentie te controleren.

3. Technieken gebaseerd op meervoudig schieten

3.1. Inleiding

Bij enkelvoudig schieten proberen we een globaal object, de limietcyclus, voor te stellen door één punt. Dit leidt tot een aantal problemen. Wanneer de limietcyclus sterk onstabiel is, kan het attractiegebied van de Newtonmethode klein zijn en kunnen we convergentieproblemen krijgen. We kunnen de eigenschappen van het algoritme verbeteren door meer punten te gebruiken om de limietcyclus voor te stellen. Dit leidt tot een techniek gekend als “meervoudig schieten”.

Beschouw een partitie van het eenheidsinterval

$$s_i, \quad i = 0, \ldots, m \tag{37}$$

met

$$0 = s_0 < s_1 < \cdots < s_{m-1} < s_m = 1.$$

We kunnen de periodieke oplossing voorstellen door de m punten

$$x_i = x(s_i T), \quad i = 0, \ldots, m-1.$$

Stel

$$\nabla s_i = s_i - s_{i-1}.$$

Meervoudig schieten berekend de onbekende punten x_i, $i = 0, \ldots, m-1$, T en γ uit het niet-lineaire stelsel

$$\left\{ \begin{array}{rcl} \varphi(x_0, \nabla s_1 T, \gamma) - x_1 & = & 0, \\ \varphi(x_1, \nabla s_2 T, \gamma) - x_2 & = & 0, \\ & \vdots & \\ \varphi(x_{m-2}, \nabla s_{m-1} T, \gamma) - x_{m-1} & = & 0, \\ \varphi(x_{m-1}, \nabla s_m T, \gamma) - x_0 & = & 0, \\ s(x_0, x_1, \ldots, x_{m-1}, T, \gamma) & = & 0, \\ n(x_0, x_1, \ldots, x_{m-1}, T, \gamma; \eta) & = & 0. \end{array} \right. \tag{38}$$

De Newton–Raphson techniek leidt tot een sequentie van lineaire stelsels van de vorm

$$\begin{bmatrix} G_1 & -I & & & b_{T,1} & b_{T,1} \\ & G_2 & -I & & b_{T,2} & b_{T,2} \\ & & \ddots & \ddots & \vdots & \vdots \\ -I & & & G_m & b_{T,m} & b_{T,m} \\ c_{s,1}^T & c_{s,2}^T & \cdots & c_{s,m}^T & d_{s,T} & d_{s,\gamma} \\ c_{n,1}^T & c_{n,2}^T & \cdots & c_{n,m}^T & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_{m-1} \\ \Delta T \\ \Delta \gamma \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \\ s \\ n \end{bmatrix}, \tag{39}$$

met

$$G_i = \left. \frac{\partial \varphi(x, T, \gamma)}{\partial x} \right|_{(x_{i-1}, \nabla s_i T, \gamma)} . \tag{40}$$

In de oplossing van (38) is $G_m G_{m-1} \cdots G_1$ de systeemmatrix geëvalueerd in het punt x_0 en is $b_{T,m}$ de eigenvector voor de triviale Floquet vermenigvuldiger. $G_1 G_m \cdots G_2$ is de systeemmatrix in het punt x_1 en $b_{T,1}$ de bijhorende eigenvector voor de triviale Floquet vermenigvuldiger, enz.

Merk op dat de numerieke techniek om (39) op te lossen met zorg gekozen moet worden. Technieken die enkel op een voorwaartse recursie steunen zijn onstabiel wannneer de limietcyclus onstabiel is. In het proefschrift stellen we twee technieken voor die steunen op de methodes uit [4] om dit stelsel op een numeriek stabiele manier op te lossen gebruik makende van een combinatie van een voorwaartse en een achterwaartse recursiebetrekking. De opsplitsing van het stelsel in een deel dat met voorwaartse recursie opgelost wordt en een deel waarvoor achterwaartse recursie gebruikt moet worden, wordt gegeven door de periodieke Schur ontbinding [11]. We stellen eerst kort de deze ontbinding voor, waarna we de principes van de veralgemening van de Newton–Picard techniek naar meervoudig schieten aangeven.

3.2. De periodieke Schur ontbinding

Uit het vorige deel herinneren we dat de Floquet vermenigvuldigers de eigenwaarden zijn van de matrix $G_m \cdots G_1$. De Floquet vermenigvuldigers kunnen worden berekend door expliciet het product van deze matrices te vormen en dan het QR-algoritme te gebruiken om de eigenwaarden te vinden. Dit is echter geen erg goede oplossing wanneer er heel grote eigenwaarden zijn. Deze zorgen immers voor een verlies aan nauwkeurigheid voor de kleinere eigenwaarden, terwijl juist de eigenwaarden met modulus rond 1 het meest interessant zijn. Zij verschaffen ons immers informatie over mogelijke bifurcatiepunten.

Zoals we later zullen zien zijn we bovendien niet enkel geïnteresseerd in de eigenvectoren of Schur vectoren van de matrix $G_m \cdots G_1$, maar ook in deze vectoren voor de systeemmatrices in de andere punten x_i van de methode voor meervoudig schieten.

Een gelijkaardige situatie treedt op wannneer Gauss-Legendrecollocatie gebruikt wordt om een periodieke oplossing te berekenen, zoals in het softwarepakket AUTO ([27]). Daar kunnen de Floquet vermenigvuldigers berekend worden als de eigenwaarden van een product

$$H_m^{-1} G_m \cdots H_1^{-1} G_1 \tag{41}$$

waar de matrices G_i en H_i bepaalde matrices zijn die optreden tijdens de oplossingsprocedure van het gelineariseerde stelsel. Ook hier is het een slecht idee het product expliciet te vormen.

Een oplossing voor dit probleem wordt gegeven door de periodieke Schur ontbinding voorgesteld in ([11]) en het bijhorende periodieke QR-algoritme voor de berekening ervan. Wij stellen hier enkel de reëele variante voor. Bojanczyk, Golub en Van Dooren bewijzen volgend lemma voor de matrix (41).

Lemma 2 *Veronderstel G_i, $i = 1, \ldots, m$ en H_i, $i = 1, \ldots, m$ zijn reële $N \times N$ matrices. Er bestaan orthogonale $N \times N$ matrices Q_i, $i = 0, \ldots, m-1$ en Z_i, $i = 1, \ldots, m$ zodanig dat*

$$\begin{aligned} \hat{G}_i &= Z_i^T G_i Q_{i-1} \\ \hat{H}_i &= Z_i^T H_i Q_i, \; Q_m := Q_0 \end{aligned}$$

waarbij alle matrices $\hat{G}_i$ en $\hat{H}_i$ bovendriehoeksmatrices zijn behalve de matrix $\hat{G}_m$. $\hat{G}_m$ is een quasi-bovendriehoeksmatrix met 1×1-blokken op de diagoneel voor de reële eigenwaarden en 2×2 blokken overeenkomend met de paren van complex toegevoegde eigenwaarden van $H_m^{-1} G_m \ldots H_1^{-1} G_1$.

De nummering van de matrices Q_i, startende vanaf 0, lijkt onlogisch op dit ogenblik. De indices komen overeen met de nummering van de randpunten van de roosterintervallen in de methode voor meervoudig schieten. Ieder van de matrices

$$Q_{k-1}^T H_{k-1}^{-1} G_{k-1} \cdots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_k^{-1} G_k Q_{k-1}$$

is een quasi-bovendriehoeksmatrix. Bijgevolg kunnen we uit de periodieke Schur ontbinding eenvoudig de Schur ontbinding afleiden voor ieder van de matrices

$$H_{k-1}^{-1} G_{k-1} \ldots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_k^{-1} G_k.$$

We kunnen ook eenvoudig de eigenwaarden afleiden uit de periodieke Schur ontbinding.

In dit proefschrift zijn we bijzonder geïnteresseerd in het geval $H_i = I$. In dit geval kan men het volgende lemma aantonen.

Lemma 3 *Veronderstel dat G_i, $i = 1, \ldots, m$ reële $N \times N$ matrices zijn. Er bestaan dan orthogonale $N \times N$ matrices Q_i, $i = 0, \ldots, m-1$ zodanig dat de matrices*

$$\hat{G}_i = Q_i^T G_i Q_{i-1}, \; Q_m := Q_0$$

allen bovendriehoeksmatrices zijn behalve de matrix $\hat{G}_m$. $\hat{G}_m$ is een quasi-bovendriehoeksmatrix met 1×1-blokken op de diagoneel voor de reële eigenwaarden en 2×2 blokken overeenkomend met de paren van complex toegevoegde eigenwaarden van $G_m \ldots G_1$.

Het is mogelijk om de periodieke Schur ontbinding te berekenen zonder eerst expliciet het product (41) of $G_m \cdots G_1$ te vormen. In [11] wordt hiertoe een algoritme voorgesteld. Dit algoritme is een veralgemening van het QR- en QZ-algoritme. Wij hebben dit algoritme verder uitgewerkt voor het geval van lemma 3, waarbij we wel rekening hielden met het gegeven dat ieder van de matrices G_i regulier is wanneer de matrices G_i komen van de linearisatie van het stelsel (38).

3.3. De Newton–Picard methode voor meervoudig schieten

We kunnen de ontbinding voorgesteld voor het algoritme voor enkelvoudig schieten veralgemenen tot het algoritme voor meervoudig schieten. Hierbij moeten we er wel rekening mee houden dat de eigenwaarden van de systeemmatrix in ieder punt x_i hetzelfde zijn, maar de eigenvectoren veranderen. Er is echter een verband tussen de eigenvectoren in de verschillende punten x_i gegeven door volgend lemma. In onze methode zullen dit verband uitbuiten.

Lemma 4 *Veronderstel dat $M_0 = G_m \cdots G_1$ regulier is. Stel $M_0 X_0 = X_0 \Lambda$ is de Jordanontbinding van M_0 Stel $M_j = G_j \cdots G_1 G_m \cdots G_{j+1}$ en $X_j = G_j \cdots G_1 X_0$. Dan is $M_j X_j = X_j \Lambda$ de Jordanontbinding van M_j.*

We veralgemenen veronderstelling 1.

Veronderstelling 5 *Veronderstel dat $y^* = (x_0^*, \ldots, x_{m-1}^*, T^*, \gamma^*)$ een geïsoleerde periodieke oplossing is van (38). Beschouw een kleine omgeving $\mathcal{B}$ van y^*. Stel $M(y) = G_m(y) \cdots G_1(y)$ voor $y \in \mathcal{B}$. We noemen de eigenwaarden van $M(y)$ μ_i, $i = 1, \ldots, N$. Veronderstel dat voor alle $y \in \mathcal{B}$ er precies p eigenwaarden buiten de schijf*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1$$

liggen en dat geen enkele eigenwaarde modulus ρ heeft. Dit houdt in dat voor alle $y \in \mathcal{B}$

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

Veronderstel dat we basissen $V_{p,i}$ hebben voor de deelruimtes $\mathcal{U}_i$ van $\mathbb{R}^N$ bepaald door de veralgemeende eigenvectoren van M_i, $i = 0, \ldots, m-1$ overeenkomend met de eigenwaarden μ_i, $i = 1, \ldots, p$ en basissen $V_{q,i} \in \mathbb{R}^{(N-p)\times N} = \mathbb{R}^{N \times q}$ voor $\mathcal{U}_i^\perp$. De matrices $V_{q,i}$ definiëren bases voor hoogdimensionale ruimten. Net als in het geval van enkelvoudig schieten zullen we de berekening van deze matrices vermijden. We hebben ze echter nodig om de techniek af te leiden. Men kan dan eenvoudig aantonen dat er reguliere matrices $L_i \in \mathbb{R}^{p \times p}$ bestaan zodat

$$G_i V_{p,i-1} = V_{p,i} L_i.$$

We construeren orthogonale projectoren P_i en Q_i op $\mathcal{U}_i$ en $\mathcal{U}_i^\perp$ volgens

$$\begin{array}{rcl} P_i & := & V_{p,i} V_{p,i}^T, \\ Q_i & := & V_{q,i} V_{q,i}^T = I - V_{p,i} V_{p,i}^T. \end{array} \qquad (42)$$

Om een aantal notaties te vereenvoudigen stellen we

$$V_{p,m} := V_{p,0}, V_{q,m} := V_{q,0}, P_m := P_0 \text{ en } Q_m := Q_0.$$

Elke vector $x_i \in \mathbb{R}^N$ kan dan worden ontbonden volgens

$$x_i = V_{p,i}\bar{p}_i + V_{q,i}\bar{q}_i, \quad p_i = V_{p,i}\bar{p}_i = P_i x_i, \quad q_i = V_{q,i}\bar{q}_i = Q_i x_i \tag{43}$$

waarbij $\bar{p}_i \in \mathbb{R}^p$ en $\bar{q}_i \in \mathbb{R}^{N-p}$. Na substitutie van (43) in (39), vermenigvuldiging van de i^{de} blokrij van N rijen met $[V_{q,i} \;\; V_{p,i}]^T$ langs links en de i^{de} blokkolom van N kolommen met $[V_{q,i-1} \;\; V_{p,i-1}]$ bekomen we het stelsel

$$\begin{bmatrix} F_{qq,1} & F_{qp,1} & -I_q & & & & & & V_{q,1}^T B_1 \\ F_{pq,1} & F_{pp,1} & & -I_p & & & & & V_{p,1}^T B_1 \\ & & \ddots & \ddots & \ddots & & & & \vdots \\ & & \ddots & \ddots & & \ddots & & & \vdots \\ & & & & F_{qq,m-1} & F_{qp,m-1} & -I_q & & V_{q,m-1}^T B_{m-1} \\ & & & & F_{pq,m-1} & F_{pp,m-1} & & -I_p & V_{p,m-1}^T B_{m-1} \\ -I_q & & & & & & F_{qq,m} & F_{qp,m} & V_{q,m}^T B_m \\ & -I_p & & & & & F_{pq,m} & F_{pp,m} & V_{p,m}^T B_m \\ C_1^T V_{q,0} & C_1^T V_{p,0} & \cdots & \cdots & \cdots & \cdots & C_m^T V_{q,m-1} & C_m^T V_{p,m-1} & D \end{bmatrix}$$
$$\begin{bmatrix} \Delta\bar{q}_0 \\ \Delta\bar{p}_0 \\ \vdots \\ \vdots \\ \Delta\bar{q}_{m-1} \\ \Delta\bar{p}_{m-1} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_{q,1}^T r_1 \\ V_{p,1}^T r_1 \\ \vdots \\ \vdots \\ V_{q,m}^T r_m \\ V_{p,m}^T r_m \\ s \\ n \end{bmatrix}, \tag{44}$$

waarbij

$$\begin{bmatrix} F_{qq,i} & F_{qp,i} \\ F_{pq,i} & F_{pp,i} \end{bmatrix} = \begin{bmatrix} V_{q,i}^T G_i V_{q,i-1} & V_{q,i}^T G_i V_{p,i-1} \\ V_{p,i}^T G_i V_{q,i-1} & V_{p,i}^T G_i V_{p,i-1} \end{bmatrix}, \quad i = 1, \cdots, m, \tag{45}$$

$$B_i = \begin{bmatrix} b_{T,i} & b_{\gamma,i} \end{bmatrix}, \; C_i = \begin{bmatrix} c_{s,i} & c_{n,i} \end{bmatrix} \text{ en } D = \begin{bmatrix} d_{s,T} & d_{s,\gamma} \\ d_{n,T} & d_{n,\gamma} \end{bmatrix}.$$

Laat ons de volgende notaties invoeren:

$$F_{qq}^o = \begin{bmatrix} F_{qq,1} & -I_q & \\ & \ddots & \ddots \\ -I_q & & F_{qq,m} \end{bmatrix}, \; F_{qp}^o = \begin{bmatrix} F_{qp,1} & & \\ & \ddots & \\ & & F_{qp,m} \end{bmatrix},$$

$$F_{pq}^o = \begin{bmatrix} F_{pq,1} & & \\ & \ddots & \\ & & F_{pq,m} \end{bmatrix}, \; F_{pp}^o = \begin{bmatrix} F_{pp,1} & -I_p & \\ & \ddots & \ddots \\ -I_p & & F_{pp,m} \end{bmatrix}$$

en

$$\{V_{q,i}^T b_{*,i}\} = \begin{bmatrix} V_{q,1}^T b_{*,1} \\ \vdots \\ V_{q,m}^T b_{*,m} \end{bmatrix},$$

$$\left\{c_{*,i}^T V_{q,i-1}\right\} = \left[\begin{array}{ccc} c_{*,1}^T V_{q,0} & \cdots & c_{*,m}^T V_{q,m-1} \end{array}\right], \text{ enz.}$$

$\{v_i\}$ with v_i een kolomvector is de vector bekomen wanneer we de m kolomvectoren v_i, $i = 1, \ldots, m$ boven elkaar plaatsen, $\{v_i^T\}$ met v_i^T een rijvector is de rijvector bekomen wanneer we de vectoren v_i^T, $i = 1, \ldots .m$ naast elkaar plaatsen. We kunnen de notatie veralgemenen naar sets van twee rij- of kolomvectoren. (44) kan worden herordend tot

$$\left[\begin{array}{ccc} F_{qq}^o & F_{qp}^o & \left\{V_{q,i}^T B_i\right\} \\ F_{pq}^o & F_{pp}^o & \left\{V_{p,i}^T B_i\right\} \\ \left\{C_i^T V_{q,i-1}\right\} & \left\{C_i^T V_{p,i-1}\right\} & D \end{array}\right] \left[\begin{array}{c} \{\Delta\bar{q}_{i-1}\} \\ \{\Delta\bar{p}_{i-1}\} \\ \Delta T \\ \Delta\gamma \end{array}\right] = - \left[\begin{array}{c} \left\{V_{q,i}^T r_i\right\} \\ \left\{V_{p,i}^T r_i\right\} \\ s \\ n \end{array}\right]. \tag{46}$$

Vermits $G_i V_{p,i-1} = V_{p,i} L_i$ verdwijnt de term F_{qp}^o. Ieder blok in (46) komt overeen met het blok in (12) op dezelfde positie. We kunnen de ganse redenering voor het algoritme voor enkelvoudig schieten uitbreiden tot het geval van meervoudig schieten wanneer we een goede techniek vinden om stelsels met de matrix F_{qq}^o en stelsels van de vorm

$$\left[\begin{array}{cc} F_{pp}^o & \left\{V_{p,i}^T \hat{B}_i\right\} \\ \left\{C_i^T V_{p,i-1}\right\} & \hat{D} \end{array}\right]$$

op te lossen. Eerst worden $\Delta\bar{q}_{i,r}$, $\Delta\bar{q}_{i,T}$ en $\Delta\bar{q}_{i,\gamma}$ berekend uit

$$F_{qq}^o \left[\begin{array}{ccc} \{\Delta\bar{q}_{i-1,r}\} & \{\Delta\bar{q}_{i-1,T}\} & \{\Delta\bar{q}_{i-1,\gamma}\} \end{array}\right] = - \left[\begin{array}{ccc} \left\{V_{q,i}^T r_i\right\} & \left\{V_{q,i}^T b_{T,i}\right\} & \left\{V_{q,i}^T b_{\gamma,i}\right\} \end{array}\right]. \tag{47}$$

Net als bij het algoritme voor enkelvoudig schieten zijn de termen $\{\Delta\bar{q}_{i-1,T}\}$ vaak verwaarloosbaar. Daarna kunnen $\Delta\bar{p}_i$'s, ΔT en $\Delta\gamma$ opgelost worden uit

$$\begin{array}{l} \left[\begin{array}{cc} F_{pp}^o & \left\{V_{p,i}^T B_i + F_{pq}^o \left[\begin{array}{cc} \Delta\bar{q}_{i-1,T} & \Delta\bar{q}_{i-1,\gamma} \end{array}\right]\right\} \\ \left\{C_i^T V_{p,i-1}\right\} & D + \sum_{i=1}^m C_i^T V_{q,i-1} \left[\begin{array}{cc} \Delta\bar{q}_{i-1,T} & \Delta\bar{q}_{i-1,\gamma} \end{array}\right] \end{array}\right] \left[\begin{array}{c} \{\Delta\bar{p}_{i-1}\} \\ \Delta T \\ \Delta\gamma \end{array}\right] = \\ \left[\begin{array}{c} \left\{V_{p,i}^T r_i + F_{pq}^o \Delta\bar{q}_{i-1,r}\right\} \\ s + \sum_{i=1}^m c_{s,i}^T V_{q,i-1} \Delta\bar{q}_{i-1,r} \\ n + \sum_{i=1}^m c_{n,i}^T V_{q,i-1} \Delta\bar{q}_{i-1,r} \end{array}\right] \end{array} \tag{48}$$

en uiteindelijk berekenen we

$$\Delta\bar{q}_i = \Delta\bar{q}_{i,r} + \Delta T\, \Delta\bar{q}_{i,T} + \Delta\gamma\, \Delta\bar{q}_{i,\gamma}.$$

Stelsels van de vorm $F_{qq}^o \{\Delta\bar{q}_{i-1,*}\} = \left\{V_{q,i}^T r_i, *\right\}$ worden opgelost met behulp van voorwaartse recursie en Picarditeratie of de GMRES methode. Hierbij wordt de Picard- of GMRES methode toegepast op het $(N-p) \times (N-p)$ stelsel dat bekomen wordt door condensatie van de matrix F_{qq}^o met voorwaartse recursie. In het proefschrift tonen we aan hoe de voorwaartse recursie geïntegreerd kan worden met het Picarditeratieschema of de GMRES techniek. Het stelsel (48) is een laagdimensionaal stelsel. Dit wordt ofwel rechtstreeks opglost met Gauss eliminatie of een kleinste-kwadratenmethode ofwel eerst verder gecondenseerd tot een $(p+2)\times(p+2)$-stelsel met behulp van achterwaartse recursie of een combinatie van voorwaartse en achterwaartse recursie. In het proefschrift breiden we ook het algoritme voor deelruimteïteratie met projectie en deflatie uit en combineren we dit algoritme met de periodieke Schur ontbinding zodanig dat we alle basis $V_{p,i}$ gelijktijdig kunnen berekenen, waarbij de totale rekentijd nauwelijks verschilt van die in het geval van enkelvoudig schieten. De praktische convergentieanalyse en de strategie voor het controleren van de convergentie kunnen ook eenvoudig uitgebreid worden.

4. De berekening van bifurcatiepunten

4.1. Periodeverdubbelingspunten

In het proefschrift bekijken we ook de mogelijke uitbreiding van de methode tot de berekening van keerpunten, torusbifurcatiepunten en periodeverdubbelingspunten met behulp van uitgebreide stelsels en enkelvoudig- of meervoudig schieten. Als voorbeeld beschouwen we een techniek voor de berekening van periodeverdubbelingspunten met enkelvoudig schieten.

In een periodeverdubbelingspunt is één van de Floquet vermenigvuldigers -1. Onze methode is gebaseerd op het uitgebreide stelsel

$$\begin{cases} r(x(0),T,\gamma) = \varphi(x(0),T,\gamma) - x(0) = 0, \\ s(x(0),T,\gamma) = 0, \\ (M(x(0),T,\gamma)+I)v = 0, \\ l^T v - 1 = 0. \end{cases} \tag{49}$$

Dit is een stelsel van $2N+2$ vergelijkingen in $2N+2$ onbekenden. In de oplossing van dit stelsel is v een eigenvector voor de Floquet vermenigvuldiger -1. We kunnen eenvoudig een startwaarde voor deze vector afleiden uit de initiële basis voor de Newton–Picard techniek. Het stelsel (49) wordt opgelost met behulp van de Newton–Raphson techniek. Het gelineariseerde stelsel is

$$\begin{bmatrix} M-I & b_T & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & d_{s,\gamma} \\ M_x v & M_T v & M+I & M_\gamma v \\ 0 & 0 & l^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta v \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r(x(0),T,\gamma) \\ s(x(0),T,\gamma) \\ (M(x(0),T,\gamma)+I)v \\ l^T v - 1 \end{bmatrix}, \tag{50}$$

waarbij

$$\begin{bmatrix} M-I & b_T & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & d_{s,\gamma} \\ M_x v & M_T v & M+I & M_\gamma v \\ 0 & 0 & l^T & 0 \end{bmatrix} = \frac{\partial(r,s,(M+I)v,l^T v)}{\partial(x(0),T,v,\gamma)}.$$

Veronderstelling 1 kan worden aangepast aan dit geval en basissen en projectoren kunnen worden opgebouwd zoals bij de techniek voor enkelvoudig schieten. De vectoren $\Delta x(0)$ en Δv worden opgesplitst in componenten volgens beide deelruimten. Het gelineariseerde stelsel (50) kan worden omgevormd tot

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & 0 & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & 0 & 0 & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & 0 & 0 & d_{s,\gamma} \\ V_q^T M_x v V_q & V_q^T M_x v V_p & V_q^T M_T v & V_q^T M V_q + I_q & 0 & V_q^T M_\gamma v \\ V_p^T M_x v V_q & V_p^T M_x v V_p & V_p^T M_T v & V_p^T M V_q & V_p^T M V_p + I_p & V_p^T M_\gamma v \\ 0 & 0 & 0 & l^T V_q & l^T V_p & 0 \end{bmatrix} \begin{bmatrix} \Delta \bar{x}_q \\ \Delta \bar{x}_p \\ \Delta T \\ \Delta \bar{v}_q \\ \Delta \bar{v}_p \\ \Delta \gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ V_q^T (M+I) v \\ V_p^T (M+I) v \\ l^T v - 1 \end{bmatrix} \quad (51)$$

Hierin verwaarloosden we de term $V_q^T b_T$. Dit is opnieuw een bovendriehoeksstelsel op de laatste kolom na. De oplossingstechniek gebruikt voor enkelvoudig schieten kan eenvoudig uitgebreid worden tot dit stelsel. Eerst worden twee stelsels met de matrix $V_q^T M V_q - I_q$ opgelost (voor de rechterleden $-V_q^T r$ en $-V_q^T b_\gamma$) met behulp van Picarditeratie en daarna worden twee stelsels met de matrix

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix}$$

opgelost. Dit zijn kleine reguliere stelsels. Hiervoor kan Gauss eliminatie met pivotering gebruikt worden. Daarna worden twee stelsels met de matrix $V_q^T M V_q + I_q$ opgelost. Dit kan opnieuw met een Picarditeratieschema gebeuren. Uiteindelijk moet nog een $(p+1) \times (p+1)$ stelsel opgelost worden. Merk op dat de matrix $M_x v$ nooit moet worden berekend. We hebben enkel enkele matrix–vectorproducten met deze matrix nodig.

In het proefschrift tonen we aan dat het uitgebreide stelsel (50) regulier is wanneer het periodeverdubbelingspunt niet gedegenereerd is en dat ieder van de deelstelsels die we in de Newton–Picard procedure moeten oplossen in dit geval ook regulier zijn indien $V_q^T b_T$ inderdaad verwaarloosbaar is en alle vorige deelstelsels voldoende nauwkeurig opgelost worden.

4.2. Andere bifurcatiepunten en meervoudig schieten

Merk op dat het matrix–vectorproduct Mv kan worden berekend als de oplossing van een beginwaardeprobleem voor de variationele vergelijkingen (zie (19)). Dit laat toe het uitgebreide stelsel (49) om te vormen tot het randwaardeprobleem

$$\begin{aligned} \frac{dx(t)}{dt} &= f(x(t), \gamma), \\ \frac{dv(t)}{dt} &= f_x(x(t), \gamma)\, v(t) \end{aligned}$$

met de randvoorwaarden

$$\begin{cases} x(T) = x(0), \\ v(T) = -v(0), \\ s(x(0), T, \gamma) = 0, \\ l^T v(0) = 1. \end{cases}$$

Dit laat een eenvoudige veralgemening naar meervoudig schieten toe. Een soortgelijk randwaardeprobleem wordt gebruikt in AUTO97 [31].

We bespreken in het proefschrift ook de aanpassingen die nodig zijn aan de techniek voor periodeverdubbelingspunten om ook torusbifurcatiepunten en keerpunten te kunnen berekenen. In beide gevallen baseren we ons op uitgebreide stelsels gelijkaardig aan die gebruikt in AUTO97.

Een torusbifurcatiepunt wordt gekenmerkt door een paar van complex toegevoegde Floquet vermenigvuldigers op de eenheidscirkel. Het is interessant om het gebruik van complexe getallen te vermijden. Wij stellen voor het uitgebreid randwaardeprobleem

$$\begin{aligned}\frac{dx(t)}{dt} &= f(x(t),\gamma),\\ \frac{dv(t)}{dt} &= f_x(x(t),\gamma)\,v(t),\\ \frac{dw(t)}{dt} &= f_x(x(t),\gamma)\,w(t)\end{aligned}$$

met randvoorwaarden

$$\begin{cases} x(T)=x(0),\\ v(T)-\cos(\theta)\,v(0)+\sin(\theta)\,w(0)=0,\\ w(T)-\sin(\theta)\,v(0)-\cos(\theta)\,w(0)=0,\\ s(x(t),T,\gamma)=0,\\ l_v^T v(0)=1,\\ l_w^T w(0)=0,\end{cases}$$

te gebruiken. Bij een techniek gebaseerd op enkelvoudig schieten wordt dit randwaardeprobleem herleid tot het niet-lineaire stelsel

$$\begin{cases} r(x(0),T,\gamma)=\varphi(x(0),T,\gamma)-x(0)=0,\\ s(x(0),T,\gamma)=0,\\ (M-\cos(\theta)I)\,v+\sin(\theta)w=0,\\ -\sin(\theta)v+(M-\cos(\theta)I)\,w=0,\\ l_v^T v-1=0,\\ l_w^T w=0.\end{cases}$$

In de oplosing heeft de systeemmatrix een paar complex toegevoegde eigenwaarden $\exp(\pm i\theta)$ en bijhorende eigenvectoren $v \pm iw$. Het afleiden van de Newton–Picard techniek verloopt op een analoge manier als voor periodeverdubbelingspunten. Er is echter een aangepast Picarditeratieschema nodig om stelsels met de matrix

$$\begin{bmatrix} V_q^T M V_q-\cos(\theta)I_q & \sin(\theta)I_q\\ -\sin(\theta)I_q & V_q^T M V_q-\cos(\theta)I_q\end{bmatrix}$$

op te lossen. We ontwikkelen een dergelijk schema in het proefschrift en tonen aan dat het schema convergeert indien $r_\sigma(V_q^T M V_q)<1$.

Voor de berekening van een keerpunt stellen we voor het uitgebreide randwaardeprobleem

$$\begin{aligned}
\frac{dx(t)}{dt} &= f(x(t),\gamma), \\
\frac{dv(t)}{dt} &= f_x(x(t),\gamma)\,v(t) + \frac{\alpha}{T} f(x(t),\gamma)
\end{aligned}$$

met randvoorwaarden

$$\begin{cases}
x(T) = x(0), \\
v(T) = v(0), \\
s(x(t),T,\gamma) = 0, \\
l_1^T v(0) = 0, \\
l_2^T v(0) = 1
\end{cases}$$

te gebruiken. In de oplossing heeft de systeemmatrix een dubbele eigenwaarde 1. In het algemene geval is $\alpha \neq 0$ en is v een hoofdvector voor deze eigenwaarde. De bijhorende eigenvector is $f(x(0),\gamma)$. Hierin is α een onbekende. Het eerste P-stelsel in de Newton–Picard methode voor dit uitgebreid stelsel is singulier en de oplossingsprocedure moet hieraan aangepast worden. We beschrijven in het proefschrift een techniek om dit te doen.

4.3. Alternatieven

Er zijn verschillende alternatieven voor onze aanpak.

Indirecte methoden. Onze Newton–Picard technieken kunnen de dominante Floquet vermenigvuldigers nauwkeurig berekenen met relatief weinig extra rekenwerk. Deze kunnen gebruikt worden om bifurcatiepunten te detecteren. We kunnen ook technieken voor het berekenen van nulpunten van scalaire functies gebruiken om een punt te berekenen waar een Floquet vermenigvuldiger de eenheidscirkel kruist. Het voordeel van deze aanpak is dat er geen aanpassingen aan de Newton–Picard code nodig zijn. De routine voor de berekening van het nulpunt (hier het punt waar een Floquet vermenigvuldiger 1 wordt in modulus) roept gewoon de bestaande routine voor de berekening van een periodieke oplossing aan.

Uitgebreide stelsels op basis van scalaire testfuncties. Sommige van deze technieken vereisen de volledige systeemmatrix en zijn daarom niet bruikbaar voor een hoogdimensionaal probleem. Een voorbeeld hiervan is de testfunctie

$$\det(M(x(0),T,\gamma) + I) = 0.$$

Een andere aanpak is het gebruik van gerande stelsels om een scalaire testfunctie te definiëren. Dergelijke methoden hebben dan weer het nadeel dat er voor de berekening van de afgeleiden stelsels opgelost moeten worden waarin de matrix M^T optreedt. Het is niet eenvoudig om in een code gebaseerd op enkelvoudig schieten matrix–vectorprodukten met deze matrix te berekenen. Dit is zeker onmogelijk wanneer de tijdsintegratiecode als een zwarte doos beschouwd wordt.

Uigebreide stelsels gebaseerd op de geprojecteerde systeemmatrix. Alle informatie over Floquet vermenigvuldigers in de buurt van de eenheidscirkel blijft bewaard in de geprojecteerde systeemmatrix $V_p^T M V_p$. Ook de voorwaarde dat bepaalde vectoren al dan niet in het bereik van de kolommen van de matrix M liggen kan overgezet worden naar het laagdimensionale systeem. Het is bijgevolg mogelijk om het uitgebreide stelsel op te bouwen op basis van de geprojecteerde systeemmatrix. Nochtans zijn we ervan overtuigd dan onze aanpak in de meeste gevallen voordelen biedt. Immers, testfuncties gebaseerd op de geprojecteerde systeemmatrix vereisen een nauwkeurige constructie van deze matrix. Dit betekent dat de basis in iedere Newton–Picardstap met een vrij grote nauwkeurigheid berekend moet worden, en op het einde zelfs met heel hoge nauwkeurigheid. In onze aanpak beperkt de nauwkeurigheid van de basis V_p de convergentiesnelheid. Wanneer het uigebreide stelsel echter op basis van de geprojecteerde systeemmatrix opgebouwd is, beperkt de nauwkeurigheid van V_p de nauwkeurigheid van de oplossing.

5. Besluit

In dit proefschrift ontwikkelden we een familie van robuuste en efficiënte technieken voor de berkening van takken van periodieke oplossingen van partiële differentiaalvergelijkingen met laagdimensionale dynamica gebaseerd op de ideeën van de recursieve projectiemethode [107] en de "gecondenseerde Newton–ondersteunende Picard" methode van Jaruasch en Mackens [56, 57, 58]. We construeerden methodes gebaseerd op enkelvoudigen meervoudig schieten. De nadruk werd gelegd op het gebruik van de technieken voor bifurcatieanalyse. Daarom hebben we de methode uitgebreid om bifurcatiepunten nauwkeurig te berekenen en hebben we robuuste strategieën gebruikt voor de berekening van de Floquet vermenigvuldigers.

Dit proefschrift levert verschillende belangrijke bijdragen:

- We hebben de recursieve projectiemethode uitgebreid tot de berekening en voortzetting van periodieke oplossingen met gebruik van enkelvoudig schieten. Hierbij namen we een origineel startpunt dat ons toeliet nieuwe, meer robuuste varianten te ontdekken. We ontwikkelden een krachtig kader voor de convergentieanalyse en -controle. Dit liet ons toe om de methode te combineren met allerhande iteratieve technieken voor het hoogdimensionale deelstelsel en directe technieken voor het laagdimensionale deelstelsel.

- We toonden het belang aan van de periodieke Schur ontbinding voor de nauwkeurige berekening van de Floquet vermenigvuldigers in collocatiecodes en codes gebaseerd op meervoudig schieten. Ook implementeerden we een variante van het QR algoritme voor de berekening van deze ontbinding.

- We veralgemeenden onze techniek tot meervoudig schieten. Voor zover we weten is dit de eerste uitbreiding in die richting.

- We ontwikkelden ook technieken voor de berekening van allerhande bifurcatiepunten. We bespraken ook mogelijke alternatieven om tot de conclusie te komen dat onze aanpak zeker redelijk is.

6. Suggesties voor verder onderzoek

Er is nog veel ruimte over voor verdere verfijningen en uitbreidingen van de techniek. Ook is er nog een lange weg af te leggen naar een code die echt bruikbaar is om grote modellen zoals ze zich voordoen in industriële problemen vlot te analyseren. We vermelden de volgende suggesties.

- De a priori en a posteriori convergentieanalyse behoeft verdere verfijning.
- De methode voor meervoudig schieten is nog niet compleet. In dit proefschrift toonden we enkel aan de Newton–Picard idee veralgemeend kan worden tot dit geval. Om de code compleet te maken is er echter ook nog een strategie voor de selectie van de roosterintervallen nodig. Het is ook zinvol om een alternatieve, geometrische fasevoorwaarde te onderzoeken. In dit geval leggen we voor ieder roosterpunt in de methode een aparte fasevoorwaarde op en zoeken we de periode als de som van de tijdsintevallen nodig om ieder van de roosterintervallen te overbruggen.
- De techniek voor bifurcatieanalyse kan ook nog verder uitgebreid worden. De methodes voor keerpunten en torusbifurcatiepunten en de uitbreidingen tot meervoudig schieten werden nog niet geïmplementeerd. De methode kan ook uitgebreid worden om takken van codimensie-1 bifurcatiepunten te berekenen. De alternatieve technieken die we vermeldden zouden interessant kunnen zijn voor meer complexe bifurcatiefenomenen.
- Onze code voor de berekening van de periodieke Schur ontbinding kan nog verder verfijnd worden en uitgebreid worden tot het geval dat optreedt in collocatiecodes.
- Enkele experimenten hebben uitgewezen dat het in een aantal gevallen interessant kan zijn om de staplengte in een voortzettingscode afhankelijk te maken van het verloop van testfuncties.
- De methode zou ook gecombineerd moeten worden met technieken voor roosterverfijning, zowel in de tijd als in de ruimte. Immers, we willen een nauwkeurige benadering van alle facetten van het continue systeem, en niet enkel van het grote stelsel GDVen bekomen met behulp van een vooraf bepaalde ruimtediscretisatie.
- Er zijn ongetwijfeld andere types van vergelijkingen met soortgelijke spectrale eigenschappen. Het is interessant om de techniek ook uit te breiden tot dergelijke vergelijkingen. Als voorbeeld pasten we de techniek aan voor stelsels van gewone differentiaalvergelijkingen met vertragingen.
- De code zou ook uitgebreid moeten worden met goede numerieke technieken voor de bifurcatieanalyse van evenwichtsoplossingen. Een efficiënte techniek ontwikkelen op basis van de Newton–Picard idee is niet triviaal. Eventueel kunnen andere technieken aangewezen zijn.
- We zijn ervan overtuigd dat de techniek uitgebreid kan worden tot het geval van Gauss-Legendre orthogonale collocatie. In het proefschrift geven we aan hoe dit

kan gebeuren. Het probleem is de efficiënte implementatie van Gauss-Legendre impliciete Runge-Kutta methodes voor partiële differentiaalvergelijkingen.

- Het is duidelijk dat de methode geïmplementeerd moet worden op parallelle computers om echt grote problemen te kunnen aanpakken. Merk op dat de berekening van takken van periodieke oplossingen van een probleem met d ruimtedimensies eigenlijk een $(d+2)$-dimensionaal probleem is: we zoeken functies van de d ruimtecoordinaten, de tijd en de parameterisatieparameter langsheen de curve. Het is duidelijk dat enkel parallellisatie in de ruimte geen techniek zal opleveren die scaleert naar massief parallelle computers en uitvoeringstijden in de ordegrootte van enkele uren oplevert. Het is interessant om technieken op ruimte-tijdsroosters te gebruiken om efficiënte parallelle codes te bekomen.

- Er is ook nood aan een goede gebruikersinterface om de parameters van de algoritmen in te stellen en na de uitvoering de resultaten te interpreteren.

Notations

List of symbols

This list gives the usual meaning of the symbols used throughout the text. In some cases, we locally use another meaning which is not included in this list.

α	accuracy goal in convergence analysis
γ	system parameter
Γ	system parameter vector
ε	convergence thresholds, etc.
ρ	threshold for the low-dimensional subspace of unstable or weakly stable modes in a Newton–Picard procedure
ϕ	initial condition for a delay equation: $\phi : [-\tau, 0] \mapsto \mathbb{R}^n$
φ	time integration operator
σ	singular value
τ	time delay
μ	– Floquet multipliers – eigenvalues of a Picard iteration scheme
ν	step number in an iterative process (Newton or Newton–Picard)
η	artificial parameter in a (pseudo)-arclength method
λ	eigenvalues at steady state solutions
ξ	used in convergence analysis
ω	– imaginary part of the eigenvalue crossing the imaginary axis in a Hopf bifurcation point – also locally used for some other purposes
Ω	hypersurface
ψ	polynomial
θ	weights in the phase condition and parameterizing equations
B	partial derivatives of the time integration operator φ with respect to T and γ
C	partial derivatives of the phase condition and parameterizing equation with respect to the state
D	partial derivatives of the phase condition and parameterizing equation with respect to T and γ
E	used for matrices that select rows or columns (e.g., basis accuracy)
F	submatrices in multiple shooting context

G	matrix $M(x(0), \Delta s\, T, \gamma)$ in multiple shooting methods
H	– Hessenberg matrix – also sometimes denotes an upper triangular matrix
I	unit matrix
J	– rotator – Jacobian matrix
M	partial derivative of the time integration operator $\varphi(x(0), T, \gamma)$ to its initial state $x(0)$. At the limit cycle, M is the monodromy matrix.
N	– dimension of the ODE system – dimension of the vector expressing the initial function of a DDE after discretization
P	projector on the “unstable” subspace
Q	projector on the “stable” subspace
R	– upper triangular matrix – Also used for the Schur decomposition of a matrix. R is then an upper block triangular matrix with 1×1 and 2×2 blocks on the diagonal.
S	low-dimensional projection of M
T	period of the limit cycle
U	– left singular vectors – also used in some of the algorithms for $V^T M V$
V	– basis vectors – right singular vectors
W	MV
X	generalized eigenvectors
Y	Schur vectors
Z	used in the error criterion for the bases
b	derivatives of the time integrator or Picard scheme with respect to time or parameter
c	derivatives of phase- and arclength condition with respect to the initial state
d	derivatives of phase- and arclength condition with respect to the period and parameter
k	– number of delays in a DDE – counter
l	– number of Picard steps or dimension of the Krylov space – vector in a normalization condition
m	number of multiple shooting intervals or number of time discretization points of the initial condition in a DDE problem
n	– safety factor – number of equations in a DDE system
p	dimension of the “unstable” subspace
q	dimension of ”stable” subspace
r	residual

s	shooting interval boundaries in multiple shooting code as a fraction of the period or time in a boundary value problem rescaled to $[0, 1]$
t	time variable
v	right eigenvector
w	left eigenvector
x	unknown function
y	x and some other variables

Other notations

$V[k{:}l]$	columns k to l of V
$\mathcal{K}_l(A, r)$	Krylov subspace of dimension l for the matrix A based on the vector r, i.e., $\mathrm{Span}\{r, Ar, \ldots, A^{l-1}r\}$
$R[k{:}l, m{:}n]$	subblock of the matrix R from row k to row l and column m to column n
$r_\sigma(A)$	spectral radius of the matrix A (i.e., the modulus of the largest eigenvalue)
$\sigma_{\max}(A)$	largest singular value of A
$T_\Omega(x)$	time to intersect the hypersurface Ω starting the time integration from x
$\psi_{*,l}$	polynomial of degree l
$<x,\ y>$	scalar product of x and y
e_i	i^{th} unit vector (a 1 at position i)
$\bar{p},\ \bar{q}$	the bar over a letter denotes the p or $q = N - p$-dimensional projection of a N-dimensional vector
$\mathrm{Rank}(A)$	rank of the matrix A
$\mathrm{Ker}(A)$	kernel of the transformation determined by the matrix A, i.e., $\{x \mid Ax = 0\}$
$\mathrm{Range}(A)$	range of the transformation determined by the matrix A, i.e., $\{y \mid \exists x : y = Ax\}$
$\mathrm{Span}(A)$	span of the basis determined by A and synonym of $\mathrm{Range}(A)$

Contents

Introduction

0.1 Motivation and aim of the thesis

In this thesis, we study algorithms for the computation and bifurcation analysis of periodic solutions of systems of parametrized nonlinear autonomous partial differential equations (PDEs). More specifically, we exploit the property that many PDEs have periodic orbits with only few unstable or weakly stable modes. This property is present in many important models, such as reaction–diffusion systems and the incompressible Navier–Stokes equations. Our aim is to develop a family of methods based on single and multiple shooting that exploits this property in order to efficiently compute branches of periodic solutions using a new iterative solution technique and continuation methods. We are also interested in gaining information on the stability of the solutions and in determining parameter values at which stability changes occur.

Theoretical and numerical bifurcation analysis and continuation methods have evolved rapidly since the famous paper of Keller [63] on the numerical solution of bifurcation and nonlinear eigenvalue problems appeared. Nowadays many packages for bifurcation analysis of steady-state and periodic solutions of small systems of ordinary differential equations (ODEs) or discrete maps are available, some of them for over ten years. We name just a few of them. AUTO [27, 28, 29, 31], which became available in 1986, is probably the most popular package and has evolved a lot since its first appearance. The current version (AUTO97) is a robust and very extensible program that is capable of analyzing steady-state, periodic and homoclinic solutions of ODEs. Because of its very generic problem definition, it can also be used to tackle other problems, such as the computation of heteroclinic connections [41]. Several graphical front-ends have been developed, e.g., [113] and the interface included with the AUTO97 distribution. Locbif [67, 68] exists in versions for studying maps, steady-state and periodic solutions of autonomous ODE systems and periodic solutions of non-autonomous ODEs. Besides a purely text-based version, there also exists a DOS-based version with a graphical front-end and a version integrated in the dstool package [5] developed at Cornell University. Locbif is capable of detecting and continuing codimension one and two bifurcations and even some codimension three bifurcations. CONTENT [72] is a new package, still under development, that intends to combine most features of AUTO and Locbif. It has a very user-friendly and powerful graphical user interface with a well-developed help facility and is portable to UNIX, MS Windows and Macintosh computer systems. Other well-known packages are Pitcon [97] and Bifpack [105]. All these packages are optimized for small problems. Although PDE systems can be transformed into large ODE systems using a space discretization scheme such as (pseudo-)spectral discretization, finite differences or finite elements, they can barely be analyzed with one of the above packages. The computational cost and memory requirements of those packages grow excessively as the system size increases. Furthermore, not all of these packages are able to exploit the sparsity resulting from finite difference or finite element discretizations, let alone other properties of the PDE.

There is little available software for continuation and bifurcation analysis of steady-state solutions of PDEs. A well-known package is PLTMG [8, 9]. This package is restricted to scalar elliptic PDEs and is only capable of computing branches of steady-state solutions and detecting and computing fold points and transcritical or pitchfork bifurcation points on these branches. It has also some facilities for branch switching. The package does not perform a complete stability analysis of the solutions and is not able to approximate or even just detect other types of bifurcation points such as Hopf points. In [86, 87], a technique is developed to compute branches of steady-state solutions of PDEs and to analyze the stability of these solutions by computing the rightmost eigenvalues. The problem of computing steady-state solutions of PDEs is rather well understood for many equations. There is a vast literature on computing such solutions using various techniques such as multigrid methods, Krylov-based techniques or domain decomposition techniques. Also a lot of work has been done on the computation of the rightmost stability-determining eigenvalues, see, e.g., [81] for an overview of such techniques.

Less work has been done on the computation of periodic solutions of PDEs. In recent years, studies of periodic solutions of PDEs have been presented on various conferences, but most of these are based on results from dynamical simulation or are limited to rather small problems with only few degrees of freedom in the discretization.

There are various techniques that intend to exploit the property that dissipative systems usually have a low-dimensional attractor. The inertial manifold theory states that the attractor can often be imbedded in a low-dimensional and smooth manifold that attracts all other orbits with exponential speed. The long-term dynamics of the model can then be described by restricting the PDE system to the inertial manifold. This idea is exploited by the (numerical) approximate inertial manifold methods [108]. The older papers use the nonlinear Galerkin method based on (pseudo-)spectral discretizations (e.g., [12]), but currently, methods based on other discretization techniques are also being investigated, e.g., the method of incremental unknowns based on finite differences [18]. The gain obtained with these methods is often limited. If accurate results are desired, the number of unknowns can usually only be reduced by a factor of two per space dimension, leading to little or no reduction in computing time [42]. Using fewer modes can lead to inaccurate results, see, e.g., [12]. Another family of techniques is based on the statistical analysis of the time behaviour of the system on the attractor by means of the Karhunen-Loève decomposition [7, 10, 23, 46] (also known as the proper orthogonal decomposition or the method of empirical eigenfunctions). Here, the most important modes in the system are determined from the covariance matrix or operator and the system is projected on these modes using a Galerkin projection. The decomposition is based on the analysis of the attractor at a specific parameter value. Extrapolation of the results to other parameter values can lead to dubious results. In [23], the authors report both a fairly successful case and a rather unsuccessful case of reconstructing part of the bifurcation diagram by analyzing the low-dimensional model at one parameter value.

Both of the above techniques have some serious disadvantages. Besides the fact that they can return very inaccurate results or otherwise do not offer a big gain in computing time, they require changes at the space discretization level, requiring a complete rewrite of the codes. This is an expensive task. A lot of time and money has been spent already on the development of simulation codes for PDEs. Clearly, there is a need for techniques that produce accurate and reliable results at a reasonable computing cost without requiring complete rewrites of existing codes. The method which we present in this thesis can be used with many simulation codes and requires only few changes to the codes.

The aim of this thesis is to show that the idea of combining cheap iterative methods (such as Picard iteration) and Newton's method with a direct solver (which is more robust but also more expensive) in a clever way can lead to very efficient algorithms for computing and analyzing periodic solutions. This idea was first proposed for steady-state solutions of large symmetric problems by Jarausch and Mackens [56, 57, 58]—they called their method the "*condensed Newton – supported Picard*" method—and later extended to non-symmetric problems by Shroff and Keller [107] in their "*recursive projection method*". We will first present some techniques based on single shooting. Here we basically try to combine straightforward time integration with Newton-based single shooting. There are some differences in the way we derive our method compared to [107], which will lead to a family of techniques, one of which is a straightforward extension of the recursive projection method of Shroff and Keller. Some of our other variants show much better performance on nonnormal problems. The method is particularly interesting to compute branches of periodic solutions and also returns stability information at little or no extra cost. Then, a new, more algebraic view on the method is developed, leading to more robust methods. We will also show how the method can be generalized to multiple shooting and can be used to compute various bifurcation points accurately. To further demonstrate the power of the ideas behind the method, we will present some results on using the method for the computation of periodic solutions of ordinary differential equations with a finite number of discrete delays (delay differential equations or DDEs) and present some ideas on how we feel the method can be extended to collocation.

0.2 Overview of the thesis

Let us now briefly present an overview of this thesis.

Chapter 1. Preliminaries. We introduce the problem which we study in this thesis. First, some ideas used throughout this thesis will be introduced. Then we briefly discuss the classical algorithms that are used in various packages for small problems, and we indicate why they do not scale well to larger problems. This discussion is followed by a brief review of continuation methods and techniques to compute bifurcation points.

Chapter 2. Single shooting: the basic ideas. We introduce our method and develop a first view of the method as a Newton-like method with an approximated Jacobian matrix. We will first derive a method for computing a periodic solution at a predetermined parameter value and show how we can interpret this method as an orthogonal transformation of the state variables followed by a certain approximation of the Jacobian matrix such that the product of the inverse of that matrix with a given vector can be computed easily. This allows us to prove the asymptotic convergence of the method (under some restrictions). Then an extension towards pseudo-arclength continuation is developed. We also show how the orthogonal transformation can be computed easily and end the chapter with a discussion on several variants of the derived method and on the computational complexity and with some test results demonstrating the properties of these variants.

Chapter 3. Single shooting: advanced methods. We develop a more algebraic view of the method: the high-dimensional and "hard" linear problem (resulting from the linearization of the single shooting system) is split in a low-dimensional but "hard" subsystem solved by direct methods and one or more high-dimensional but "easy" subsystems solved by appropriate iterative techniques. This view is supported by an analysis of the results of a single iteration step which allows us to better understand why and how the method works and why some variants are far superior to others. In fact, our two variants (one for fixed parameter computations and one for pseudo-arclength continuation) that differ most from the ones presented in [107], are clearly superior for problems with a nonnormal Jacobian matrix. These two variants are further developed. Their robustness is greatly enhanced by exploiting the results of the convergence analysis. The chapter ends with some results that demonstrate the increased robustness and a discussion on the use of the method to compute periodic solutions of delay differential equations.

Chapter 4. Steady-state solutions. We first rewrite our methods for the computation of steady-state solutions. We point out the basic difficulties in obtaining an efficient Newton–Picard scheme. We also discuss the differences with the recursive projection method [107] and with work done by K. Burrage, J. Erhel, B. Pohl and A. Williams [13, 14, 36, 118] on this type of methods, and also briefly compare with the work of Jarausch and Mackens [56, 57, 58] and later work of Jarausch [54, 55].

Chapter 5. The periodic Schur decomposition. The main purpose of this chapter is to introduce this powerful tool to the reader, since this little known decomposition plays an important role in our multiple shooting algorithm. We discuss the periodic Schur decomposition for a product $G = G_m \cdots G_1$ of m matrices. This decomposition is a generalization of the Schur decomposition. It gives the Schur decomposition of $G_m \cdots G_1$ and all cyclic permutations of this product. This decomposition and a corresponding algorithm was first presented in [11]. It offers a very stable way to compute the Schur decomposition of the product G, even if there are very large differences in the magnitude of the eigenvalues of G—much larger than $1/\epsilon_{mach}$ with ϵ_{mach} the machine precision for the floating point data type used. We will briefly discuss the algorithm, give some information about our implementation and show that this method is indeed capable of computing an eigenvalue spectrum characterized by extreme differences in the magnitude of the eigenvalues.

Chapter 6. Multiple shooting methods. Starting from the algebraic view of our single shooting method, we generalize the method to multiple shooting by combining the ideas behind our single shooting based methods with the solution techniques presented in [4]. The chapter starts with a discussion on solving the linearized multiple shooting system since the low-dimensional subsystem in our method has precisely the same form, and continues with a detailed derivation of the Newton–Picard multiple shooting method. To compute the bases needed for the transformation of the variables, we derive a new variant of the orthogonal subspace iteration method based on the periodic Schur decomposition.

Chapter 7. Computing bifurcation points. We present a single-shooting based method using the Newton–Picard idea to compute period doubling points based on using a large but structured extended system that introduces the eigenvector for the Floquet multiplier equal to -1. We also show how this algorithm can be adapted for the computation of torus bifurcation points and fold points and discuss why approaches based on large extended systems that introduce eigenvectors are often superior to methods that add only one or a few scalar equations based on the low-dimensional subsystem.

Chapter 8. Conclusions. We first state our conclusions and outline our own contributions. Next, we introduce some topics for future research not yet discussed in one of the previous chapters. We first discuss some ideas on the extension towards orthogonal collocation and finite difference methods. Then, we elaborate on the software aspects and on the requirements for an efficient and user-friendly interface.

Chapter 1

Preliminaries

1.1 Periodic solutions of PDEs

1.1.1 From PDE to ODE

This thesis is concerned with the efficient computation of periodic solutions of certain parabolic partial differential equations. We will consider the PDE system after space discretization with an appropriate space discretization scheme such as, e.g., finite differences, finite elements or (pseudo-)spectral elements. In most cases, the ODE system resulting from this discretization can be written as

$$\frac{dx}{dt} = f(x, \Gamma),\ f : \mathbb{R}^N \times \mathbb{R}^{n_\gamma} \mapsto \mathbb{R}^N \tag{1.1}$$

where N is "large" and Γ represents the parameters of the system. For some PDEs, a straightforward application of the discretization method results in a system

$$B\frac{dx}{dt} = f(x, \Gamma) \tag{1.2}$$

with B a singular $N \times N$-matrix. The singularity of B comes from restrictions on the vector x (e.g., the second equation in (1.3)). In this case, not every possible x represents a valid state and we must assure that only valid states x are obtained. This occurs for instance for the discretization of the incompressible Navier-Stokes equations [23, 52]

$$\begin{cases} \dfrac{\partial v}{\partial t} + (v \cdot \nabla)v = -\dfrac{1}{\rho}\nabla p + \nu\nabla^2 v \\ \nabla v = 0. \end{cases} \tag{1.3}$$

Note that there also exists a divergence free form of these equations. In the latter case, we obtain a discretized system of type (1.1). This approach is used in [7].

In this thesis, we assume that a system of the type (1.1) is obtained after the space discretization. Generalizing our methods to (1.2) may require substantial changes. We also assume there is only one parameter γ which we wish to vary during our computations, so $n_\gamma = 1$. Therefore, we will omit the parameter vector Γ from our notations and continue with

$$\frac{dx}{dt} = f(x, \gamma),\ f : \mathbb{R}^N \times \mathbb{R} \mapsto \mathbb{R}^N. \tag{1.4}$$

We will assume that f is C^2-continuous in x and γ throughout the region of interest.

1.1.2 Periodic solutions

We will mainly be concerned with periodic solutions of (1.4). Let

$$\varphi(x(0), T, \gamma),\ \varphi : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^N$$

denote the result of a time integration of (1.4) starting from $x(0)$ over a time interval of length T. A *periodic solution* or *limit cycle* is a closed non-constant trajectory (i.e., not an equilibrium point). In this case,

$$\varphi(x(0), t, \gamma) = \varphi(x(0), t + T, \gamma) \quad \forall t \in [0, T)$$

for some non-zero T. The smallest positive value of T for which this condition holds is called the *period* of the periodic orbit. For the autonomous ODE system (1.4), an orbit is completely determined by one point of the orbit, so it is sufficient to compute one point $x(0)$ on the limit cycle. Hence a limit cycle can be found as the solution of the boundary value problem (1.4) with the periodic boundary conditions

$$x(T) = x(0). \tag{1.5}$$

The solution of this problem is not unique. Indeed, since we are dealing with autonomous systems, every point on the limit cycle can be used to determine the limit cycle (or "be the point at time 0"). Therefore we need to add an equation $s(x(0), T, \gamma) = 0$ to the system, called a *phase condition.* At a predetermined value of the parameter γ, the initial condition $x(0)$ and the period T are a solution of the nonlinear problem

$$\begin{cases} \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) = 0. \end{cases} \tag{1.6}$$

A suitable phase condition should intersect the limit cycle transversally. Then (1.6) usually has isolated solutions and a nonsingular Jacobian matrix.

For most of this thesis, we will be interested not just in computing a single periodic solution at a given parameter value γ, but also in computing branches of periodic solutions. In this case, the parameter γ is allowed to vary and another equation, the *parametrizing equation*, is added to (1.6). The unknowns $x(0)$, T and γ can be computed from the nonlinear system

$$\begin{cases} \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) = 0, \\ n(x(0), T, \gamma) = 0. \end{cases} \tag{1.7}$$

We will briefly elaborate on the choice of the parametrizing equation in section 1.5.

An important tool in the study of the behaviour of the system in the neighbourhood of a periodic solution is the *monodromy matrix.* Let $(x(0)^*, T^*, \gamma^*)$ denote a solution of (1.7). The monodromy matrix M^* is then defined as

$$M^* = \left. \frac{\partial \varphi(x(0), T, \gamma)}{\partial x(0)} \right|_{(x(0)^*, T^*, \gamma^*)}, \tag{1.8}$$

the partial derivative of the time integration operator with respect to the initial state. The matrix-vector product M^*v expresses how (up to first order terms) a small perturbation at $x(0)^*$ in the direction v will evolve if we integrate round the limit cycle once. The monodromy matrix thus contains information on the stability of the limit cycle. The eigenvalues of the monodromy matrix are called the *Floquet multipliers* or *characteristic multipliers.* We will denote the Floquet multipliers using the symbol μ.

Lemma 1.1 *Suppose $(x(0)^*, T^*, \gamma^*)$ determines a periodic solution of (1.4). Then, M^* as defined in (1.8) has an eigenvalue 1 and a corresponding eigenvector $f(x(0)^*, \gamma^*)$, i.e., the tangent vector to the limit cycle in $x(0)$.*

Proof. The proof of this lemma can be found in [106], p. 255. □

Note that

$$\left.\frac{\partial \varphi(x(0), T, \gamma)}{\partial T}\right|_{(x(0),T,\gamma)} = f(\varphi(x(0), T, \gamma), \gamma) \tag{1.9}$$

(which is true for any $(x(0), T, \gamma)$) and in particular at the limit cycle $(x(0)^*, T^*, \gamma^*)$

$$\left.\frac{\partial \varphi(x(0), T, \gamma)}{\partial T}\right|_{(x(0)^*,T^*,\gamma^*)} = f(x(0)^*, \gamma^*).$$

The monodromy matrix and its eigenvectors depend on the choice of the point $x(0)^*$ along the limit cycle, but the Floquet multipliers and the Jordan structure do not depend on the point $x(0)^*$ ([39] p. 53).

Remark that M^* is a non-singular matrix ([39], p. 53). In fact, one can even prove the more general lemma

Lemma 1.2 *Suppose $\varphi(x(0), T, \gamma)$ exists and is sufficiently differentiable as a function of $x(0)$ in the region of interest. Then*

$$M(x(0), T, \gamma) = \left.\frac{\partial \varphi(x(0), T, \gamma)}{\partial x(0)}\right|_{(x(0),T,\gamma)}$$

is nonsingular.

Proof. M can be computed from the variational problem

$$\frac{\partial M(x(0), t, \gamma)}{\partial t} = \left.\frac{\partial f(x, \gamma)}{\partial x}\right|_{\varphi(x(0),t,\gamma)} M(x(0), t, \gamma) \text{ with } M(x(0), 0, \gamma) = I,$$

see, e.g., [106]. The nonsingularity of M then follows from Liouville's formula, [39] p. 12. □

This property will be used in the construction of our multiple shooting based method.

The matrix $M(x(0), T, \gamma)$ in general and the monodromy matrix in particular are full matrices. If (1.4) is the result of a space discretization of a PDE, the Jacobian matrix of $f(x, \gamma)$ often has a sparse structure. This sparsity does not carry over to M and the monodromy matrix.

1.2 Stability and bifurcations of periodic solutions

In the literature, the problem of studying bifurcations of periodic solutions is usually reduced to that of bifurcations of maps by defining a *Poincaré section* and studying the *Poincaré map* or *return map* [48, 71]. In this section, $x(0)$ always denotes a point along the limit cycle and the matrix $M(x(0), T, \gamma)$ is always computed at the limit cycle. Therefore we will omit the superscript $*$ from our notations in this section. Let $b = f(x(0), \gamma)$ be the tangent vector to the limit cycle in $x(0)$ at a given parameter value γ and

$$\Omega = \{x \mid s(x) = 0\} \tag{1.10}$$

an $(N-1)$-dimensional hypersurface that cuts the limit cycle transversally in $x(0)$, i.e.,

$$\begin{aligned} &\text{a) } s(x(0)) = 0, \\ &\text{b) } \frac{\partial s}{\partial x} b = <s, b> \neq 0. \end{aligned} \tag{1.11}$$

The Poincaré map is then defined as

$$P_\Omega(x) \ : \ \Omega \mapsto \Omega \ : \ P_\Omega(x) = \varphi(x, T_\Omega(x), \gamma) \tag{1.12}$$

where $T_\Omega(x)$ is the time needed to intersect the Poincaré section again after one round around the limit cycle. $P_\Omega(x)$ is basically an $(N-1)$-dimensional map. The limit cycle is a fixed point of the Poincaré map and the study of the stability of the limit cycle reduces to the study of the stability of the fixed point $x(0)$ of the map $P_\Omega(x)$. Hence the stability is determined by the eigenvalues of the linearization P_Ω of $P_\Omega(x)$ around $x(0)$. P_Ω is an $(N-1)$-dimensional matrix if a local coordinate system on Ω is used.

Lemma 1.3 *(lemma 1.3 in [71]) Suppose Ω (1.10) satisfies the conditions (1.11). The eigenvalues of P_Ω are independent of the point $x(0)$, the cross-section Ω and the local coordinate system on Ω.*

The stability of the limit cycle is given by the following lemma:

Lemma 1.4 *The fixed point $x(0)$ of $P_\Omega(x)$ is asymptotically stable if all eigenvalues of P_Ω lie within the unit circle. The limit cycle is unstable if at least one of the eigenvalues of P_Ω lies outside the unit circle.*

A similar theorem was first proved by Lyapunov in 1892 for equilibrium points of a vector field.

The relation between the eigenvalues of the $(N-1)$-dimensional linearized Poincaré map P_Ω and the monodromy matrix $M(x(0), T, \gamma)$ is given by the following theorem [39]:

Theorem 1.5 *The eigenvalues of the linearization P_Ω of the Poincaré map $P_\Omega(x)$ attached to the periodic solution of a vector field, where the Poincaré map is considered as an $(N-1)$-dimensional linear operator, are equal to those of the monodromy matrix $M(x(0), T, \gamma)$ for that periodic solution provided that the number 1 is deleted once from the set of eigenvalues of $M(x(0), T, \gamma)$.*

Proof. See [39] p. 223. □

From lemma 1.4 and theorem 1.5 one can easily derive the following corollary.

Corollary 1.6 *Let $(x(0), T, \gamma)$ determine a periodic solution of (1.4). Let M be the monodromy matrix for this orbit. Let $\mu_1 = 1$, μ_2, $\ldots$, μ_N be the N eigenvalues of M. Then:*

(a) The periodic orbit determined by $(x(0), T, \gamma)$ is asymptotically stable if $|\mu_i| < 1$, $i = 2, \ldots, N$.

(b) The periodic orbit determined by $(x(0), T, \gamma)$ is unstable if $|\mu_i| > 1$ for at least one value of i.

From theorem 1.5 and lemma 1.2 also follows that P_Ω is nonsingular. Theorem 1.5 can be further extended to prove a relationship between the Jordan structure of P_Ω and $M - bb^T/b^Tb$ with $b = f(x(0), \gamma)$.

Theorem 1.7 *Let $x(0)$ be an arbitrary point on a limit cycle of $\frac{dx}{dt} = f(x, \gamma)$ with period T at parameter value γ. Let $b = f(x(0), \gamma)$ be the tangent vector to the limit cycle in $x(0)$ and*

$$M = \frac{\partial \varphi(x(0), T, \gamma)}{\partial x(0)}$$

the monodromy matrix of the limit cycle in $x(0)$. Let $\Omega = \{x \mid s(x) = 0\}$ be a hypersurface such that $s(x(0)) = 0$ and $c^T b \neq 0$ with

$$c^T = \frac{\partial s(x(0), T, \gamma)}{\partial x(0)}.$$

Let $P_\Omega(x)$ be the Poincaré map defined using the cross-section Ω.

If Λ is the Jordan structure of the linearization of $P_\Omega(x)$ around $x(0)$ seen as an $(N-1)$-dimensional operator, then $\begin{bmatrix} 0 & 0_{1\times(N-1)} \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix}$ is the Jordan structure of $M - \frac{bb^T}{b^Tb}$.

Proof. We will first choose another coordinate system locally around $x(0)$ to study the monodromy matrix and the linearization of $P_\Omega(x)$. Let $\bar{Y} \in \mathbb{R}^{N\times(N-1)}$ be a basis for $\tilde{\Omega} = \{x \mid c^T(x - x(0)) = 0\}$ (the linearization of the hypersurface Ω around $x(0)$). Then $Y = \begin{bmatrix} b & \bar{Y} \end{bmatrix}$ is a nonsingular matrix since $c^T b \neq 0$. The coordinate system is transformed according to $x = Yy + x(0)$. From now on, we will use the superscript $^{(y)}$ to indicate that a matrix or vector should be interpreted in the y-coordinate system. In the y-coordinate system, the monodromy matrix is given by

$$M^{(y)} = Y^{-1} M Y$$

and

$$b^{(y)} = Y^{-1} b = e_1 \text{ (the first unit vector).} \tag{1.13}$$

The Poincaré map is given by

$$y \leftarrow \varphi^{(y)}(y, T(y), \gamma) \tag{1.14}$$

where

$$\varphi^{(y)}(0, T(0), \gamma) = 0.$$

Computing the derivative of (1.14) with respect to y results in

$$P^{(y)} := \frac{\partial \varphi^{(y)}(y, T(y), \gamma)}{\partial y} = M^{(y)} + \frac{\partial \varphi^{(y)}}{\partial T}\frac{\partial T}{\partial y} = M^{(y)} + e_1 \frac{\partial T}{\partial y}. \tag{1.15}$$

Let

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} := \left(\frac{\partial T}{\partial y}\right)^T$$

with $f_1 \in \mathbb{R}$ and $f_2 \in \mathbb{R}^{N-1}$. From the construction of the Poincaré map and the choice of our coordinate system, we know

- $P^{(y)}e_1 = 0$, i.e., the first column of $P^{(y)}$ is a zero column. (An infinitesimal perturbation in the direction of the tangent vector results in the same intersection with Ω.)
- The first row of $P^{(y)}$ is a zero row. (Since the image of an arbitrary point always lies in Ω, the linearized map must map every perturbation in $\tilde{\Omega}$.)
- Since $M^{(y)}e_1 = e_1$, we have $P^{(y)}e_1 = M^{(y)}e_1 + e_1 f_1 = e_1 + e_1 f_1$. Since $P^{(y)}e_1 = 0$, we get $f_1 = -1$.

Thus

$$P^{(y)} = \begin{bmatrix} 0 & 0_{1\times(N-1)} \\ 0_{(N-1)\times 1} & \bar{P} \end{bmatrix} \text{ and } M^{(y)} = P^{(y)} - e_1 f^T = \begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \bar{P} \end{bmatrix} \tag{1.16}$$

where $\bar{P}$ is the linearized Poincaré operator as an $(N-1)$-dimensional operator in the local coordinate system determined by Y.

Let

$$\bar{P} = \bar{Z}\bar{\Lambda}\bar{Z}^{-1} \tag{1.17}$$

be the Jordan decomposition of $\bar{P}$. From lemma 1.2 and theorem 1.5 follows that $\bar{P}$ and thus $\bar{\Lambda}$ are always nonsingular. Note also that the Jordan structure of $\bar{P}$ is independent of the choice of the vectors $\bar{Y}$. Switching to another basis corresponds to a similarity transform of $\bar{P}$. Let

$$Z = \begin{bmatrix} 1 & 0_{1\times(N-1)} \\ 0_{(N-1)\times 1} & \bar{Z} \end{bmatrix}. \tag{1.18}$$

From (1.16) and (1.17) it follows that

$$M^{(y)} = Y^{-1}MY = Z \begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix} Z^{-1}$$

or

$$M = (YZ) \begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix} (YZ)^{-1}$$

and

$$\begin{aligned} M - \frac{bb^T}{b^Tb} &= (YZ)\begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix}(YZ)^{-1} - \frac{bb^T}{b^Tb} \\ &= (YZ)\left(\begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix} - \frac{1}{b^Tb}Z^{-1}Y^{-1}bb^TYZ\right)(YZ)^{-1} \\ &= (YZ)\left(\begin{bmatrix} 1 & -f_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix} - \begin{bmatrix} \frac{b^TYZ}{b^Tb} \\ 0_{(N-1)\times N} \end{bmatrix}\right)(YZ)^{-1}. \end{aligned}$$

The last step holds since $Z^{-1}Y^{-1}b = Z^{-1}e_1 = e_1$ (see (1.13) and (1.18)). Since $b^TYZe_1 = b^Tb$, $\frac{b^TYZ}{b^Tb} = \begin{bmatrix} 1 & g_2^T \end{bmatrix}$ and

$$M - \frac{bb^T}{b^Tb} = \begin{bmatrix} 0 & h_2^T \\ 0_{(N-1)\times 1} & \Lambda \end{bmatrix}$$

where $h_2^T = -g_2^T - f_2^T$. Let

$$V = \begin{bmatrix} 1 & h_2^T\bar{\Lambda}^{-2} \\ 0_{(N-1)\times 1} & \bar{\Lambda}^{-1} \end{bmatrix},$$

then

$$V^{-1} = \begin{bmatrix} 1 & -h_2^T\bar{\Lambda}^{-1} \\ 0_{(N-1)\times 1} & \bar{\Lambda} \end{bmatrix}$$

and some easy computations show that

$$M - \frac{bb^T}{b^Tb} = (YZV)\begin{bmatrix} 0 & 0_{1\times(N-1)} \\ 0_{(N-1)\times 1} & \bar{\Lambda} \end{bmatrix}(YZV)^{-1}$$

which is the Jordan decomposition of $M - \frac{bb^T}{b^Tb}$. □

Theorem 1.7 is interesting since it allows us to study the stability of a limit cycle by computing the eigenvalues of its monodromy matrix. All conditions on the eigenstructure of the linearized map in bifurcation theorems of maps can be restated as conditions on the eigenstructure of $M - bb^T/b^Tb$. This matrix can usually be derived easily from the computations as we shall see in section 1.4.

Let us now consider how the stability of the limit cycle can change as system parameters are varied. At certain critical values of the parameters, the topological structure of the phase portrait may change. This phenomenon is called a *bifurcation.*

Definition 1.8 *([71] definition 2.10) The appearance of a topologically nonequivalent phase portrait under variation of parameters is called a bifurcation.*

One can plot equilibrium points, periodic solutions, etc. and indicate stability changes in diagrams by plotting a characteristic quantity of the equilibrium point or periodic solution versus the parameter. Such diagrams are called *bifurcation diagrams.*

Bifurcations can be *local*, i.e., the changes occur locally in a small region of the phase plane, or *global* if global changes occur in the phase portrait. Bifurcations of limit cycles that correspond to local bifurcations of the Poincaré map are often considered to be local bifurcations despite the fact that they cause global changes in the phase portrait. In this thesis, we will only consider local bifurcations that are generic as one parameter in the system is varied.

Local bifurcations of limit cycles occur if the stability of the limit cycle changes, i.e., if one or more of the Floquet multipliers μ_i cross the unit circle. There are three generic scenarios: $\mu_i = 1$ (a second Floquet multiplier equal to 1), $\mu_i = -1$ or $\mu_i = e^{\pm i\theta}$, $\theta \in]0, \pi[$. We will now study each of these scenarios in a little more detail.

$\mu = 1$: **fold point, pitchfork or transcritical bifurcation.** The Poincaré map has a simple eigenvalue 1 and the monodromy matrix has a double eigenvalue 1 with a geometric multiplicity of either one or two (i.e., a Jordan chain of two vectors or two linearly independent eigenvectors). This scenario usually corresponds to a *fold point*, sometimes also called a *turning point* or *limit point*. For smaller values of the parameter γ, there are locally two solutions in the bifurcation diagram, for larger values none or vice-versa. When moving along the branch, the stability of the periodic solution changes in the fold point since one Floquet multiplier moves outside of or inside the unit circle. Sometimes pitchfork and transcritical bifurcations may also occur. These bifurcations are not generic in a one-parameter family of solutions, but may occur in some systems, e.g., if the model has certain symmetries. At a *pitchfork bifurcation point*, branches intersect such that at one side in the parameter direction, there is locally only one solution and at the other side there are three solutions. This bifurcation often occurs when the symmetry of a solution is broken. When moving along the two-sided branch, a Floquet multiplier crosses the unit circle through 1 at the bifurcation point and a stability change occurs. Along the one-sided branch, one of the Floquet multipliers touches the unit circle at 1, but does not cross it and no stability change occurs. At a *transcritical bifurcation point*, branches intersect such that there are locally two solutions on both sides of the bifurcation point in the parameter direction. When moving along one of the branches, a Floquet multiplier crosses the unit circle through 1. At the transcritical bifurcation, a stability exchange between the two intersecting branches occurs. These cases are depicted in Figure 1.1. In other nongeneric cases, more complex phenomena such as cusp points may also occur. A double Floquet multiplier 1 of the monodromy matrix also occurs when a branch of limit cycles ends or starts in another bifurcation point. This occurs at a steady-state Hopf bifurcation point, where we do not really have a limit cycle anymore, but an equilibrium point, and at a period doubling bifurcation point where the branch with double period ends or starts.

The distinction between a fold point on the one hand and a transcritical or pitchfork bifurcation on the other hand can be made using

Theorem 1.9 *([66] theorem 3.9) Let $(x(0), T, \gamma)$ be a solution of*

$$\begin{cases} \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) = 0, \end{cases}$$

where $s(x(0), T, \gamma) = 0$ is a nondegenerate phase condition, i.e.,

$$\frac{\partial s(x(0), T, \gamma)}{\partial x(0)} \frac{\partial \varphi(x(0), T, \gamma)}{\partial T} \neq 0.$$

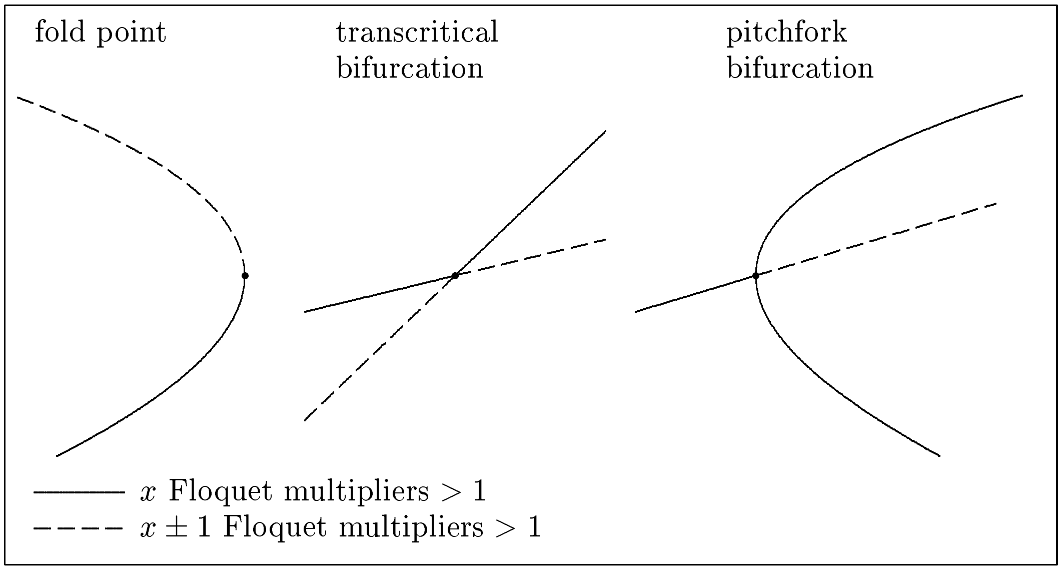

fold point
transcritical bifurcation
pitchfork bifurcation
x Floquet multipliers > 1
$x \pm 1$ Floquet multipliers > 1

Figure 1.1: Fold point, transcritical- and pitchfork bifurcation points. Each point in the figure actually denotes a limit cycle.

Then $(x(0), T, \gamma)$ is a fold point if and only if

$$a) \left\{ \begin{array}{l} \dim \text{ Null}(M - I) = 2, \\ b_T := \frac{\partial \varphi(x(0),T,\gamma)}{\partial T} \notin \text{Range}(M - I), \\ b_\gamma := \frac{\partial \varphi(x(0),T,\gamma)}{\partial \gamma} \notin \text{Range}\left(\left[\begin{array}{cc} M - I & b_T \end{array} \right] \right), \end{array} \right.$$

or

$$b) \left\{ \begin{array}{l} \dim \text{ Null}(M - I) = 1, \\ b_T \in \text{Range}(M - I), \\ b_\gamma \notin \text{Range}\left(\left[\begin{array}{cc} M - I & b_T \end{array} \right] \right). \end{array} \right.$$

$b_T \in \text{Range}(M - I)$ implies that there exists a generalized eigenvector v satisfying $Mv = v + b_T$. Note that theorem 1.9 does not give nondegeneracy conditions for the fold point. Such conditions, which state that the fold point is a quadratic fold point and not a higher-codimension point, require higher-order derivatives. Conditions for a simple bifurcation point (either a simple transcritical or a simple pitchfork bifurcation point) are given by the following theorem:

Theorem 1.10 *([66] corollary 4.23) Let $(x(0), T, \gamma)$ be a solution of*

$$\left\{ \begin{array}{l} \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) = 0, \end{array} \right.$$

where $s(x(0), T, \gamma) = 0$ is a nondegenerate phase condition, i.e.,

$$\frac{\partial s(x(0), T, \gamma)}{\partial x(0)} \frac{\partial \varphi(x(0), T, \gamma)}{\partial x(0)} \neq 0.$$

Then $(x(0), T, \gamma)$ is a simple bifurcation point if and only if

$$a)\ \begin{cases} \dim\ \mathrm{Null}(M-I)=2, \\ b_T \notin \mathrm{Range}(M-I), \\ b_\gamma \in \mathrm{Range}\left(\left[\begin{array}{cc} M-I & b_T \end{array}\right]\right), \end{cases}$$

or

$$b)\ \begin{cases} \dim\ \mathrm{Null}(M-I)=1, \\ b_T \in \mathrm{Range}(M-I), \\ b_\gamma \in \mathrm{Range}\left(\left[\begin{array}{cc} M-I & b_T \end{array}\right]\right). \end{cases}$$

Distinguishing between a transcritical and pitchfork bifurcation point requires the analysis of certain higher order derivatives.

In case $b_\gamma \notin \mathrm{Range}\left(\left[\begin{array}{cc} M-I & b_T \end{array}\right]\right)$ and 1 is an algebraically double eigenvalue of the monodromy matrix, we can see from the bifurcation diagram whether there are two linearly independent eigenvectors for the double 1 Floquet multiplier or not. Let

$$\begin{cases} \varphi(x(0), T, \gamma) - x(0) = 0 \\ s(x(0), T, \gamma) = 0 \end{cases} \tag{1.19}$$

define a curve of periodic solutions where the phase condition is chosen such that the nondegeneracy condition $\frac{\partial s}{\partial x(0)} \frac{\partial \varphi}{\partial T}$ is satisfied in any point that solves (1.19). Let

$$\left[\begin{array}{ccc} M & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \end{array}\right] = \frac{\partial(\varphi, s)(x(0), T, \gamma)}{\partial(x(0), T, \gamma)}$$

at a point $(x(0), T, \gamma)$ that solves (1.19). The tangent direction $(dx, dT, d\gamma)$ to the curve of points that solve (1.19)—each point representing a different periodic orbit—is a non-zero solution of

$$\left[\begin{array}{ccc} M & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \end{array}\right] \left[\begin{array}{c} dx \\ dT \\ d\gamma \end{array}\right] = \left[\begin{array}{c} 0_{N\times 1} \\ 0 \end{array}\right]. \tag{1.20}$$

Now suppose $b_T \notin \mathrm{Range}(M-I)$. Since we assumed that 1 is an algebraically double eigenvalue of M and since

$$\nexists v : (M-I)v = b_T,$$

there must exist a second linearly independent eigenvector of M for the eigenvalue 1 and M has a Jordan block

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right].$$

Since $b_\gamma \notin \mathrm{Range}\left(\left[\begin{array}{cc} M-I & b_T \end{array}\right]\right)$, $d\gamma = 0$, and since $b_T \notin \mathrm{Range}(M-I)$, $dT = 0$. So in the bifurcation diagram, the curve of solutions of (1.19) has an extremum in both the T and γ direction.

On the other hand, suppose $b_T \in \mathrm{Range}(M-I)$, i.e., M has a Jordan block

$$\left[\begin{array}{cc} 1 & 1 \\ 0 & 1 \end{array}\right].$$

We still have $d\gamma = 0$, but now it is possible to find a solution $(dx, 1, 0)$ of (1.20) as follows.

$$(M - I)\, dx = 0,$$

so

$$dx = -v + k\, b_T$$

where v is a generalized eigenvector of M for the eigenvalue 1 such that $Mv = v + b_T$. From the last equation of (1.20), we can determine k:

$$c_s^T dx = -c_s^T v + k\, c_s^T b_T = 0 \Leftrightarrow k = \frac{c_s^T v}{c_s^T b_T}.$$

Hence in this case, the bifurcation curve has no extremum in T at that point. So for a non-degenerate fold point (where the monodromy matrix has an algebraically double eigenvalue 1), two linearly independent eigenvectors correspond to an extremum in both the T and γ directions in the bifurcation diagram.

$\mu = -1$: period doubling bifurcation. A non-degenerate period doubling occurs when a single Floquet multiplier crosses the unit circle transversally at -1 and no other Floquet multiplier is one in modulus except the trivial Floquet multiplier 1. In a period doubling point, a branch of periodic solutions with approximately doubled period appears or disappears. These solutions go twice around the original limit cycle. On the period doubled branch, the bifurcation point is seen as a pitchfork bifurcation with two independent eigenvectors. The monodromy matrix at the bifurcation point computed on the period doubled branch is the square of the monodromy matrix on the two-sided branch. The period doubling bifurcation is also known as *flip bifurcation* or *subharmonic bifurcation* (the latter since it gives rise to a new solution branch of periodic solutions with approximately half the frequency of the original branch).

The mathematical conditions for the transversal crossing of the unit circle by the Floquet multiplier -1 are given in terms of the time integration operator by the following theorem.

Theorem 1.11 *Suppose $(x(0), T, \gamma)$ is a period doubling point. Suppose $+1$ and -1 are algebraically simple eigenvalues of $\varphi_x = M(x(0), T, \gamma)$. Suppose w_{-1} and v_{-1} are left and right eigenvalues of φ_x for the eigenvalue -1. The Floquet multiplier -1 crosses the unit circle with nonzero speed $\underline{iff}$*

$$k = w_{-1}^T \left[\varphi_{x\gamma} v_{-1} - \varphi_{x(x,T)} v_{-1} \begin{bmatrix} \varphi_x - I & \varphi_T \\ s_x & s_T \end{bmatrix}^{-1} \begin{bmatrix} \varphi_\gamma \\ s_\gamma \end{bmatrix} \right] \neq 0$$

where the subscript x denotes differentiation with respect to $x(0)$ and $s(x(0), T, \gamma) = 0$ is any phase condition satisfying $s_x \varphi_T \neq 0$.

This condition is very similar to the conditions in [88], where they are given in terms of the $(N-1)$-dimensional Poincaré map. The proof of this theorem is rather long and left out. Note that it can be shown that the coefficient k is independent of the choice of the phase condition as long as the phase condition satisfies $s_x \varphi_T \neq 0$.

Period doubling bifurcations frequently occur as a period doubling sequence: a period doubling bifurcation is often quickly followed by another one on the period doubled branch. This scenario is known as the Feigenbaum route to chaos (see, e.g., [1]).

$\mu = \mp e^{i\theta}$: **bifurcation into torus.** In this scenario, a pair of complex conjugate eigenvalues of the monodromy matrix crosses the unit circle. Except for this pair and the trivial Floquet multiplier 1, no other Floquet multiplier lies on the unit circle. The torus bifurcation is also known as the *Neimark–Sacker* bifurcation or *Hopf bifurcation of limit cycles.*

In the analysis of the torus bifurcation, there is a distinction between whether the eigenvalues cross the unit circle at a root of unity or not, i.e., between whether there exists an $n \in \mathbb{N} : \mu^n = 1$ or not or equivalently, whether $2\pi/\theta$ is a rational or irrational number. In the first case, we have resonance phenomena. Especially the cases $\theta = 2\pi/3$ and $\theta = 2\pi/4$ require special analysis. These two regimes are known as strong resonance. If the eigenvalues are another root of unity, the regime is known as weak resonance. In the latter case, the Poincaré section has at one side of the bifurcation an invariant "circle" of periodic points with alternating stability, corresponding to the existence of two periodic solutions with different stability nearby the original solution. When the critical Floquet multipliers are not a root of unity, there are no new periodic trajectories, but there is an invariant surface at one side of the bifurcation point. The precise analysis is beyond the scope of this work. A complete analysis can be found in [2, 43].

More complex situations are possible where multiple eigenvalues cross the unit circle simultaneously or where a Floquet multiplier touches the unit circle and returns or crosses the unit circle with zero speed. These situations are not generic in a one-parameter system. If the system is perturbed, the bifurcations unfold and the bifurcation diagram changes considerably. Pitchfork and transcritical bifurcations are also nongeneric, but because of the symmetries in our test problems, some pitchfork bifurcations will occur.

1.3 Stability of periodic solutions of PDEs

If (1.4) is the result of the space discretization of a PDE, the monodromy matrix has a very high dimension. However, for many important problems, the periodic solutions have no or just a few unstable modes and most of the Floquet multipliers are extremely close to 0. Moreover, as the dimension of the space discretization is increased, the dominant eigenvalues of the monodromy matrix show little or no changes and additional Floquet multipliers are added very close to 0 (at least if the original discretization captured the important physics of the problem well enough). The dominant Floquet multipliers are also independent of the discretization method used. They represent perturbation modes that damp out slowly or grow in time. Floquet multipliers very close to 0 correspond to perturbation modes that damp out very quickly and are not observed in the medium- or long-time behaviour of the system.

As an illustration, consider Figure 1.2 and Table 1.1. They show the spectrum for a periodic solution of the one-dimensional Brusselator model

$$\begin{aligned} \frac{\partial X}{\partial t} &= \frac{D_X}{L^2}\frac{\partial^2 X}{\partial z^2} + X^2 Y - (B+1)X + A, \qquad (1.21) \\ \frac{\partial Y}{\partial t} &= \frac{D_Y}{L^2}\frac{\partial^2 Y}{\partial z^2} - X^2 Y + BX, \end{aligned}$$

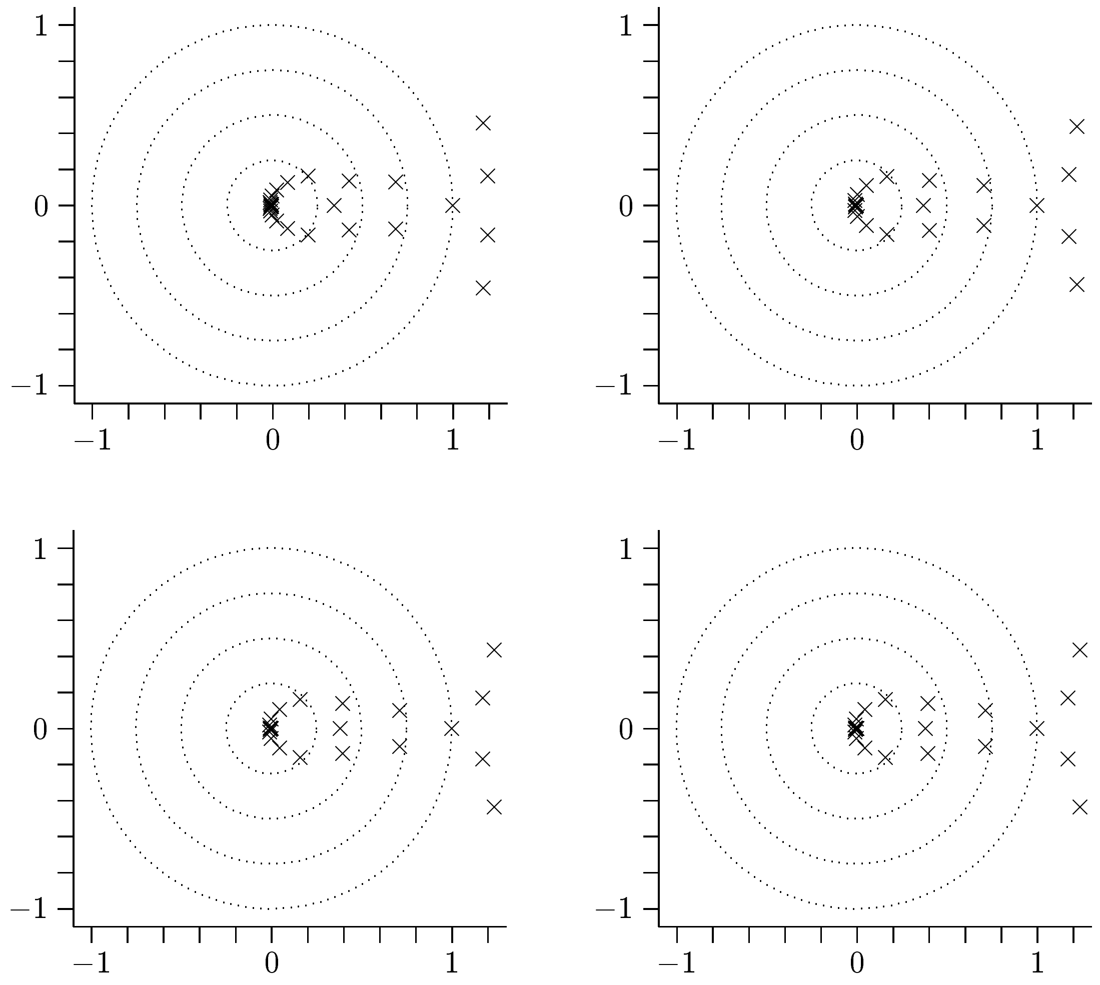


Figure 1.2: Floquet multipliers for a periodic solution of the Brusselator model at $L = 1.9$, with 15 (upper left), 31 (upper right), 63 (lower left) and 127 (lower right) discretization points.

with Dirichlet boundary conditions

$$\begin{aligned} X(t, z = 0) &= X(t, z = 1) = A, \\ Y(t, z = 0) &= Y(t, z = 1) = \frac{B}{A}, \end{aligned} \tag{1.22}$$

for $L = 1.9$, $A = 2$, $B = 5.45$, $D_X = 0.008$ and $D_Y = 0.004$. We used a second order finite difference scheme for the space discretization of the equations with 15, 31, 63 and 127 discretization points. Note that the dominant Floquet multipliers do not change a lot. We observe some differences between the pictures for 15 and 31 discretization points. When switching from 63 to 127 discretization points, the dominant Floquet multipliers only change in the fourth significant digit.

Similar observations have been made by other authors, e.g., in [7], where periodic solutions of the incompressible Navier-Stokes equations on a 2D-geometry are studied. In [121], the authors study an extensively chaotic regime in the Kuramoto-Sivashinsky

15 points	31 points	63 points	127 points
$1.1684 \pm 0.4576i$	$1.2219 \pm 0.4387i$	$1.2367 \pm 0.4360i$	$1.2367 \pm 0.4361i$
$1.1958 \pm 1.1631i$	$1.1780 \pm 0.1716i$	$1.1710 \pm 0.1692i$	$1.1710 \pm 0.1692i$
1.0000	1.0000	1.0000	1.0000
$0.6840 \pm 0.1320i$	$0.7065 \pm 0.1114i$	$0.7131 \pm 0.1003i$	$0.7131 \pm 0.1003i$
$0.4257 \pm 0.1341i$	$0.4026 \pm 0.1389i$	$0.3951 \pm 0.1393i$	$0.3951 \pm 0.1393i$
0.3425	0.3720	0.3823	0.3823
$0.1986 \pm 0.1641i$	$0.1673 \pm 0.1622i$	$0.1593 \pm 0.1611i$	$0.1593 \pm 0.1611i$
$0.0866 \pm 0.1280i$	$0.0537 \pm 0.1112i$	$0.0464 \pm 0.1064i$	$0.0464 \pm 0.1064i$

Table 1.1: Dominant Floquet multipliers for one of the periodic solutions of the Brusselator model at $L = 1.9$.

equation

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, \quad u = \frac{\partial u}{\partial x} = 0 \text{ at } x = 0 \text{ and } x = L,$$

by computing a lot of unstable orbits in the attractor for $L = 50$. The attractor in this problem has an already large Lyapunov fractal dimension of 8.8. However, the periodic orbits have seldomly more than five Floquet multipliers outside of the unit circle or twelve Floquet multipliers larger than 0.1 in modulus [120].

Although partial differential equations are intrinsically infinite-dimensional problems, in many cases, only a few Floquet multipliers are rather large and play an active role. We will exploit this behaviour in our methods to reduce the computing time of periodic solutions to a more acceptable level.

1.4 Techniques for computing periodic solutions

Several techniques have been developed for the computation of periodic solutions of ODEs. Single or multiple shooting and orthogonal collocation with spline basis functions are the most widely used techniques in existing software packages for the continuation of periodic solutions of ODEs. Finite difference techniques should also be considered for PDE problems. In this section, we will discuss each of these techniques in some more detail with some of their merits and disadvantages and indicate why traditional implementations of these methods are computationally too expensive for solving large ODE problems coming from the space discretization of PDEs. It is not our intention to give a complete overview of all possible techniques, rather we want to introduce single- and multiple shooting to the reader. We will also discuss some collocation and finite difference techniques since we are convinced that the solution techniques for shooting proposed in this thesis can also be generalized to those particular collocation and finite difference methods.

1.4.1 Single shooting

A single shooting technique solves the nonlinear system (1.6) for the unknowns $x(0)$ and T. $\varphi(x(0), T, \gamma)$ is computed with an initial value solver. Since ODE systems resulting from the discretization of parabolic PDEs are very stiff, implicit techniques should be used. The nonlinear system (1.6) is usually solved using Newton's method. This is an iterative method. At each iteration step the dense linear system

$$\begin{bmatrix} \varphi_x - I & \varphi_T \\ s_x & s_T \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \end{bmatrix} = - \begin{bmatrix} \varphi(x(0), T, \gamma) - x(0) \\ s(x(0), T, \gamma) \end{bmatrix} \tag{1.23}$$

is solved, e.g., using Gauss elimination with pivoting, and the variables are updated according to

$$\begin{aligned} &x(0) \leftarrow x(0) + \Delta x(0), \\ &T \leftarrow T + \Delta T. \end{aligned}$$

At a solution point $(x(0)^*, T^*, \gamma^*)$ of (1.6), the matrix in (1.23) is nonsingular if 1 is an algebraically simple eigenvalue of φ_x and if $s_x \varphi_T \neq 0$. The latter condition is precisely the transversality condition (1.11) for the Poincaré section since φ_T is the tangent vector to the limit cycle in $(x(0)^*, T^*, \gamma^*)$. Under the above conditions, φ_T is an eigenvector for the algebraically simple eigenvalue 0 of $\varphi_x - I$. The nonsingularity of (1.23) is then a simple corollary of the following lemma.

Lemma 1.12 *Let $A \in \mathbb{R}^{N \times N}$ have an algebraically simple eigenvalue 0 with corresponding eigenvector v. Let $b, c \in \mathbb{R}^N$, $d \in \mathbb{R}$. Suppose $b \notin \text{Range}(A)$. Let*

$$\bar{A} = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix}$$

then $\bar{A}$ is singular $\underline{\text{iff}}$ $c^T v = 0$.

Proof. $\bar{A}$ is singular $\underline{\text{iff}}$ the system

$$\begin{bmatrix} A & b \\ c^T & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad x \in \mathbb{R}^N, \quad y \in \mathbb{R}, \tag{1.24}$$

has nonzero solutions. The first N equations read $Ax + by = 0$. Since $b \notin \text{Range}(A)$, $y = 0$. From $Ax = 0$ follows $x = kv$, $k \in \mathbb{R}$, and after substitution in the last equation we get $k\,c^T v = 0$. So (1.24) only has nonzero solutions if $c^T v = 0$. □

$\varphi_x(x(0), T, \gamma) = M(x(0), T, \gamma)$ can be computed using finite difference techniques or by solving the variational equation. In the former case, column i of M is computed as the directional derivative of $\varphi(x(0), T, \gamma)$ in de direction of the i^{th} unit vector e_i, e.g., using the formula

$$M[i] = Me_i = \frac{\varphi(x(0) + \varepsilon e_i, T, \gamma) - \varphi(x(0), T, \gamma)}{\varepsilon}. \tag{1.25}$$

With this scheme, $N + 1$ time integrations of (1.4) have to be performed to compute the matrix M. The formula (1.25) is only first order accurate and more sophisticated schemes exist. These schemes use more time integrations of (1.4) to obtain a higher accuracy. It is important to make sure that the same time grid—and for a variable step-variable order method also the same sequence of orders—is used for all related time integrations to avoid loss of accuracy. In the variational equation approach, $M(t = T)$ is solved from the initial value problem

$$\frac{dM}{dt} = f_x(\varphi(x(0), t, \gamma), \gamma)\, M \tag{1.26}$$

with $M(t = 0) = I$ and $\varphi_x(x(0), T, \gamma) = M(t = T)$. This is an $N \times N$-dimensional linear initial value problem (IVP) provided $\varphi(x(0), t, \gamma)$ is known for all $t \in [0, T]$. Note that because of the structure of (1.26), the numerical solution of the variational equations can be computed very efficiently if direct linear system solvers are used to solve the linear systems that arise in implicit integrator schemes. At a given time step one gets the same linear system with a different right-hand side for each of the columns of M. It is clear that the computation of M is expensive if the ODE system is large, especially if finite differences are used or the variational equations have to be solved with iterative techniques. Furthermore, a dense $(N + 1) \times (N + 1)$-system has to be solved at every step in Newton's method. Usually the cost of this operation is much lower than the cost for the construction of M, although in some cases where (1.4) and (1.26) can be integrated efficiently, the cost of solving (1.23) can become dominant [121]. Storing the large and dense matrix M can also pose problems. It is clear that if we want to use single shooting for large systems, we must develop a method that avoids the construction of $M(x(0), T, \gamma)$.

The single shooting method has several advantages.

1. The scheme is very simple. This explains the popularity of the method. Several packages are based on this method, e.g., Locbif [68] and CANDSYS/QA [40].

2. The scheme can be implemented easily on top of an existing simulation code if finite difference techniques are used to construct M. This is particularly interesting if $f(x, \gamma)$ comes from a PDE since PDEs usually require special-purpose time integration codes. These codes are often hard to write and hard to change.

3. The monodromy matrix shows up during the computations, so it is easy to compute the Floquet multipliers of the periodic solution and to locate bifurcation points.

However, despite these advantages, single shooting also has some serious disadvantages which limit its use for some problems:

1. The method has a very limited domain of attraction, especially for unstable limit cycles. Even a slight error in the initial conditions may result in an endpoint for the time integrator far from the periodic solution. Higher-order terms then dominate the linearization of (1.6) and Newton's method may fail. Note that an appropriate damping strategy can somewhat extend the domain of attraction.

2. For unstable limit cycles, integrating over a long time interval is nearly impossible. Numerical errors made at the first time steps will grow and the error at the end of the time integration can be very large.

3. If many periodic solutions are close to each other, we might not compute the intended periodic solution but a different one. If branches of periodic solutions are computed, there is a chance for branch jumps.

These problems are mainly caused by the fact that a global object, the limit cycle, is represented by only one point. Therefore we will now describe three techniques that use more points to represent the limit cycle or even represent the limit cycle as a function of time.

1.4.2 Multiple shooting

Multiple shooting for boundary value problems was proposed by Keller in [65]. The disadvantages of single shooting are alleviated by representing the limit cycle by a discrete set of points instead of just one point. Consider a partition of the unit interval

$$s_i, \quad i = 0, \ldots, m, \tag{1.27}$$

with

$$0 = s_0 < s_1 < \cdots < s_{m-1} < s_m = 1.$$

The limit cycle is represented by the m points

$$x_i = x(s_i T), \quad i = 0, \ldots, m-1.$$

Let

$$\nabla s_i = s_i - s_{i-1}.$$

Multiple shooting solves the nonlinear system

$$\left\{ \begin{array}{rcl} \varphi(x_0, \nabla s_1 T, \gamma) - x_1 & = & 0, \\ \varphi(x_1, \nabla s_2 T, \gamma) - x_2 & = & 0, \\ & \vdots & \\ \varphi(x_{m-2}, \nabla s_{m-1} T, \gamma) - x_{m-1} & = & 0, \\ \varphi(x_{m-1}, \nabla s_m T, \gamma) - x_0 & = & 0, \\ s(x_0, x_1, \ldots, x_{m-1}, T, \gamma) & = & 0, \end{array} \right. \tag{1.28}$$

for the unknown points x_i, $i = 0, \ldots, m-1$, and T. The Newton–Raphson procedure requires the repetitive solution of linear systems

$$\begin{bmatrix} G_1 & -I & & & b_{T,1} \\ & G_2 & -I & & b_{T,2} \\ & & \ddots & \ddots & \vdots \\ -I & & & G_m & b_{T,m} \\ c_{s,1}^T & c_{s,2}^T & \cdots & c_{s,m}^T & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_{m-1} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \\ s \end{bmatrix}, \tag{1.29}$$

where

$$G_i = \left. \frac{\partial \varphi(x, t, \gamma)}{\partial x} \right|_{(x_{i-1}, \nabla s_i T, \gamma)}. \tag{1.30}$$

At the limit cycle, $G_m G_{m-1} \cdots G_1$ is the monodromy matrix computed in x_0 and $b_{T,m}$ is the eigenvector for the trivial 1 Floquet multiplier. Similarly, $G_1 G_m \cdots G_2$ is the monodromy matrix computed in x_1 with $b_{T,1}$ the eigenvector for the trivial Floquet multiplier, and so on. The matrices G_i can again be computed using finite differences or the variational equations. The total cost for the construction of (1.29) is approximately the same as for the construction of (1.23) in the single shooting technique since each of the matrices G_i require only time integration over a fraction of the period. One way to exploit the structure of (1.29) is the use of Gauss elimination with a special partial pivoting strategy that avoids the creation of too much fill-in. However, such a method is not unconditionally stable and may fail. A method of this type is used in the linear system solver of the collocation based package AUTO [27] (see the next section). Techniques that rely on a forward recursion to solve (1.29) can be dangerous since the forward recursion is unstable for unstable limit cycles. In [26], the authors propose an iterative refinement technique to cure this problem in the case of a two-point boundary value problem. In [4], a strategy is proposed using a combined forward and backward recurrence for two-point boundary value problems. We will indicate how this procedure can be adapted for periodic boundary conditions in chapter 6. Some changes are needed because of the additional bordering from the phase condition and the derivatives with respect to T. This technique uses the periodic Schur decomposition [11] discussed in chapter 5. The periodic Schur decomposition can also be used to compute the Floquet multipliers with much greater accuracy than techniques that construct the monodromy matrix $G_m G_{m-1} \cdots G_1$ explicitly.

Multiple shooting has a lot of advantages.

1. It can be implemented on top of an existing simulation code just as the single shooting method.

2. It is still possible to recover the Floquet multipliers from the computations. In chapter 5 we shall see that using the periodic QR algorithm, the Floquet multipliers can be computed with very high accuracy, even in the presence of very large or small Floquet multipliers.

3. The disadvantages of the single shooting method are reduced. As more and more time intervals are used, the domain of attraction increases and branch jumps become less likely. Also, since we integrate over shorter time intervals, errors in the time integration do not build up that much.

Compared to single shooting, the main disadvantage is the higher complexity. One must be very careful as some techniques that exploit the structure of (1.29) are numerically unstable and can ruin the performance of the multiple shooting code.

Several multiple shooting codes were written for two-point and multipoint boundary value problems, e.g., BOUNDPACK [4, 79, 80] and BOUNDS [26]. In Bifpack [105], multiple shooting is used for the computation of periodic solutions. Multiple shooting methods are useful for studying PDE problems since they can easily be combined with existing time integration codes and offer better performance than single shooting methods if implemented carefully. The collocation and finite difference techniques discussed next are also very reliable—probably more reliable than multiple shooting—but they cannot be combined as easily with existing simulation codes.

Note that sometimes a variant of (1.28) is used where an extra unknown x_m is introduced to express the periodic boundary conditions as a separate set of equations. In this approach, the system

$$\left\{\begin{array}{rcl} \varphi(x_0, \nabla s_1 T, \gamma) - x_1 & = & 0, \\ & \vdots & \\ \varphi(x_{m-1}, \nabla s_{m-1} T, \gamma) - x_m & = & 0, \\ x_m - x_0 & = & 0, \\ s(x_0, x_1, \ldots, x_{m-1}, x_m, T, \gamma) & = & 0, \end{array}\right. \tag{1.31}$$

is solved. The separation of the periodic boundary conditions from the other equations may have some advantages if codes are developed that are also to be used for the computation of bifurcation points, since in this case, similar boundary value problems with different non-separated boundary conditions have to be solved.

1.4.3 Collocation techniques

Collocation methods search for an approximate solution in a finite-dimensional function space and impose that that solution should satisfy the boundary conditions (if not trivially satisfied by the chosen function space) and satisfy (1.4) in a given set of points. AUTO, one of the most popular packages for bifurcation analysis, is based on collocation. In this section, we will restrict ourselves to collocation as implemented in AUTO because of the following reasons.

1. This is one of the most popular types of collocation for the computation of periodic solutions.
2. The collocation technique used in AUTO has some links with time integration; this will become clear in this section. We expect that it is possible to generalize the techniques proposed in this thesis to the collocation method as used in AUTO (see section 8.2.2 for some suggestions on how this could be done).
3. Because of the link with time integration, AUTO can compute the Floquet multipliers cheaply during the solution procedure.

Although AUTO can solve more general boundary value problems, we will restrict ourselves to the computation of periodic solutions.

In AUTO, the boundary value problem (1.4) with boundary conditions (1.5) is transformed using the transformation $t = Ts$ to the problem

$$\frac{dx}{ds} = Tf(x, \gamma) \tag{1.32}$$

with boundary conditions

$$x(1) = x(0) \tag{1.33}$$

on the unit interval. AUTO uses generalized spline functions which are continuous at the boundary of mesh intervals but have discontinuous derivatives. Suppose we have m mesh intervals. Let

$$s_i, \quad i = 0, \ldots, m, \tag{1.34}$$

be a mesh on the unit interval such that

$$0 = s_0 < s_1 < \cdots < s_{m-1} < s_m = 1. \tag{1.35}$$

On each interval i, the spline polynomial of degree n is expressed in terms of the Lagrange polynomials

$$l_{i,j}(\tau) = \prod_{\substack{l=0\\ l\neq j}}^{l=n} \frac{s - s_{i+l/n}}{s_{i+j/m} - s_{i+l/n}} \text{ with } s_{i+j/n} = s_i + \frac{j}{n}(s_{i+1} - s_i),\ j = 0, \dots, n.$$

Here the index i, $i = 0, \dots, m-1$, denotes the mesh interval and j, $j = 0, \dots, n$, the point in the mesh interval. Remark that

$$l_{i,j}(s_{i+l/n}) = \delta_{l,j}$$

with $\delta_{l,j}$ the Kronecker delta. Polynomials of degree n have $n+1$ degrees of freedom, but one degree of freedom is used per mesh interval for the continuity requirements and the periodicity constraint. (There are m mesh intervals and $m-1$ continuity constraints, one for each mesh point s_i, $i = 1, \dots, m-1$, and one periodicity constraint.) On each interval, the spline must fulfill (1.32) in the n Gauss points $z_{i,k}$, $k = 1, \dots, n$, which are the zeros of the Legendre polynomial of degree n on $[s_i, s_{i+1}]$. If

$$p_i(s) = \sum_{j=0}^{n} x_{i+j/n} l_{i,j}(s)$$

is the spline restricted to the mesh interval $[s_i, s_{i+1}]$, the collocation requirements are

$$\left.\frac{dp_i}{ds}\right|_{s\,=\,z_{i,k}} = Tf(p_i(z_{i,k}), \gamma), \quad i = 0, \dots, m-1,\ k = 1, \dots, n,$$

or

$$\begin{gathered}\sum_{j=0}^{n} \left.\frac{dl_{i,j}}{ds}\right|_{s\,=\,z_{i,k}} x_{i+j/n} = Tf\left(\textstyle\sum_{j=0}^{n} l_{i,j}(z_{i,k})\, x_{i+j/n}, \gamma\right), \\ i = 0, \dots, m-1,\ k = 1, \dots, n.\end{gathered} \tag{1.36}$$

This type of collocation is known as Gauss–Legendre collocation and is also used in COLSYS [3], a package for multipoint boundary value problems. In [4, 117], it is shown that this type of collocation is equivalent to a particular n-step implicit Runge–Kutta formula which is therefore also called a Gauss–Legendre Runge–Kutta formula. As will become clear in the remainder of this section, Gauss–Legendre collocation is very similar to multiple shooting with a Gauss–Legendre Runge–Kutta time integrator on the same grid. In fact, properties of the accuracy of Gauss–Legendre collocation also hold for the results of the corresponding multiple shooting method, and properties proved for Gauss–Legendre Runge–Kutta time integration schemes (such as properties of the recovery of qualitative features of the solution) also apply to the result of the Gauss–Legendre collocation method.

The system (1.36) is further augmented with the periodic boundary condition

$$x_m = x_0 \tag{1.37}$$

and a phase condition. In [27], the author argues for the use of an integral phase condition

$$\int_0^1 x(\tau)^T \left.\frac{d\tilde{x}}{ds}\right|_{s\,=\,\tau} d\tau = 0, \tag{1.38}$$

where $\tilde{x}(s)$ is a reference solution. The aim of this phase condition is to avoid a phase shift of the solution compared to the reference solution. Phase conditions that involve only one point or a few points on the limit cycle have the disadvantage that sharp peeks may move, requiring remeshing if a time-adaptive mesh is used. (1.38) is discretized with a quadrature formula that only uses the function values $x_{i+j/n}$ at the points $s_{i+j/n}$. Note that on each interval $[s_i, s_{i+1}]$, the points $s_{i+j/n}$ are equidistant. Hence we obtain the condition

$$\sum_{i=0}^{m-1} \sum_{j=0}^{n} w_{i,j} x_{i+j/n}^T \left.\frac{d\tilde{x}}{ds}\right|_{s\,=\,s_{i+j/n}} = 0, \tag{1.39}$$

where $w_{i,j}$ are the weights of the Newton-Cotes quadrature formula on the interval $[s_i, s_{i+1}]$ [21]. These weights depend on n and $s_{i+1} - s_i$. Note that AUTO always uses pseudo-arclength continuation as explained in the next section. This method allows the parameter γ to vary and adds another integral constraint to the system. We will omit this from our discussion. (1.36), (1.37) and (1.39) together constitute a large nonlinear system in the $nm + 1$ vector unknowns $x_{i+j/n}$, $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$, and x_m, and the scalar unknown T. This system is solved using Newton's method. Suppose $n = 2$ and $m = 3$. After linearization, we obtain a system with the matrix

$$\left[\begin{array}{ccccccc|c}
* & * & * & & & & & \\
* & * & * & & & & & \\
 & & * & * & * & & & \\
 & & * & * & * & & & \\
 & & & & * & * & * & \\
 & & & & * & * & * & \\
-I & & & & & & I & \\
- & - & - & - & - & - & - &
\end{array}\right]. \tag{1.40}$$

The last column in this matrix comes from the derivatives with respect to T; the last row contains the partial derivatives of the phase condition. Each $*$ in this matrix represents a $N \times N$-matrix

$$A_{i,j,k} = l_{i,j}(z_{i,k}) T \left.\frac{\partial f}{\partial x}\right|_{(p_i(z_{i,k}),\gamma)} - \left.\frac{dl_{i,j}}{ds}\right|_{s\,=\,z_{i,k}} I_N \tag{1.41}$$

for some i, j and k. In (1.40), the three blocks of 2×3 stars correspond to $i = 0$, $i = 1$ and $i = 2$. Within each 2×3 block, the rows correspond to different values for the index k and the columns with the index j. If f is the result from the space discretization of a PDE with a finite difference or finite element technique, this is a large but sparse matrix. AUTO has a specialized linear system solver that exploits the block structure of (1.40) and solves (1.40) in such a way that the Floquet multipliers can also be computed. However, AUTO cannot exploit the sparsity of each of the matrices (1.41). Therefore AUTO does not scale to large problems.

We will briefly discuss how (1.40) is solved because the procedure is of further interest in this text. In the first step, the internal unknowns on each mesh interval are eliminated using the method of condensation of parameters [22]. By means of Gauss elimination, (1.40) is transformed into

$$\left[\begin{array}{cccccccc}
* & \bigtriangledown & * & & & & & | \\
\boxed{*} & & \boxed{*} & & & & & \boxed{|} \\
 & & * & \bigtriangledown & * & & & | \\
 & & \boxed{*} & & \boxed{*} & & & \boxed{|} \\
 & & & & * & \bigtriangledown & * & | \\
 & & & & \boxed{*} & & \boxed{*} & \boxed{|} \\
\boxed{-I} & & & & & & \boxed{I} & \\
\boxed{-} & & \boxed{-} & & \boxed{-} & & \boxed{-} &
\end{array}\right] \tag{1.42}$$

where $\bigtriangledown$ denotes an upper triangular matrix. AUTO86 uses no pivoting at this step. However, one can use row pivoting within the block-row without destroying the structure. System (1.42) is separated. We can first solve the unknowns at the boundaries of each mesh interval from the blocked entries which read

$$\left[\begin{array}{ccccc}
G_1 & -H_1 & & & | \\
 & G_2 & -H_2 & & | \\
 & & G_3 & -H_3 & | \\
-I & & & I & \\
- & - & - & - &
\end{array}\right] \tag{1.43}$$

and then compute the other unknowns. At the limit cycle, the Floquet multipliers are the eigenvalues of $H_m^{-1}G_m \cdots H_1^{-1}G_1$ and can be recovered from the computations.

Now imagine we premultiply block-row i, $i = 1, \ldots, m = 3$ in (1.43) with H_i^{-1}. This would result in a system with matrix

$$\left[\begin{array}{ccccc}
\tilde{G}_1 & -I & & & | \\
 & \tilde{G}_2 & -I & & | \\
 & & \tilde{G}_3 & -I & | \\
-I & & & I & \\
- & - & - & - &
\end{array}\right]. \tag{1.44}$$

This matrix has the same form as the matrix one obtains after linearization of the multiple shooting system (1.31). In fact, it is basically the same as the linear system one would obtain with multiple shooting with the corresponding Gauss–Legendre Runge–Kutta method on the same time grid and using one time-step per shooting interval!

The linear system solver in AUTO continues with the elimination of G_2 in (1.43) by using H_1 (and creating fill-in in the first block column) and by then eliminating G_3

in the third block-row with the help of the new H_2-block. At the same time, the H_i, $i = 1, \ldots, m-1$, blocks are triangularized. This results in the system

$$\begin{bmatrix} * & \triangledown & & & | \\ * & & \triangledown & & | \\ \boxed{G_0} & & & \boxed{-H_0} & \boxed{|} \\ \boxed{-I} & & & \boxed{I} & \\ \boxed{-} & & & \boxed{-} & \end{bmatrix}. \tag{1.45}$$

The unknowns Δx_0, Δx_m and ΔT are solved from the blocked subsystem. Then the other unknowns at the mesh interval boundaries can be computed, and finally the unknowns in the interior of the mesh intervals.

In AUTO86, the Floquet multipliers are computed from the matrix $H_0^{-1}G_0$. As argued in [37, 38], this leads to loss of accuracy of the smaller Floquet multipliers in the presence of larger Floquet multipliers. Therefore, in [37, 38] the authors propose to solve the generalized eigenproblem $G_0 v = \mu H_0 v$ instead using the QZ-algorithm, which gives better results in many cases. However, this method will also fail as the big Floquet multipliers get even bigger and the small ones smaller. Note that it is especially important to be able to compute the Floquet multipliers in the neighbourhood of the unit circle accurately. This is a hard problem for limit cycles with very large Floquet multipliers (as can occur on the road to a homoclinic bifurcation). Therefore we will suggest in chapter 5 to use the periodic Schur decomposition to compute the Floquet multipliers from (1.43). It is expected that this algorithm can compute the Floquet multipliers with better accuracy than the QZ-based approach proposed in [37, 38] if extremely large or small Floquet multipliers occur.

The main advantage of collocation is its robustness. AUTO has been successfully applied to many hard problems, and has been used successfully to compute periodic solutions with large periods or steep time gradients, see, e.g., the various demos supplied with AUTO97 [31]. However, collocation methods also have some disadvantages.

1. Collocation techniques have higher memory requirements than multiple shooting methods since the whole orbit is stored. The AUTO implementation is also not able to exploit the structure of the Jacobian of f, which further increases the amount of memory needed.

2. The time integration is an inherent part of the technique. Therefore, AUTO cannot interface with existing simulation packages. Instead, one needs to develop an efficient solver for (1.40) that can exploit the structure completely if one wants to compute periodic solutions of PDEs with reasonable efficiency using collocation.

As already noted before, Gauss–Legendre collocation has many similarities with multiple shooting combined with the corresponding implicit Runge–Kutta time integrator. The main difference lies in how the nonlinear problem is solved. In multiple shooting, the internal unknowns on each mesh interval and—if more than one time-step is used on each multiple shooting interval—some of the unknowns at the mesh interval boundaries are solved from the time integration of the nonlinear system. This requires some Newton–Raphson steps at every time step. There is another sequence of Newton–Raphson steps needed to solve the multiple shooting system (1.28) itself. In a collocation technique, all unknowns are solved from one huge nonlinear system, which is linearized in one global Newton–Raphson step.

It is often said that one of the advantages of collocation over multiple shooting is the possibility to use an integral phase condition, which has clear advantages if time-adaptive meshes are used. However, it is possible to use an integral phase condition in shooting codes also, as we will show in section 3.7, but an efficient implementation requires the storage of the full trajectory, which is not in the spirit of shooting methods. It is clear that hybrid techniques combining elements from collocation such as an integral phase condition and multiple shooting are worth to consider.

Note that much of what has been said in this section will generalize to other collocation methods that are based on a (generalized) spline representation of the limit cycle or some other types of piecewise functions as long as there is a strict relation with an initial value problem method. This is not the case for every collocation method. It must be possible to derive a formula from which the value x_{i+1} in the end point can be computed in terms of the value x_i in the start point. This is clearly the case in the technique which we discussed since the basis functions on each interval are chosen in such a way that the coefficients of the function representing the limit cycle are nothing but the value of x in the start- and end point of each interval and in some intermediate points. Not all collocation methods have this property. Consider for instance the use of Fourier modes up to a given wavenumber to represent the limit cycle (see, e.g., [32] for such a method for delay differential equations). The coefficients of the function representing the limit cycle in a sine-cosine basis cannot be interpreted as values of x in some points. There is no direct relationship with a time integration scheme, and the linearized problem has a totally different matrix structure. Therefore, the Fourier technique does not allow to easily recover the Floquet multipliers from the computations.

1.4.4 Finite difference techniques

This type of method is not used in the more popular bifurcation analysis packages. It has been used though for the solution of two-point boundary value problems and implemented in the code PASVA3 [73, 91]. We will briefly recall the ideas behind the method since we feel it might be an interesting method for PDE systems. A detailed discussion can be found in [4].

Let us start again from the rescaled problem (1.32) and suppose we have a mesh (1.34)-(1.35). We search for approximate solutions of (1.32) in the meshpoints (1.34). The time derivative in (1.32) is replaced with the finite difference approximation

$$\frac{x_{i+1}-x_i}{s_{i+1}-s_i}, \tag{1.46}$$

which is a second order approximation for the derivative of x with respect to the scaled time in the midpoint of the interval $[s_i, s_{i+1}]$ and a first order approximation for the derivative in the end points. To obtain a second order accurate scheme, we have to approximate

$$f\left(x\left(\frac{s_i+s_{i+1}}{2}\right),\gamma\right)$$

with enough accuracy. Two options are often used in the literature:

$$f\left(x\left(\frac{s_i+s_{i+1}}{2}\right),\gamma\right) \approx \frac{f(x_i,\gamma)+f(x_{i+1},\gamma)}{2}, \tag{1.47}$$

and

$$f\left(x\left(\frac{s_i+s_{i+1}}{2}\right),\gamma\right) = f\left(\frac{x_i+x_{i+1}}{2},\gamma\right). \tag{1.48}$$

(1.46) combined with (1.47) results in the trapezoidal rule

$$\frac{x_{i+1}-x_i}{s_{i+1}-s_i} = \frac{f(x_i,\gamma)+f(x_{i+1},\gamma)}{2}, \tag{1.49}$$

while (1.46) with (1.48) is the midpoint rule

$$\frac{x_{i+1}-x_i}{s_{i+1}-s_i} = f\left(\frac{x_i+x_{i+1}}{2},\gamma\right). \tag{1.50}$$

The midpoint rule is also the Gauss–Legendre collocation formula for $n = 1$. The trapezoidal rule corresponds to the familiar Crank–Nicholson scheme used for discretizing time-dependent PDEs.

Both schemes combined with the periodic boundary conditions (1.37) and a suitable phase condition, e.g., a discretization of the integral phase condition (1.38) with the trapezoidal rule, result in a large nonlinear system for the $m+1$ vector unknowns x_i, $i = 0,\ldots,m$, and the scalar unknown T. After linearization of the system, we obtain a system of the form (1.43) and the solution procedure and the computation of the Floquet multipliers can proceed as in the collocation case.

A disadvantage of the above schemes is that they have only second order time accuracy and thus need a rather fine mesh. For PDEs, classical simulation techniques are often only first-order (based on backward Euler time-stepping) or second order accurate (e.g., parabolic PDEs with the Crank–Nicholson scheme), so this is not a big point. Various solutions have been proposed to this problem, such as Richardson extrapolation, deferred correction (used in PASVA3) or the use of higher order methods. A discussion of these methods is given in [4]. Let us briefly elaborate on the last option. To avoid destroying the block structure of (1.43), we have to limit ourselves to methods of the type

$$\frac{x_{i+1}-x_i}{s_{i+1}-s_i} = \Phi(x_i, x_{i+1}, \gamma, s_{i+1}-s_i).$$

One way to obtain such schemes is by integrating (1.32) on each subinterval

$$x_{i+1}-x_i = T\int_{s_i}^{s_{i+1}} f(x(s),\gamma)\,ds$$

and then replacing the integral with a quadrature formula. To obtain higher order methods, we will either have to use derivatives of f in x_i and x_{i+1} or use values of f at intermediate points in the interval. The first option is only interesting if directional derivatives of f are easily computed. The second option again leads to implicit Runge–Kutta schemes and linearized systems of the type (1.40) instead of (1.43). In fact, if Gauss–Legendre implicit Runge–Kutta schemes are used, we recover the collocation method discussed in section 1.4.3. The difference between collocation and finite difference techniques is often blurred, see, e.g., the schemes proposed in [30]. But not every finite difference technique corresponds with a collocation method and vice-versa.

Since finite difference methods are so similar to collocation techniques, they basically share the same advantages and disadvantages. However, in contrast to collocation, we have more freedom in the choice of the time integration scheme so that it can be easier to combine a finite difference method with traditional simulation codes as long as the simulation code is based on 1-step time integration schemes. As with collocation techniques, the scalability problems come from the fact that traditional codes do not fully exploit the sparse structure of (1.40) or (1.43). Furthermore, the high memory consumption (since the whole trajectory has to be stored) can also cause problems for high-dimensional problems.

1.4.5 Conclusions

Let us briefly recall some of the points made in this section.

We discussed shooting techniques (multiple shooting and the special case of single shooting), collocation and finite difference techniques. All these techniques have serious scalability problems in their traditional implementations which prohibit the use of these codes for the analysis of systems (1.4) resulting from the space discretization of PDEs. For shooting techniques, the main problem comes from the construction of the linearized system, which requires the construction of one or more full $N \times N$-matrices. In some cases, solving these linearized systems or storing the matrix can also be too expensive. Collocation or finite difference techniques typically lead to much larger but very sparse systems. The main problem with these techniques is that the traditional implementations do not exploit the sparsity of the Jacobian matrix. This leads to an extremely high memory consumption since many sparse matrices similar to the Jacobian matrix of f are stored as full $N \times N$-matrices, and it leads to a high computation time in the linear system solver. Even with a sparse matrix solver, memory consumption can still be a problem since a lot of points on the limit cycle have to be stored.

Shooting, collocation and finite difference techniques are not totally unrelated methods. For instance, finite difference techniques based on Gauss–Legendre Runge–Kutta integrators correspond to Gauss–Legendre collocation schemes. Gauss–Legendre collocation also corresponds to a shooting technique if the same time mesh and phase condition and the corresponding Runge–Kutta formula are used, and so do finite difference and shooting techniques using the same time mesh, phase condition and one-step time integrator. If the corresponding methods both converge, both solutions will be the same. However, the basic ideas behind the methods are different. In shooting methods, we start with one or more functions defined over a time interval that satisfy (1.4) and try to fix the initial conditions of these functions and T and γ to obtain a continuous periodic solution that satisfies the constraints imposed by the phase condition and parametrizing equation. In collocation methods and finite difference techniques one usually starts with a periodic representation that does not satisfy (1.4) and the additional constraints and one tries to fix that representation. Note that in AUTO, the initial representation is periodic only if $x_0 = x_m$. However, after one Newton step, this condition is always satisfied. Collocation methods usually use a periodic function to represent the solution. The collocation conditions express that this function should satisfy (1.4) in a number of points and does not violate the additional constraints. Finite difference techniques use a set of points to represent the limit cycle. The conditions are derived from finite difference or quadrature formulas.

The shooting procedure is a two-level procedure. Time integration of the nonlinear system (1.4) is used to construct the functions on the time segments and the nonlinear system (1.28) or (1.31). This requires a Newton–Raphson procedure at each time step. Solving (1.28) or (1.31) requires another Newton–Raphson procedure. After each step of this procedure, we have to integrate (1.4) again to generate the functions with the new initial conditions. Collocation and finite difference techniques compute all unknowns at once from a big nonlinear system. The time integration of the nonlinear system (1.4) can cause multiple shooting to fail while the corresponding collocation or finite difference technique still works. Also, since less points are used to represent the limit cycle, there is a larger chance for branch jumps. It is expected that a multiple shooting technique can be almost as robust as a collocation or finite difference technique provided enough and well-chosen multiple shooting points are used. With respect to the application to PDE systems, shooting however has one big advantage over collocation and finite difference techniques. Shooting can be implemented easily by making use of an existing simulation code in a "black box" manner.

In this thesis, we will present an efficient iterative technique to solve the linearized systems (1.23) and (1.29) arising from single- and multiple shooting methods respectively and to compute the Floquet multipliers. In section 8.2.2, we will also present some ideas on how to generalize the method to finite difference or collocation schemes that lead to linearized problems of the type (1.40) or (1.43).

1.5 Pseudo-arclength continuation methods

In this thesis, we are mainly interested in computing branches of periodic solutions and not just one isolated limit cycle. Numerical algorithms for this task are known as *continuation methods.* After computing one periodic solution, a starting value for the next solution at a different parameter value is generated (the *predictor* step) and refined by the solver (the *corrector* step), and the procedure is restarted. This process is detailed in

Algorithm 1.1 *Basic continuation*

```
Generate a starting orbit for $\gamma = \gamma_0$.
loop
    $\gamma \leftarrow \gamma + \Delta\gamma$
    Generate a starting value for the orbit at parameter value $\gamma$.
    Compute the periodic orbit.
end loop
```

The first two steps in the loop implement the predictor, the last step is the corrector step.

The problem with algorithm 1.1 is that it is not able to continue past a fold point since the parameter value is always increased. Clearly, γ is a bad choice for a parametrization of the solution curve. Keller [63] proposed a reparametrization of the solution path such that the curve has no fold points in the new parameter. Let us call this parameter η. The boundary value problems (1.4)-(1.5) or (1.32)-(1.33) with a suitable phase condition need to be augmented with another condition, the *parametrizing equation*, to define the new parametrization. For the remainder of this section, we will continue with the time-scaled problem (1.32)-(1.33). The solution path $(x(s,\eta), T(\eta), \gamma(\eta))$ is implicitly defined by the boundary value problem

$$\begin{cases} \dfrac{dx}{ds} = Tf(x(s),\gamma), \\ x(1) = x(0), \\ s(x(s),T,\gamma) = 0, \\ n(x(s),T,\gamma;\eta) = 0. \end{cases} \tag{1.51}$$

With a good choice of the parametrizing equation, the solution curve $(x(s,\eta), T(\eta), \gamma(\eta))$ has no fold points in the parameter η. System (1.51) can be solved with any of the techniques discussed in the previous section. Note that γ is now also unknown. The linearized systems of any of the techniques which we discussed in the previous section have an additional column with the derivatives with respect to the parameter γ and an extra row for the derivatives of the parametrizing equation. Algorithm 1.1 using η as the control parameter instead of γ is able to successfully compute solution paths of (1.51) even if fold points in γ occur. However, (1.51) remains singular at points where multiple branches intersect.

In [63], several choices for the parametrizing equation are proposed. Ideally, γ should express some form of arclength, leading to the condition

$$n(x(s),T,\gamma;\eta) = \theta_x \int_0^1 \left\| \frac{\partial x(s,\eta)}{\partial \eta} \right\|_2^2 ds + \theta_T \left(\frac{dT}{d\eta} \right)^2 + \theta_\gamma \left(\frac{d\gamma}{d\eta} \right)^2 - 1. \tag{1.52}$$

θ_x, θ_T and θ_γ are positive real weights with $\theta_x + \theta_T + \theta_\gamma = 1$. This is a rather impractical condition. Usually, some approximation to (1.52) is used, e.g.,

$$\begin{aligned} n(x(s),T,\gamma;\eta) \;=\; & \theta_x \int_0^1 \|x(s,\eta) - x(s,\eta_0)\|_2^2 \, ds + \theta_T \left(T(\eta) - T(\eta_0)\right)^2 + \\ & \theta_\gamma \left(\gamma(\eta) - \gamma(\eta_0)\right)^2 - (\eta - \eta_0)^2 \end{aligned} \tag{1.53}$$

where $(x(s,\eta_0), T(\eta_0), \gamma(\eta_0))$ is a known solution of (1.32)-(1.33)-(1.38), or

$$\begin{aligned} n(x(s),T,\gamma;\eta) \;=\; & \theta_x \int_0^1 \left(\left.\frac{\partial x(s,\eta)}{\partial \eta}\right|_{\eta=\eta_0} \right)^T (x(s,\eta) - x(s,\eta_0)) \, ds + \\ & \theta_T \left.\frac{dT(\eta)}{d\eta}\right|_{\eta=\eta_0} (T(\eta) - T(\eta_0)) + \\ & \theta_\gamma \left.\frac{d\gamma(\eta)}{d\eta}\right|_{\eta=\eta_0} (\gamma(\eta) - \gamma(\eta_0)) - (\eta - \eta_0). \end{aligned} \tag{1.54}$$

AUTO uses an approximation to (1.54) where the derivatives with respect to η are replaced by an approximation based on the two previous solutions on the branch for $\eta = \eta_{-1}$

and $\eta = \eta_0$:

$$\begin{aligned} n(x(s),T,\gamma;\eta) \;=\;& \theta_x \int_0^1 \left(\frac{x(s,\eta_0) - x(s,\eta_{-1})}{\eta_0 - \eta_{-1}} \right)^T (x(s,\eta) - x(s,\eta_0))\, ds + \\ & \theta_T \frac{T(\eta_0) - T(\eta_{-1})}{\eta_0 - \eta_{-1}} (T(\eta) - T(\eta_0)) + \\ & \theta_\gamma \frac{\gamma(\eta_0) - \gamma(\eta_{-1})}{\eta_0 - \eta_{-1}} (\gamma(\eta) - \gamma(\eta_0)) - (\eta - \eta_0). \end{aligned} \tag{1.55}$$

Continuation strategies based on (1.53), (1.54) or (1.55) are known as *pseudo-arclength continuation* methods. (1.53), (1.54) or (1.55) are ideal for use in collocation or finite difference techniques. For shooting methods, it is better to use a parametrizing equation that only involves the shooting points. Integral parametrizing equations can be used with shooting techniques, but just as for integral phase conditions, an efficient implementation requires the storage of extra data.

Several options exist for the predictor in algorithm 1.1. A naive choice is to take the last computed orbit $(x(s,\eta_0), T(\eta_0), \gamma(\eta_0))$ as the starting orbit at $\eta_1 = \eta_0 + \Delta\eta$. Another option is to compute the secant "vector" based on the two previous solutions and then do a step in the direction of the secant vector, i.e.,

$$\begin{aligned} x(s,\eta_1) &= x(s,\eta_0) + \Delta\eta \frac{x(s,\eta_0) - x(s,\eta_{-1})}{\eta_0 - \eta_{-1}}, \\ T(\eta_1) &= T(\eta_0) + \Delta\eta \frac{T(\eta_0) - T(\eta_{-1})}{\eta_0 - \eta_{-1}}, \\ \gamma(\eta_1) &= \gamma(\eta_0) + \Delta\eta \frac{\gamma(\eta_0) - \gamma(\eta_{-1})}{\eta_0 - \eta_{-1}}. \end{aligned}$$

The secant predictor is easy to implement, but requires two previous solutions and therefore cannot be used for the second point on the branch (the first run through the loop in algorithm 1.1). Yet another option is to compute the tangent to the solution path and take a step in that direction. Let $F(x(s),T,\gamma;\eta) = 0$ represent (1.51) in operator form. The normalized tangent $(\dot{x}(s), \dot{T}, \dot{\gamma}, \dot{\eta} = 1)$ is found as the solution of

$$F_x \dot{x}(s) + F_T \dot{T} + F_\gamma \dot{\gamma} + F_\eta = 0$$

or, by differentiating (1.51) with respect to η and reordering the derivatives,

$$\left\{ \begin{array}{l} \dfrac{d\dot{x}(s)}{ds} = T f_x \dot{x}(s) + f\dot{T} + f_\gamma \dot{\gamma}, \\ \dot{x}(1) = \dot{x}(0), \\ s_x \dot{x}(s) + s_T \dot{T} + s_\gamma \dot{\gamma} = 0, \\ n_x \dot{x}(s) + n_T \dot{T} + n_\gamma \dot{\gamma} + n_\eta = 0. \end{array} \right. \tag{1.56}$$

Note that s_x and n_x are operators acting on functions! The normalization $\dot{\eta} = 1$ can be used since a good parametrizing equation guarantees that there are no fold points in η. The linear boundary value problem (1.56) is not as hard to solve as it seems. In fact, if the shooting, collocation or finite difference method is applied to it, one recovers the linearized systems obtained for the nonlinear problem (1.51) except for the right-hand side. If direct methods are used to solve the linearized system, computing the solution for an additional right-hand side is cheap and the tangent vector (or at least a good approximation to it) can be computed easily from the last Newton–Raphson step for the solution of (1.51). Both the secant and tangent predictor are first order methods. With moderate stepsizes, both perform about equally. The main advantage of the tangent predictor over the secant predictor is that the tangent predictor requires only one solution on the branch and so can be used for the second point also. The disadvantage is that it requires solving an extra boundary value problem. If most of the computations for the solution of (1.56) can be shared with those for (1.51), the computation of the tangent direction is cheap. This is not the case if the linearized systems are solved with iterative techniques. Many of these methods will not be able to fully exploit the fact that a similar system with different right-hand side has been solved already. One of the advantages of the approach we present in this thesis is the ability to compute a cheap approximation to the tangent vector because of the particular combination of direct and iterative methods we use. Since (1.56) is singular in pitchfork or transcritical bifurcation points, the tangent predictor cannot be computed in those points (or in the immediate neighbourhood because of an ill-conditioned linearized system). However, the same (almost-) singularity occurs in the linearized systems for the computation of the periodic solution and their computation will be hard also and likely to fail. Higher order predictors can also be constructed, but they seldomly give a noticeable improvement. Some experiments we did with a parabolic predictor indicated that such methods might be interesting if a pitchfork bifurcation point is approached along the one-sided branch, reducing the chance of jumping to the two-sided branch.

Another method is proposed by Rheinboldt [96]. The solution curve is locally parametrized by one of the solution components or parameters: x at one of the points of the time mesh, T or γ. A different component or parameter is chosen as needed. This strategy removes one unknown from the system and allows to reduce the size of the linearized systems with one row and column compared to the pseudo-arclength approach. In multiple shooting, collocation or finite difference methods, only the period T and the parameter γ are choices that can lead to a serious reduction of the computation time of the linear system solver. Choosing one of the components of x in one of the points on the limit cycle and removing the corresponding column from the linearized system destroys the structure and prevents the use of linear system solvers that exploit the structure. In such a case it is better to not reduce the system size and add an equation expressing that one of the components remains constant. Situations may occur where neither T nor γ is a good choice for the local parametrization. If the reduction in computation time compared to the pseudo-arclength approach is significant, one might consider to use T or γ as a local parametrization whenever possible and switch to pseudo-arclength continuation in the other regions of the branch.

In algorithm 1.1, the choice of $\Delta\gamma$ was not specified. A good continuation code will select the stepsize in function of several parameters. This leads to the more advanced algorithm

Algorithm 1.2 *Variable step pseudo-arclength continuation*

```
    Generate a starting orbit $(x^{(0)}(s), T^{(0)}, \gamma^{(0)})$
```

```
loop
    Generate a predictor direction $(\dot{x}(s), \dot{T}, \dot{\gamma})$.
    Determine the stepsize $\Delta\eta$.
    loop
        $x^{(1)}(s) \leftarrow x^{(0)}(s) + \Delta\eta\, \dot{x}(s)$, $T^{(1)} \leftarrow T^{(0)} + \Delta\eta\, \dot{T}$, $\gamma^{(1)} \leftarrow \gamma^{(0)} + \Delta\eta\, \dot{\gamma}$.
        Solve (1.51) using $(x^{(1)}(s), T^{(1)}, \gamma^{(1)})$ as the starting value. Overwrite with
        the result.
        if not successful then
            decrease the stepsize
        else
            $x^{(0)} \leftarrow x^{(1)}$, $T^{(0)} \leftarrow T^{(1)}$, $\gamma^{(0)} \leftarrow \gamma^{(1)}$.
    until convergence or minimum stepsize reached
until branch computed or minimum stepsize reached
```

If the corrector fails, the stepsize is decreased and a new starting value is generated until the minimum stepsize is reached. Several other factors may influence the stepsize.

1. If the corrector converged quickly in the previous step, the stepsize can be increased (but should not exceed a user-determined maximum).

2. If the tangent direction changes too much, the stepsize should be decreased. Strong changes in the tangent direction denote the approach of a fold point or regions where the solution curve is not as smooth as in other regions. Taking too large steps may cause failure of the corrector iterations, the skipping of details on the curve, or even worse, the computation of a point on a different branch.

3. One can also monitor bifurcation test functions along the branch. Such a function changes sign in certain bifurcation points. Examples are $\det(M + I)$ for period doubling points or the distance to the unit circle of the non-trivial Floquet multiplier closest to the unit circle. If a zero of a test function is approached, the stepsize should be reduced. This is useful to avoid jumping over two bifurcation points of the same type that annihilate each other. It can also be used to reduce the stepsize in the neighbourhood of fold points. Such strategies are most successful for smooth test functions.

A good stepsize strategy should assure that no important details of a branch are skipped and that all bifurcation points are detected. In particular, jumps to other branches should be avoided. Furthermore, the stepsize criterion should be conservative enough to avoid too frequent failure of the corrector iterations.

The discussion on the parametrizing equation and the predictor in this section was rather abstract. Those points will be made more concrete when we discuss our methods for single and multiple shooting. More information on the aspects of parametrization and steplength control can be found in [104, 106].

1.6 Computation of bifurcation points of periodic solutions

In this section, we will briefly discuss some methods for the computation of bifurcation points on a curve of periodic solutions. The computation consists of two phases. First, the continuation procedure should detect the presence of a bifurcation point. This is done by monitoring bifurcation test functions, functions that change sign at certain bifurcation points. Once a bifurcation point is detected, the point can be computed accurately. There are two families of methods for the latter purpose. Indirect methods compute a zero of a test function along the periodic solution curve by using one of the various iterative techniques for the computation of zeros of functions. Direct methods add equations expressing the conditions for a bifurcation point of the given type to the boundary value problem that defines the periodic solution, and then solve the resulting system. There are two important classes of direct methods: methods that add only one equation to the boundary value problem (or just a few for more complex bifurcation phenomena), and methods that add the (generalized) eigenvector(s) of the eigenvalue(s) related to the bifurcation to the set of unknowns and add a set of equations expressing that those extra unknowns are an eigenvector (or set of eigenvectors) for an eigenvalue on the unit circle. We will now discuss each of these methods in some more detail.

1.6.1 Indirect methods

Indirect methods compute a zero of a scalar bifurcation test function along the branch. In this case, the argument of the test function is some parameter along the curve. The evaluation of the test function consists of two phases: first a periodic solution at the given parameter value is computed, and then a function of the solution profile which changes sign at the desired bifurcation point is evaluated. The latter function need only be defined on the limit cycle. Many examples of such test functions have been proposed in the literature, see, e.g., [40, 69, 67, 106]. Let us mention just a few examples.

At a non-degenerate period doubling point, a single Floquet multiplier crosses the unit circle through -1. Therefore,

$$t_1 = \det(M^* + I) \tag{1.57}$$

is zero at such a point and changes sign. The disadvantage of this test function is its high evaluation cost ($\mathrm{O}(N^3)$) if the dimension of the system is high. Note that it is easy to compute t_1 if all Floquet multipliers are known, since

$$\det(M^* + I) = \prod_{i=1}^{N} (\mu_i^* + 1).$$

Another example is

$$t_2 = \dot{\gamma}, \tag{1.58}$$

where $\dot{\gamma}$ is the component in the γ-direction of the tangent vector to the limit cycle. $\dot{\gamma}$ can be computed from (1.56). This is a test function for fold points. As we argued before, this test function is sometimes cheap to compute. A third possibility is the distance to

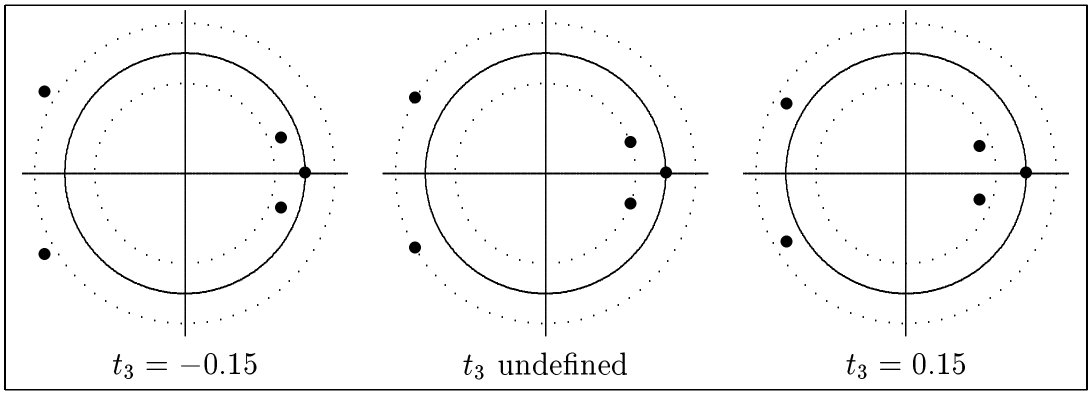


Figure 1.3: Discontinuity in the test function t_3.

the unit circle of the nontrivial Floquet multiplier closest to the unit circle. To obtain a signed function, one can use a negative sign if the Floquet multipliers lies inside the unit circle and a positive sign otherwise. Mathematically, the condition reads

$$t_3 = |\mu_i| - 1 \text{ where } \left||\mu_i| - 1\right| = \min_{j=2,\dots,N} \left||\mu_j| - 1\right| . \tag{1.59}$$

We assumed that μ_1 is the trivial Floquet multiplier. Test function (1.59) can have discontinuities at which a sign change occurs. This happens for instance if a Floquet multiplier outside of the limit circle moves to the unit circle while the one inside the limit cycle moves away from the unit circle (or vice-versa), see Figure 1.3. t_3 changes sign at every of the bifurcations discussed in section 1.2 except if a pitchfork bifurcation point is passed along the one-sided branch. Several variants on this idea are possible. Test functions based on one or two Floquet multipliers close to the unit circle are interesting for applications in PDE problems. As we shall see later, there exist good iterative techniques to compute the dominant Floquet multipliers. Many PDE problems have only few Floquet multipliers close to or outside the unit circle.

The continuation code monitors the test functions, detects sign changes and provides us with an interval on the branch that contains the bifurcation point. There exist many methods to locate the exact zero inside the interval. Of particular interest are methods that do not need the evaluation of derivatives of the test function, since those derivatives are often hard to compute. Examples are the bisection method, the regula falsi method, the secant method, the method of Muller and the method due to Van Wijngaarden, Dekker and Brent. A discussion on these methods can be found in [61, 94, 112]. The bisection method maintains an interval on which the test function changes sign and halves the interval size at every step. The regula falsi and the secant method are both based on linear interpolation. The regula falsi maintains an interval on which the function changes sign and is more robust than the secant method, but usually slower. The method of Muller is based on quadratic approximation. The method of Van Wijngaarden, Dekker and Brent is a rather complex method that tries to combine the speed of interpolation methods with the robustness of the bisection method.

1.6.2 Direct methods

In direct methods, extra equations that express the conditions for a bifurcation point are added to the boundary value problem (1.4)-(1.5) with its phase condition. The parameter γ is allowed to vary.

Methods based on scalar testfunctions

Some methods are based on the addition of a scalar test function to the boundary value problem. This test function replaces the parametrizing equation in (1.51). The value of this test function depends on the function $x(t)$, the period T and the parameter γ. This test function must not only be defined on the limit cycle, but also in a neighbourhood of the limit cycle (which is an important difference with test functions for indirect methods). Most of the test functions used with indirect methods can be generalized. t_1 (1.57) is generalized to

$$\tilde{t}_1 = \det\left(M(x(0), T, \gamma) + I\right).$$

For t_3 (1.59) the eigenvalues of $M(x(0), T, \gamma)$ are used, and for t_2 (1.58), (1.56) is solved for $\dot{\gamma}$.

This method leads to a system with the same structure as the pseudo-arclength continuation system (1.51). The parametrizing equation in (1.51) is replaced with the test function, but the solver needs no further changes. The disadvantage is that many of these test functions are expensive to compute for large problems. This certainly holds for test functions based on the determinant of a matrix derived from the monodromy matrix. Furthermore, we also need to compute first order derivatives of the test functions for the Newton linearization that is done in each of the methods we discussed in section 1.4.

Methods based on eigenvectors of the monodromy matrix

The second type of methods is based on adding the (generalized) right eigenvector(s) involved with the bifurcation to the set of unknowns and adding conditions to the boundary value problem that express that the additional vectors are eigenvectors for a $+1$ or -1 Floquet multiplier or for a pair of complex conjugate eigenvalues on the unit circle. These methods involve matrix–vector products with the monodromy matrix. The conditions on the generalized eigenvector(s) can usually be rewritten as a boundary value problem similar to the original problem. This approach is used in the AUTO package [27, 37]. We will give an example for period doubling points, fold points and torus bifurcation points.

A period doubling bifurcation is characterized by a -1 Floquet multiplier. The conditions for the corresponding eigenvector can be expressed as

$$\begin{cases} \left(M(x(0), T, \gamma) + I\right) v = 0, \\ p(v) = 0, \end{cases} \tag{1.60}$$

where $p(v)$ is a normalization on v to avoid an all-zero solution, e.g., $p(v) = v^T v - 1$ or $p(v) = l^T v - 1$ where l is a vector that is not orthogonal to the -1 eigenvector at the period doubling point. From the variational equations for the monodromy matrix (1.26) follows that $Mv(0) = v(T)$ where $v(t)$ is the solution of the initial value problem

$$\frac{dv(t)}{dt} = f_x(\varphi(x(0), t, \gamma), \gamma) v(t).$$

Hence the period doubling point can be computed from the boundary value problem

$$\begin{cases} \dfrac{dx(t)}{dt} = f(x(t), \gamma), \\ \dfrac{dv(t)}{dt} = f_x(x(t), \gamma)\, v(t), \\ x(T) = x(0), \\ v(T) = -v(0), \\ s(x(t), T, \gamma) = 0, \\ p(v(0)) = 0 \text{ or } p(v(t)) = 0. \end{cases} \tag{1.61}$$

A fold point is characterized by a double $+1$ eigenvalue of the monodromy matrix. In the generic case, M^* will have only one eigenvector direction for the $+1$ eigenvalue, but it is not hard to take into account the case with two linearly independent eigenvectors. One of the eigenvectors of M^* is known, namely $f(x(0)^*, \gamma^*)$ and we shall use that knowledge in our extended system. The conditions on the second generalized eigenvector are

$$\begin{cases} M(x(0), T, \gamma)v = v - \alpha\, f(x(0), \gamma), \\ p_1(v) = 0, \\ p_2(v) = 0. \end{cases} \tag{1.62}$$

Note that α is an unknown in this system. If α is zero at the fold point, M^* has two linearly independent eigenvectors. two normalization conditions are needed for the vector v. The first one should express that v does not lie in the same direction as $f(x(0), \gamma)$, e.g.,

$$p_1(v) = f(x(0), \gamma)^T v,$$

while the second condition then fixed the size of v, e.g.,

$$p_2(v) = v^T v - 1$$

or

$$p_2(v) = l^T v - 1$$

(where l should not be orthogonal to v at the fold point and not lie in the direction of $f(x(0)^*, \gamma^*)$). It is easy to check that

$$v(t) = \frac{\alpha t}{T} f(x(t), \gamma)$$

is the solution of the initial value problem

$$\frac{dv}{dt} = f_x(\varphi(x(0), t, \gamma), \gamma)\, v + \frac{\alpha}{T} f(\varphi(x(0), t, \gamma), \gamma) \tag{1.63}$$

with the initial condition $v(0) = 0$. Using the superposition principle, one can show that the solution at time T of (1.63) with the initial condition $v(0) = v$ is $Mv +$

$\alpha f(\varphi(x(0), T, \gamma), \gamma)$. Therefore, the equations for the periodic solution and the conditions (1.62) can be combined to the boundary value problem

$$\begin{cases} \dfrac{dx(t)}{dt} = f(x(t), \gamma), \\ \dfrac{dv(t)}{dt} = f_x(x(t), \gamma)\, v(t) + \dfrac{\alpha}{T} f(x(t), \gamma), \\ x(T) = x(0), \\ v(T) = v(0), \\ s(x(t), T, \gamma) = 0, \\ p_1(v(0)) = 0 \text{ or } p_1(v(t)) = 0, \\ p_2(v(0)) = 0 \text{ or } p_2(v(t)) = 0. \end{cases} \tag{1.64}$$

Some other variants for fold points are discussed in [37].

A torus bifurcation point is characterized by a complex eigenvalue pair $e^{\pm i\theta} = \cos\theta + i\sin\theta$ and corresponding complex eigenvectors $v \pm iw$. To avoid complex arithmetic, the condition

$$M(v + iw) = e^{i\theta}(v + iw)$$

is split in its real and imaginary parts, leading to

$$\begin{cases} Mv - \cos(\theta)\, v + \sin(\theta)\, w = 0, \\ Mw - \sin(\theta)\, v - \cos(\theta)\, w = 0. \end{cases} \tag{1.65}$$

The normalization of the complex eigenvector requires two real conditions. The eigenvector $v + iw$ multiplied with an arbitrary complex number is again an eigenvector for the same eigenvalue. In [37], the author suggest the use of

$$\begin{cases} p_1(v, w) = l_v^T v - 1 = 0, \\ p_2(v, w) = l_w^T w = 0, \end{cases} \tag{1.66}$$

where l_v^T is not orthogonal to v and l_w^T is not parallel to w. The conditions for a torus bifurcation point can be rewritten as the large boundary value problem

$$\begin{cases} \dfrac{dx(t)}{dt} = f(x(t), \gamma), \\ \dfrac{dv(t)}{dt} = f_x(x(t), \gamma)\, v(t), \\ \dfrac{dw(t)}{dt} = f_x(x(t), \gamma)\, w(t), \\ x(T) = x(0), \\ v(T) - \cos(\theta)\, v(0) + \sin(\theta)\, w(0) = 0, \\ w(T) - \sin(\theta)\, v(0) - \cos(\theta)\, w(0) = 0, \\ s(x(t), T, \gamma) = 0, \\ p_1(v(0), w(0)) = 0 \text{ or } p_1(v(t), w(t)) = 0, \\ p_2(v(0), w(0)) = 0 \text{ or } p_2(v(t), w(t)) = 0. \end{cases} \tag{1.67}$$

These methods lead to systems with two or three unknown functions and two or three scalar unknowns. At first, the larger problem size seems to be a disadvantage of these methods. However, the systems (1.61), (1.64) and (1.67) have a lot of structure that can be exploited by the solver. The first ODE does not contain terms of the unknown eigenvector(s) or the scalar unknowns α (in (1.64)) or θ (in (1.67)). The linearized systems obtained with any of the methods of section 1.4 can be reordered to bordered lower block-triangular systems. Furthermore, the blocks have related structure that can sometimes be exploited.

In chapter 7, we will discuss how the methods which we will develop in the next chapters can be generalized for the computation of bifurcation points based on the extended systems with (generalized) eigenvectors. We will also give some ideas on how the scalar testfunction approach can be generalized.

1.7 Conclusions

The main purpose of this chapter was to introduce some theory and some numerical methods on which the methods we develop in this thesis are based.

In the first part of the chapter, we discussed some theory on periodic solutions and their bifurcations. This thesis starts from the autonomous system of ordinary differential equations (ODEs) that is obtained after space discretization of a system of partial differential equations (PDEs). We defined conditions for a periodic solution, defined the monodromy matrix (this matrix gives information on the stability of the periodic orbit) and discussed the basic bifurcation phenomena that are observed if a branch of periodic solutions of a dynamical system is computed: fold points, torus bifurcation points and period doubling points and in some circumstances, pitchfork- and transcritical bifurcation points. Most texts study these bifurcations in terms of the Poincaré map. We have studied them in terms of the monodromy matrix instead since the monodromy matrix will return in all methods developed in this thesis. We also discussed a property that many PDE systems have and that will be exploited in our methods to reduce the computational costs: typically, the Floquet multipliers (eigenvalues of the monodromy matrix) are clustered around 0 and only few Floquet multipliers are large. Moreover, the larger Floquet multipliers do not depend much on the discretization.

In the second part of the chapter we concentrated on numerical methods. We first studied the advantages and disadvantages of some methods to compute a periodic solution of an ODE system: single shooting, multiple shooting, collocation (with special attention to the Gauss–Legendre collocation method used in the software package AUTO [27]) and finite difference methods. We think that the basic ideas behind our methods can be applied to all the above methods. In this thesis, we will study efficient single- and multiple-shooting-based methods. At the end, we will give some ideas on how the approach can be extended to Gauss–Legendre collocation and finite difference methods. We also studied how branches of periodic solutions can be computed by using pseudo-arclength continuation and ended the chapter with a discussion on the computation of bifurcation points. We discussed indirect methods that locate a bifurcation point on a branch by applying algorithms to compute zeros of a test function defined along the branch and two types of direct methods (techniques that rely on extended systems): methods that add a scalar condition to the system (or a few scalar conditions for higher-codimension phenomena) and techniques that use extended systems based on the monodromy matrix and its eigenvectors. The latter can also be formulated as large boundary value problems and hence are easily combined with shooting, collocation or finite difference methods.

Chapter 2

Single shooting: the basic ideas

2.1 Introduction

In section 1.4.1, we discussed the single shooting method and pointed out that the traditional implementation of the method using Newton–Raphson linearization and a direct linear system solver is not suited for problem (1.4) if N is large. The computation and storage of M are prohibitively expensive for such problems, and even solving the linearized system (1.23) may be too expensive. The cost for building M if (1.4) represents the spatial discretization of a PDE can be reduced somewhat by approximating M on a coarser grid (see [99]), but a larger reduction is desired. Furthermore, the method of [99] does not solve the problem of storing M and factorizing (1.23).

A natural approach to compute stable periodic solutions is straightforward time integration of (1.4) starting from a point sufficiently close to the limit cycle. After integrating over a long enough time interval, the trajectory will converge to the limit cycle. The period can be determined by using a Poincaré section. This approach is faster than shooting if a sufficiently accurate approximation is found after integrating over a time interval of length lT with $l < cN$ where c is the number of Newton steps needed in the single shooting method. (We assume here that the cost for the computation of one column of M is about the same as for a time integration of (1.4) over one period.) This "brute force" time integration approach is appealing if N is large and if the largest non-trivial Floquet multiplier is sufficiently small.

Note that time integration is equivalent to Picard iteration in the following sense. Let $P_\Omega(x)$ be a Poincaré map (1.12) and let $x(0)^*$ be a fixed point of that map, i.e., $x(0)^*$ is an intersection point of the limit cycle with the $(N-1)$-dimensional hypersurface Ω. This point can be computed by a fixed point iteration

$$x^{(\nu+1)} = P_\Omega(x^{(\nu)}) = \varphi(x^{(\nu)}, T_\Omega(x^{(\nu)})). \tag{2.1}$$

Although we are using N-dimensional coordinate vectors, this is basically an iteration scheme in a $(N-1)$-dimensional space since each of the points $x^{(\nu)}$ lies on the hypersurface Ω. The convergence of (2.1) is determined by the eigenvalues of the linearization P_Ω of $P_\Omega(x)$ in $x(0)^*$ (seen as a $(N-1)$-dimensional operator). These eigenvalues are the Floquet multipliers of $M^* = M(x(0)^*, T^*, \gamma^*)$ except for the trivial Floquet multiplier 1 (theorem 1.5). The fixed point iteration (2.1) converges locally around $x(0)^*$ if $r_\sigma(P_\Omega) <$

1 (where $r_\sigma(\cdot)$ denotes the spectral radius) with asymptotic linear convergence speed $r_\sigma(P_\Omega)$. If $r_\sigma(P_\Omega) \ll 1$, (2.1) will converge quickly. The scheme slows down if one of the Floquet multipliers approaches the unit circle. The latter situation occurs, e.g., as a bifurcation point is approached on the branch. If the limit cycle is unstable and $r_\sigma(P_\Omega) > 1$, (2.1) diverges. Note that $r_\sigma(P_\Omega)$ is the largest eigenvalue of P_Ω and therefore (almost) independent of the number of degrees of freedom N if (1.4) is the result of a space discretization of a PDE as long as N is large enough to represent the physics of the PDE well enough. Hence the number l of Picard iteration steps (2.1) or time integrations needed to approach the limit cycle is almost independent of N.

Besides the convergence problems for weakly stable or unstable limit cycles, the time integration approach has some other disadvantages. This approach cannot be extended to pseudo-arclength continuation or to the computation of bifurcation points. The stability information provided by the method is also limited, although we can deduce an estimate for the largest Floquet multiplier from the asymptotic convergence speed. Newton-based shooting on the other hand converges well as long as the limit cycle is only moderately unstable, can return all desired stability information, can be used with pseudo-arclength continuation and can be applied to compute bifurcation points.

In section 1.3 we noted that for a typical PDE problem, only few Floquet multipliers lie close to or outside the unit circle. Only in that subspace, the error of the time integration approach will decrease slowly or grow. In this chapter, we will develop a method that exploits this knowledge by basically combining Newton-based shooting in the directions associated with the Floquet multipliers close to or outside the unit circle (say all Floquet multipliers larger than ρ in modulus where $0 < \rho < 1$, e.g., $\rho = 0.5$, and a Picard iteration in the orthogonal directions. The idea is a generalization of the recursive projection method of Shroff and Keller [107] and the condensed Newton – supported Picard method of Jarausch and Mackens [56, 57, 58]. Because our method in its simplest form is basically a combination of a Picard iteration and Newton-based shooting, we choose to name the method the *"Newton–Picard method."*

Let us end this introduction with a short overview of the remainder of this chapter. We will work out the method for the fixed parameter case and prove an asymptotic convergence result in section 2.2. In doing so, we will recover the recursive projection method of [107], but also derive a new variant that is expected to perform better. We will show how the method can be interpreted as a Newton-like method with a special approximation to the Jacobian matrix. In section 2.3, the method is generalized to pseudo-arclength continuation. Implementation details are the topic of section 2.4. We will discuss how each step in the algorithm can be implemented and also present a variant of the orthogonal subspace iteration method [101] to compute the subspace corresponding to the large Floquet multipliers. Finally, in section 2.5, we present some test results that illustrate various claims and compare some variants. These tests confirm that our newly developed variants are superior to the method of [107]. In the next chapter, we will then further develop the best fixed-parameter and pseudo-arclength variant and develop a different, more algebraic view of the method. This view will allow us to increase the robustness of the method. A detailed comparison with the earlier work of [56, 57, 58] and [107] is postponed until chapter 4. This allows to clarify our own contributions to the method and to compare our work to the more recent work in [13] and other related research.

2.2 The fixed parameter case

2.2.1 Derivation of the method

The simple shooting method solves the nonlinear problem

$$\begin{cases} r(x(0),T;\gamma) = \varphi(x(0),T;\gamma) - x(0) = 0, \\ s(x(0),T;\gamma) = 0 \end{cases} \tag{2.2}$$

for the unknowns $x(0)$ and T. Since we fix the parameter γ at the moment, we will omit γ from our notations. We assume φ and s are twice differentiable with respect to $x(0)$ and T in the region of interest. For φ, this follows from the requirement of C^2 continuity of (1.4) in section 1.1. The use of Newton's method for (2.2) leads to the repetitive solution of linear systems in the form

$$\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} \\ c_s^{(\nu)T} & d_{s,T}^{(\nu)} \end{bmatrix} \begin{bmatrix} \Delta x(0)^{(\nu)} \\ \Delta T^{(\nu)} \end{bmatrix} = - \begin{bmatrix} r(x(0)^{(\nu)}, T^{(\nu)}) \\ s(x(0)^{(\nu)}, T^{(\nu)}) \end{bmatrix}, \tag{2.3}$$

where

$$\begin{bmatrix} M^{(\nu)} & b_T^{(\nu)} \\ c_s^{(\nu)T} & d_{s,T}^{(\nu)} \end{bmatrix} = \left. \frac{\partial(\varphi, s)(x(0),T)}{\partial(x(0),T)} \right|_{(x(0)^{(\nu)}, T^{(\nu)})},$$

followed by an update

$$\begin{aligned} x(0)^{(\nu+1)} &= x(0)^{(\nu)} + \Delta x(0)^{(\nu)}, \\ T^{(\nu+1)} &= T^{(\nu)} + \Delta T^{(\nu)}. \end{aligned}$$

From now on, we will drop the superscript (ν) from our notation whenever the formula remains clear. We will try to approximate the Jacobian matrix in (2.3) such that (2.3) can be solved approximately using just the action of M on a few vectors. This approximation will correspond to the combination of a direct method and a Picard iteration scheme.

We make the following assumptions:

Assumption 2.1 *Let $y^* = (x(0)^*, T^*)$ denote an isolated solution to (2.2), and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M(y) = \frac{\partial \varphi}{\partial x(0)}(y)$ for $y \in \mathcal{B}$ and denote its eigenvalues by μ_i, $i = 1, \ldots, N$. Assume that for all $y \in \mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{2.4}$$

and that no eigenvalue has modulus ρ; i.e., for all $y \in \mathcal{B}$

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

Note that this assumption is more restrictive than that in [107]. At y^*, M^* is the monodromy matrix. Our method is designed to be efficient for $p \ll N$. Let the column vectors of $V_p \in \mathbb{R}^{N \times p}$ define an orthonormal basis for the subspace $\mathcal{U}$ of $\mathbb{R}^N$ spanned by the (generalized) eigenvectors of $M(x(0),T)$ corresponding to the eigenvalues μ_i, $i = 1, \ldots, p$, and the column vectors of $V_q \in \mathbb{R}^{N\times(N-p)} = \mathbb{R}^{N\times q}$ define an orthonormal basis for $\mathcal{U}^\perp$, the orthogonal complement of $\mathcal{U}$ in $\mathbb{R}^N$. In general, $\mathcal{U}^\perp$ is not an invariant subspace of $M(x(0),T)$. V_p and V_q can be computed by the real Schur factorization of $M(x(0),T)$. Note that in actual computations eigenvalues may move out of or into C_ρ as the iteration proceeds and our code is able to deal with this. However as the iteration comes close to convergence, assumption 2.1 is observed in practice and so is not unreasonable.

We can construct orthogonal projectors P and Q of $\mathbb{R}^N$ onto $\mathcal{U}$ and $\mathcal{U}^\perp$ respectively as

$$\begin{aligned} P &:= V_p V_p^T, \\ Q &:= V_q V_q^T = I_N - V_p V_p^T. \end{aligned} \tag{2.5}$$

For any $x(0) \in \mathbb{R}^N$ there is the unique decomposition

$$x(0) = V_p\bar{p} + V_q\bar{q} = p + q, \quad p = V_p\bar{p} = Px(0), \quad q = V_q\bar{q} = Qx(0) \tag{2.6}$$

with $\bar{p} \in \mathbb{R}^p$ and $\bar{q} \in \mathbb{R}^{N-p}$. Substituting (2.6) in (2.3) and multiplying the first N equations at the left hand side with $[V_q \;\; V_p]^T$, one obtains

$$\begin{bmatrix} V_q^T(M-I)V_q & 0 & V_q^T b_T \\ V_p^T M V_q & V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \end{bmatrix}. \tag{2.7}$$

Here we have used $V_p^T V_q = 0_{p\times q}$, $V_q^T V_p = 0_{q\times p}$ and $V_q^T M V_p = 0_{q\times p}$ with the latter equality holding since $\mathcal{U}$ is an invariant subspace of $M(x(0),T)$. Remark that $b_T^* = b_T(x(0)^*, T^*)$ is an eigenvector of $M(x(0)^*, T^*)$ for the eigenvalue 1, and thus by assumption (2.1) a vector in the subspace $\mathcal{U}$ at $(x(0)^*, T^*)$. Therefore, $V_q^T b_T$ converges to zero as the iteration converges. We will put this term to zero. With this approximation, (2.7) reduces to the block triangular system

$$\begin{bmatrix} V_q^T(M-I)V_q & 0 & 0 \\ V_p^T M V_q & V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \end{bmatrix}. \tag{2.8}$$

Since $r_\sigma(V_q^T M V_q) = |\mu_{p+1}| < 1$ (lemma 2.10 in [107]), $V_q^T M V_q - I_q$ is nonsingular. Hence the matrix in (2.8) is nonsingular <u>iff</u> the low-dimensional submatrix

$$\begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \tag{2.9}$$

is nonsingular. Note that the eigenvalues of $V_p^T M V_p$ are precisely the eigenvalues μ_1, …, μ_p of M and that the structure of the Jordan decomposition for these eigenvalues is also preserved in the matrix (2.9). All information on the eigenvalues of M that can be involved in possible bifurcations is thus preserved in the low-dimensional matrix (2.9).

(2.8) can be solved by first computing $\Delta\bar{q}$ from

$$\left[V_q^T M V_q - I_q\right] \Delta\bar{q} = -V_q^T r \tag{2.10}$$

and then computing $\Delta\bar{p}$ and ΔT from

$$\begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T(r + MV_q\,\Delta\bar{q}) \\ s + c_s^T V_q \Delta\bar{q} \end{bmatrix}. \tag{2.11}$$

We can then compute $\Delta x(0) = V_q\Delta\bar{q} + V_p\Delta\bar{p}$. Note that this is already an approximation to the solution of (2.7) since we neglected the term $V_q^T b_T$, which is small in the neighbourhood of the limit cycle but exactly zero only at the limit cycle. This procedure is no real saving yet since we want to avoid the computation of the large matrix $V_q^T M V_q$. Since $r_\sigma(V_q^T M V_q) = \left|\mu_{p+1}\right| < \rho < 1$, (2.10) can be solved approximately by the Picard iteration scheme

$$\begin{cases} \Delta\bar{q}^{[0]} = 0, \\ \Delta\bar{q}^{[i]} = V_q^T M V_q \Delta\bar{q}^{[i-1]} + V_q^T r, \quad i = 1, \dots, l, \end{cases} \tag{2.12}$$

or equivalently,

$$\Delta\bar{q} = \Delta\bar{q}^{[l]} = \sum_{i=0}^{l-1} (V_q^T M V_q)^i \, V_q^T r.$$

In section 2.4, we will show that this scheme can be used to compute $\Delta q = V_q\Delta\bar{q}$ using just $l-1$ matrix–vector products with M and without the explicit construction of the basis V_q or the matrix $V_q^T M V_q$. Obviously, (2.12) corresponds to approximately solving (2.10) by multiplying the right hand side by the l terms Neumann series for $(V_q^T M V_q - I_q)^{-1}$. Hence the Newton–Picard procedure corresponds to applying the orthogonal coordinate transformation (2.6) and approximating the Jacobian matrix in (2.7) with the block triangular matrix

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1} (V_q^T M V_q)^i\right)^{-1} & 0 & 0 \\ V_p^T M V_q & V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix}.$$

The approximation improves as l is increased and as the limit cycle is approached. Except for highly nonnormal cases, $l = 1$ is enough to make the overall scheme convergent for a starting value close enough to $(x(0)^*, T^*)$ (see section 2.2.3).

In the right-hand side of 2.11, r and s are evaluated at $(x(0)^{(\nu)}, T^{(\nu)})$. Using Taylor's theorem, the right-hand side of (2.11) can be transformed to

$$-\begin{bmatrix} V_p^T r(x(0) + \Delta q, T) \\ s(x(0) + \Delta q, T) \end{bmatrix} + \text{h.o.t.}, \tag{2.13}$$

i.e., r and s evaluated at $(x(0)^{(\nu)} + \Delta q^{(\nu)}, T^{(\nu)})$ up to higher-order terms. Hence solving (2.7) by first solving (2.10) with the l steps of the Picard scheme (2.12) followed by (2.11) is essentially to be seen as a *Gauss–Seidel approach* since we first compute an approximation to $\Delta\bar{q}$ and then use the updated q to compute $\Delta\bar{p}$ and ΔT. We will denote this variant by $NPGS(l)$, for $\underline{\text{N}}$ewton–$\underline{\text{P}}$icard $\underline{\text{G}}$auss–$\underline{\text{S}}$eidel scheme with $\underline{l}$ Picard iteration steps. *Jacobi-type variants* are derived by putting the terms $V_p^T M V_q$ and $c_s^T V_q$ in (2.8) to 0, although these terms can (and often will) be large. We will further denote this variant by $NPJ(l)$, which stands for Newton–Picard Jacobi scheme with l Picard iteration steps. As we will show in chapter 4, our $NPJ(1)$ scheme is equivalent to the recursive projection method [107]. If M is normal and s satisfies $c_s^T V_q = 0$, the $NPJ(l)$ and $NPGS(l)$ methods are equivalent since in this case $\mathcal{U}^\perp$ is the invariant subspace containing all (generalized) eigenvectors for eigenvalues of M smaller than ρ in modulus. However, in general M is nonnormal and we expect better performance of the $NPGS(l)$ method. This will be confirmed by the test results presented in section 2.5.

2.2.2 An interpretation

We will now present an interpretation of the $NPGS(l)$ scheme which indicates that one iteration step with the Newton–Picard method can be seen as a succession of a time integration (Picard) step in the high-dimensional subspace $\mathcal{U}^\perp$ followed by a Newton-based single shooting step in the low-dimensional subspace $\mathcal{U}$.

Suppose we start from $(x(0), T)$. The first step of the Picard iteration (2.12) brings us to

$$\begin{aligned} x(0)^{[1]} &= x(0) + V_q \Delta\bar{q}^{[1]} \\ &= x(0) + V_q V_q^T r \\ &= x(0) + Q\left(\varphi(x(0), T) - x(0)\right) \end{aligned} \tag{2.14}$$

so

$$V_q^T x(0)^{[1]} = V_q^T \varphi(x(0), T).$$

This step corresponds to time integration restricted to the space $\mathcal{U}^\perp$. Restricted to this space, the limit cycle is attracting. Suppose we would perform a second time integration "Picard" step (2.14), now starting from $x(0)^{[1]}$. This step would result in

$$x(0)^{[2]} = x(0)^{[1]} + Q\left(\varphi(x(0)^{[1]}, T) - x(0)^{[1]}\right)$$

so

$$\begin{aligned} \Delta q^{[2]} = x(0)^{[2]} - x(0) &= \left(x(0)^{[2]} - x(0)^{[1]}\right) + \left(x(0)^{[1]} - x(0)\right) \\ &= Q\left(\varphi(x(0)^{[1]}, T) - x(0)^{[1]}\right) + \left(x(0)^{[1]} - x(0)\right) \\ &\approx Q\varphi(x(0), T) + QM\Delta q^{[1]} - \left(Qx(0)^{[1]} - x(0)^{[1]} + x(0)\right) \\ &\qquad \left(\text{Note } x(0)^{[1]} - x(0) = Q\left(x(0)^{[1]} - x(0)\right)\right) \\ &= Q\varphi(x(0), T) + QM\Delta q^{[1]} - Qx(0) \\ &= Qr(x(0), T) + QM\Delta q^{[1]} \\ &= Q(I + M)Qr(x(0), T). \end{aligned}$$

The term $Q(I + M)Qr(x(0), T)$ is precisely the result of two Picard steps with (2.12). In general, l steps of the Picard scheme (2.12) are equivalent up to higher order terms to l time integrations over a time interval of length T where we project the update into $\mathcal{U}^\perp$ after each time integration before starting the next time integration.

In section 2.2.1, we pointed out that the right hand side of (2.11) is equivalent to the P-projection of the residual of (2.2) in the point $(x(0) + \Delta q, T)$ up to higher order terms (see (2.13)). Without the higher order terms, (2.13) looks precisely like the linearization of the linearized single shooting system (2.2) in the point $(x(0) + \Delta q, T)$, but restricted to the low-dimensional space $\mathcal{U}$.

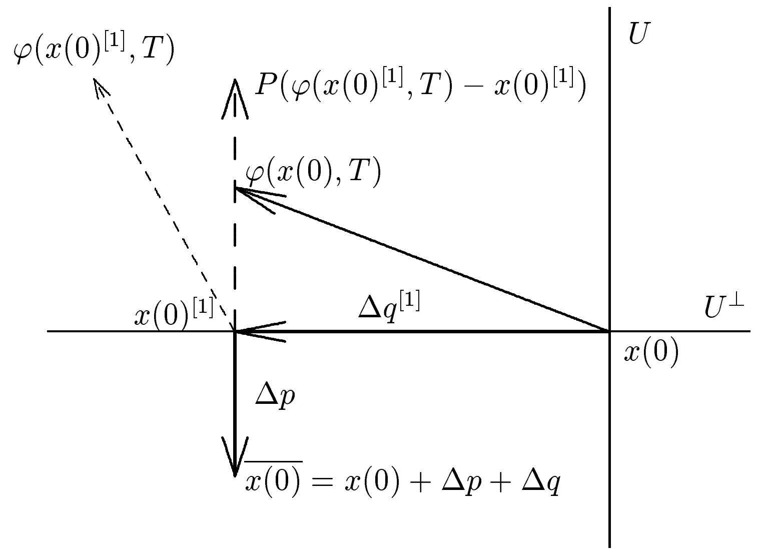


Figure 2.1: Graphical representation of $NPGS(1)$ as a combination of time integration in $\mathcal{U}^{\perp}$ followed by a single shooting step in $\mathcal{U}$.

Hence the $NPGS(l)$ method is very similar to l steps of time integration, restricting the updates to $\mathcal{U}^{\perp}$, followed by a Newton-based single shooting step in $\mathcal{U}$ starting from the end point of the time integrations. This is illustrated graphically in Figure 2.1 for the $NPGS(1)$ scheme. Instead of implementing (2.12) and (2.11) (which is done in our current code), one could also use l steps of (2.14) followed by solving the P-system

$$\begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r(x(0)+\Delta q, T) \\ s(x(0)+\Delta q, T) \end{bmatrix}.$$

This schema avoids the matrix–vector products with M in the Picard iteration scheme (2.12) and for the construction of the right-hand side of (2.11). However, as we shall discuss in section 2.4, we still need some matrix–vector products with M for the construction of the basis V_p.

2.2.3 An asymptotic convergence result

A convergence analysis for the recursive projection method or $NPJ(1)$ scheme under rather restrictive conditions is given in [107]. For our more general approach a full convergence analysis would require very careful discussion (including an account of norms of nonnormal matrices, etc.), which would probably have limited practical value. In this section, we will present a result on asymptotic convergence on which a more complete convergence account could be based.

A fixed point method

$$y^{(\nu+1)} = G(y^{(\nu)})$$

is asymptotically convergent to the solution y^* if

$$r_\sigma(G_y(y^*)) < 1$$

and the asymptotic convergence rate is $r_\sigma(G_y(y^*))$.

The $NPGS(l)$ and $NPJ(l)$ methods can both be rewritten as a method in the form

$$y^{(\nu+1)} = G(y^{(\nu)}) = y^{(\nu)} - H(y^{(\nu)})\,\tilde{r}(y^{(\nu)}).$$

Let

$$y = \begin{bmatrix} x(0) \\ T \end{bmatrix},\ \tilde{r}(y) = \begin{bmatrix} r(x(0),T) \\ s(x(0),T) \end{bmatrix} \text{ and } \tilde{V} = \begin{bmatrix} V_q & V_p & 0_{N\times 1} \\ 0_{1\times q} & 0_{1\times p} & 1 \end{bmatrix}. \tag{2.15}$$

Note that $\tilde{V}$ is an orthogonal matrix (i.e., $\tilde{V}^T\tilde{V} = \tilde{V}\tilde{V}^T = I$). For the $NPGS(l)$ scheme, we get

$$H(y) = H_{GS}(y) = \tilde{V} \begin{bmatrix} -\left(\sum_{i=0}^{l-1}(V_q^T M V_q)^i\right)^{-1} & 0 & 0 \\ V_p^T M V_q & V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix}^{-1} \tilde{V}^T$$

and for the $NPJ(l)$ scheme we get

$$H(y) = H_J(y) = \tilde{V} \begin{bmatrix} -\left(\sum_{i=0}^{l-1}(V_q^T M V_q)^i\right)^{-1} & 0 & 0 \\ 0 & V_p^T(M-I)V_p & V_p^T b_T \\ 0 & c_s^T V_p & d_{s,T} \end{bmatrix}^{-1} \tilde{V}^T.$$

Note that all quantities in these formulas depend on y.

The following lemma provides the spectral radius results for $NPGS(l)$ and $NPJ(l)$.

Lemma 2.2 *Let y^* be a solution of $\tilde{r}(y) = 0$. Suppose $\tilde{r}$ is twice differentiable and assumption 2.1 holds. Furthermore, suppose 1 is a single eigenvalue of M^* and ${c_s^*}^T b_T^* \neq 0$ (transversality condition for the phase condition). For both the $NPGS(l)$ and $NPJ(l)$ schemes, the spectral radius of the Jacobian of G at y^* satisfies*

$$r_\sigma(G_y(y^*)) = \left|\mu_{p+1}\right|^l < \rho^l < 1.$$

Proof. The proof holds for both $H = H_{GS}$ and $H = H_J$. Since 1 is a single eigenvalue of M^* and since ${c_s^*}^T b_T^* \neq 0$, it can be shown easily that the lower right block

$$\begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix}$$

is nonsingular at y^* so the functions $H_J(y)$ and $H_{GS}(y)$ are well defined in a neighbourhood of y^*.

Since $\tilde{r}(y^*) = 0$, we get

$$G_y(y^*) = I - H(y^*)\tilde{r}_y(y^*).$$

Since $\tilde{V}$ is an orthonormal matrix,

$$r_\sigma(G_y(y^*)) = r_\sigma(\tilde{V}(y^*)^T G_y(y^*) V(y^*)) = r_\sigma\left(I - (\tilde{V}^T H(y^*)\tilde{V})(\tilde{V}^T r_y(y^*)\tilde{V})\right).$$

Furthermore,

$$\tilde{V}^T H(y^*)\tilde{V} = \begin{bmatrix} -\sum_{i=0}^{l-1}(V_q^T M V_q)^i & 0 \\ * & \begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix}^{-1} \end{bmatrix},$$

$$\tilde{V}^T r_y(y^*)\tilde{V} = \begin{bmatrix} V_q^T(M-I)V_q & 0 & 0 \\ V_p^T M V_q & V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix}$$

and hence

$$\tilde{V}^T G_y(y^*)\tilde{V} = \begin{bmatrix} \left(V_q^T M V_q\right)^l & 0_{q\times(p+1)} \\ * & 0_{(p+1)\times(p+1)} \end{bmatrix}$$

where the values at the stars depend on the type of the scheme. It directly follows from assumption 2.1 that

$$r_\sigma\left(G_y(y^*)\right) = \left|\mu_{p+1}\right|^l < \rho^l < 1.$$

□

Corollary 2.3 *Under the conditions of lemma 2.2,*

1. *$NPGS(l)$ and $NPJ(l)$ schemes are asymptotically convergent in a neighbourhood of the solution if $\rho < 1$;*

2. *the asymptotic convergence rate of $NPGS(l)$ and $NPJ(l)$ is $\left|\mu_{p+1}\right|^l$.*

This is a rather restricted result. It only allows us to predict the asymptotic convergence rate. Furthermore, it assumes an accurate basis. Note that due to the nonnormality of M, the residual may increase initially for any fixed value of l and y may leave the domain of attraction of the solution y^*. As one would expect intuitively, the asymptotic convergence rate of the scheme is limited by its slowest component, the Picard scheme. In the next chapter, we shall discuss a more practical analysis that will allow us to control the convergence of the scheme during the iteration steps and to obtain a robust method.

2.3 Pseudo-arclength continuation

2.3.1 Derivation and convergence

In pseudo-arclength continuation, the parameter γ is allowed to vary and an additional equation, the parametrizing equation, is added to (2.2) (see section 1.5). A single shooting pseudo-arclength continuation method solves the nonlinear system

$$\begin{cases} r(x(0),T,\gamma) = \varphi(x(0),T,\gamma) - x(0) = 0, \\ s(x(0),T,\gamma) = 0, \\ n(x(0),T,\gamma;\eta) = 0 \end{cases} \tag{2.16}$$

for the unknowns $(x(0),T,\gamma)$. We make analogous assumptions with respect to the differentiability as in the previous section, i.e., φ, s and n are twice differentiable with respect to

$x(0)$, T and γ. For ease of implementation, the parametrizing equation $n(x(0),T,\gamma;\eta)=0$ should be a simple expression not involving time integration or integrals along the limit cycle. Examples of possible parametrizing equations are

$$\begin{aligned} n_1(x(0),T,\gamma;\eta) &= \theta_x \|x(0;\eta)-x(0;\eta_0)\|_2^2 + \theta_T\left(T(\eta)-T(\eta_0)\right)^2 + \\ &\quad \theta_\gamma\left(\gamma(\eta)-\gamma(\eta_0)\right)^2-(\eta-\eta_0)^2=0, \end{aligned} \tag{2.17}$$

where $(x(0;\eta_0),T(\eta_0),\gamma(\eta_0))$ is a known point of a periodic solution on the branch, or

$$n_2(x(0),T,\gamma;\eta)=\theta_x\left(x(0)-x_p(0)\right)^T x_s(0)+\theta_T\left(T-T_p\right)T_s+\theta_\gamma\left(\gamma-\gamma_p\right)\gamma_s=0, \tag{2.18}$$

where $(x_p(0),T_p,\gamma_p)$ is the point generated by the predictor and $(x_s(0),T_s(0),\gamma_s(0))$ is the normalized predictor direction. The latter condition defines a hypersurface through the predicted point and orthogonal to the predictor direction in some weighted scalar product. The parametrizing equation (2.17) has the slight disadvantage that it is a nonlinear condition. In our tests, we used (2.18).

If Newton's method is used to solve (2.16), we have to solve linear systems of the form

$$\begin{bmatrix} M-I & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \\ c_n^T & d_{n,T} & d_{n,\gamma} \end{bmatrix}\begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta\gamma \end{bmatrix} = -\begin{bmatrix} r(x(0),T,\gamma) \\ s(x(0),T,\gamma) \\ n(x(0),T,\gamma;\eta) \end{bmatrix} \tag{2.19}$$

repeatedly. Note that we continue to omit the superscript (ν) as in the previous section. With a good parametrizing equation, (2.19) will be nonsingular at hyperbolic periodic solutions, fold points and all bifurcation points that do not involve an (additional) $+1$ Floquet multiplier. We again make assumption 2.1, generalized to $y^*=(x(0)^*,T^*,\gamma^*)$.

Assumption 2.4 *Let $y^*=(x(0)^*,T^*,\gamma^*)$ denote an isolated solution to (2.16), and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M(y)=\frac{\partial\varphi}{\partial x(0)}(y)$ for $y\in\mathcal{B}$ and denote its eigenvalues by μ_i, $i=1,\ldots,N$. Assume that for all $y\in\mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho=\{|z|<\rho\},\quad 0<\rho<1 \tag{2.20}$$

and that no eigenvalue has modulus ρ; i.e., for all $y\in\mathcal{B}$

$$|\mu_1|\ge|\mu_2|\ge\cdots\ge|\mu_p|>\rho>|\mu_{p+1}|,\ldots,|\mu_N|.$$

Based on this assumption, a basis $[V_q\ \ V_p]$ is defined as before (2.5-2.6). In this new basis, (2.19) transforms to

$$\begin{bmatrix} V_q^TMV_q-I_q & 0 & 0 & V_q^Tb_\gamma \\ V_p^TMV_q & V_p^TMV_p-I_p & V_p^Tb_T & V_p^Tb_\gamma \\ c_s^TV_q & c_s^TV_p & d_{s,T} & d_{s,\gamma} \\ c_n^TV_q & c_n^TV_p & d_{n,T} & d_{n,\gamma} \end{bmatrix}\begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = -\begin{bmatrix} V_q^Tr \\ V_p^Tr \\ s \\ n \end{bmatrix}, \tag{2.21}$$

where we have put $V_q^Tb_T$ to zero as before. In general, b_γ does not lie close to or into the subspace $\mathcal{U}$ and we cannot neglect $V_q^Tb_\gamma$ without serious implications on the convergence of Newton's method. System (2.21) does not reduce to a block-triangular system. We will now study two different solution techniques for (2.21) that cope with this problem: the Sherman–Morrison formula (see, e.g., [44]) and a technique based on some of the ideas of Moore and Spence [83].

Sherman–Morrison formula The matrix in (2.21) is a rank-one update of a block triangular matrix and (2.21) can be rewritten as

$$(B + \xi e_{N+2}^T)\Delta u = t \tag{2.22}$$

with

$$B = \begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix}$$

and

$$\xi = \begin{bmatrix} V_q^T b_\gamma \\ 0_p \\ 0 \\ 0 \end{bmatrix}, \; e_{N+2} = \begin{bmatrix} 0_q \\ 0_p \\ 0 \\ 1 \end{bmatrix}, \; \Delta u = \begin{bmatrix} \Delta \bar{q} \\ \Delta \bar{p} \\ \Delta T \\ \Delta \gamma \end{bmatrix} \text{ and } t = -\begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}.$$

Provided both B and the matrix in (2.21) are nonsingular, (2.22) can be solved by first solving the systems

$$B u_r = t \text{ and } B u_\gamma = \xi \tag{2.23}$$

and then computing $\Delta\gamma$ from

$$\Delta\gamma = e_{N+2}^T \Delta u = \frac{e_{N+2}^T u_r}{1 + e_{N+2}^T u_\gamma} \tag{2.24}$$

and the other unknowns from

$$\Delta u = u_r - \Delta\gamma \, u_\gamma. \tag{2.25}$$

(2.23) can be solved approximately as before by first computing $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$ using the Picard iteration (2.12) (using b_γ instead of r for $\Delta\bar{q}_\gamma$), and then solving the remaining equations for the other unknowns.

If $B + \xi e_{N+2}^T$ is nonsingular then

$$1 + e_{N+2}^T u_\gamma = 1 + u_{\gamma,N+2} \neq 0$$

(where $u_{\gamma,N+2}$ denotes the last component of u_γ). Indeed,

$$B^{-1}(B + \xi e_{N+2}^T) = I + u_\gamma e_{N+2}^T = \begin{bmatrix} 1 & 0 & \cdots & 0 & u_{\gamma,1} \\ 0 & 1 & \cdots & 0 & u_{\gamma,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u_{\gamma,N+1} \\ 0 & 0 & \cdots & 0 & 1 + u_{\gamma,N+2} \end{bmatrix}$$

so $1 + e_{N+2}^T u_\gamma = 0$ would imply that $I + u_\gamma e_{N+2}^T$ and thus $B + \xi^T e_{N+2}$ is singular. Note that B is nonsingular <u>iff</u> its lower right block

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \tag{2.26}$$

is nonsingular (since $V_q^T M V_q - I_q$ is nonsingular by construction). However, the nonsingularity of (2.21) does not imply the nonsingularity of (2.26). This is a clear disadvantage of this approach. We used this approach in our early codes and produced many of the test results at the end of this section with these codes.

Moore–and–Spence-like approach Since $V_q^T M V_q - I_q$ is nonsingular by construction, the first blockrow of (2.21) can be transformed to

$$\Delta\bar{q} = -(V_q^T M V_q - I_q)^{-1}\left(V_q^T r + V_q^T b_\gamma \Delta\gamma\right). \tag{2.27}$$

Substituting this relation in the other equations of (2.21) results in

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T(b_\gamma - M V_q (V_q^T M V_q - I_q)^{-1} V_q^T b_\gamma) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} - c_s^T V_q (V_q^T M V_q - I_q)^{-1} V_q^T b_\gamma \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} - c_n^T V_q (V_q^T M V_q - I_q)^{-1} V_q^T b_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T(r - M V_q (V_q^T M V_q - I_q)^{-1} V_q^T r) \\ s - c_s^T V_q (V_q^T M V_q - I_q)^{-1} V_q^T r \\ n - c_n^T V_q (V_q^T M V_q - I_q)^{-1} V_q^T r \end{bmatrix}. \tag{2.28}$$

(2.27-2.28) are fully equivalent to the original system (2.21). We can thus solve (2.21) by first computing $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$ from

$$\left(V_q^T M V_q - I_q\right)\Delta\bar{q}_r = -V_q^T r \text{ and } \left(V_q^T M V_q - I_q\right)\Delta\bar{q}_\gamma = -V_q^T b_\gamma, \tag{2.29}$$

then solving

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T(b_\gamma + M V_q^T \Delta\bar{q}_\gamma) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} + c_s^T V_q \Delta\bar{q}_\gamma \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} + c_n^T V_q \Delta\bar{q}_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T(r + M V_q \Delta\bar{q}_r) \\ s + c_s^T V_q \Delta\bar{q}_r \\ n + c_n^T V_q \Delta\bar{q}_r \end{bmatrix} \tag{2.30}$$

for $\Delta\bar{p}$, ΔT and $\Delta\gamma$ and finally computing

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta\gamma\,\Delta\bar{q}_\gamma. \tag{2.31}$$

The two systems (2.29) are solved approximately using the Picard iteration (2.12). Note that the Q-systems are the same as in the Sherman–Morrison based approach except for the sign of one of the right hand sides. However, in the Sherman–Morrison approach we needed to solve two P-systems while in this approach there is only one P-system. Furthermore, the matrices of the P-systems are different. As can be seen from the derivation, the nonsingularity of (2.30) follows immediately from the nonsingularity of (2.21) (at least for the exact values of $\Delta\bar{q}_\gamma$). This is just a particular case of the following more general lemma.

Lemma 2.5 *Suppose $A_{1,1} \in \mathbb{R}^{q\times q}$ is a nonsingular matrix. Furthermore, let $A_{2,1} \in \mathbb{R}^{p\times q}$, $A_{2,2} \in \mathbb{R}^{p\times p}$, B_1, $C_1 \in \mathbb{R}^{q\times k}$, B_2, $C_2 \in \mathbb{R}^{p\times k}$ and $D \in \mathbb{R}^{k\times k}$. Let*

$$\tilde{A}_1 = \begin{bmatrix} A_{1,1} & 0 & B_1 \\ A_{2,1} & A_{2,2} & B_2 \\ C_1^T & C_2^T & D \end{bmatrix}$$

and

$$\tilde{A}_2 = \begin{bmatrix} A_{2,2} & B_2 - A_{2,1}A_{1,1}^{-1}B_1 \\ C_2^T & D - C_1^T A_{1,1}^{-1} B_1 \end{bmatrix}.$$

Then $\tilde{A}_1$ is nonsingular $\underline{\textit{iff}}$ $\tilde{A}_2$ is nonsingular.

Proof. Since A_{11} is a nonsingular matrix, one can apply one step of block Gauss elimination, resulting in the equality

$$\begin{bmatrix} I & 0 & 0 \\ -A_{21}A_{11}^{-1} & I & 0 \\ -C_1^T A_{11}^{-1} & 0 & I \end{bmatrix} \begin{bmatrix} A_{1,1} & 0 & B_1 \\ A_{2,1} & A_{2,2} & B_2 \\ C_1^T & C_2^T & D \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \quad B_1 \\ 0 & \\ 0 & \tilde{A}_2 \end{bmatrix}. \tag{2.32}$$

Since

$$\begin{bmatrix} I & 0 & 0 \\ -A_{21}A_{11}^{-1} & I & 0 \\ -C_1^T A_{11}^{-1} & 0 & I \end{bmatrix}$$

is a nonsingular matrix, $\tilde{A}_1$ is nonsingular <u>iff</u> the matrix in the right-hand side of (2.32) is nonsingular. Since A_{11} is nonsingular, the latter matrix is nonsingular <u>iff</u> $\tilde{A}_2$ is nonsingular. □

Note that the matrix in (2.30) is only guaranteed to be nonsingular if we solve (2.29) accurately. However, since every nonsingular matrix has a neighbourhood of nonsingular matrices, we can assume the latter matrix will still be nonsingular if good approximations to the solution of (2.29) are used. The matrix in (2.30) contains all information on bifurcations that is contained in the matrices in (2.21) and (2.19). For instance, it is easy to prove the following lemma using similar techniques as in the proof of lemma 2.5.

Lemma 2.6 *Suppose assumption 2.4 holds and the basis* $[V_q \quad V_p]$ *is defined as before. At the limit cycle*

$$b_\gamma \in \text{Range}\begin{bmatrix} M - I & b_T \end{bmatrix}$$
$$\underline{\textit{iff}}$$
$$V_p^T\left(b_\gamma - MV_q\left(V_q^T M V_q - I_q\right)^{-1} V_q^T b_\gamma\right) \in \text{Range}\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \end{bmatrix}.$$

Proof. By definition, $b_\gamma \in \text{Range}\begin{bmatrix} M - I & b_T \end{bmatrix}$ <u>iff</u> the system

$$\begin{bmatrix} M - I & b_T \end{bmatrix} \begin{bmatrix} v \\ h \end{bmatrix} = b_\gamma \tag{2.33}$$

with $v \in \mathbb{R}^N$ and $h \in \mathbb{R}$ has a solution. By premultiplying (2.33) with $[V_q \quad V_p]^T$ and by setting $\bar{v}_q = V_q^T v$, $\bar{v}_p = V_p^T v$, we obtain the equivalent system

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T \end{bmatrix} \begin{bmatrix} \bar{v}_q \\ \bar{v}_p \\ h \end{bmatrix} = \begin{bmatrix} V_q^T b_\gamma \\ V_p^T b_\gamma \end{bmatrix}. \tag{2.34}$$

Note that because of assumption 2.4 and since we are at the limit cycle, $V_q^T b_T = 0$. The matrix $V_q^T M V_q - I_q$ is a nonsingular matrix. By applying one step of block Gauss elimination, (2.34) is reduced to the system

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 \\ 0 & V_p^T M V_p - I_p & V_p^T b_T \end{bmatrix} \begin{bmatrix} \bar{v}_q \\ \bar{v}_p \\ h \end{bmatrix} = \begin{bmatrix} V_q^T b_\gamma \\ V_p^T\left(b_\gamma - MV_q\left(V_q^T M V_q - I_q\right)^{-1} V_q^T b_\gamma\right) \end{bmatrix}. \tag{2.35}$$

This system has a solution <u>iff</u>

$$\begin{aligned}&\text{(a) } V_q^T b_\gamma \in \text{Range}\left[V_q^T M V_q - I_q\right] \text{ and}\\ &\text{(b) } V_p^T \left(b_\gamma - M V_q \left(V_q^T M V_q - I_q\right)^{-1} V_q^T b_\gamma\right) \in \text{Range}\left[\begin{array}{cc} V_p^T M V_p - I_p & V_p^T b_T \end{array}\right].\end{aligned} \tag{2.36}$$

Since $V_q^T M V_q - I_q$ is nonsingular, condition (2.36a) is always satisfied. □

Lemma 2.6 allows us to distinguish between a pitchfork or transcritical bifurcation point and a fold point from the study of the low-dimensional system (2.30) instead of the high-dimensional system. Because of the greater robustness (since the nonsingularity of the low-dimensional system that has to be solved follows from the nonsingularity of the high-dimensional system) and because we can recover more information about the high-dimensional system, we used the Moore–and–Spence-like approach in more recent versions of the code.

Note that both approaches can be generalized to systems like (2.21) with more than one column of bordering. The Sherman–Morrison formula is generalized by the Sherman–Morrison–Woodbury formula [44]

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1}$$

(provided both A and $I + V^T A^{-1} U$ are nonsingular). The generalization of the Moore–and–Spence-like approach follows from the proof of lemma 2.5. In both cases, a k-column bordering will lead to $k+1$ different right-hand sides of the Q-system.

Since the above approaches are relevant in a continuation context using Newton–Picard iterations and since both of the above variants use the same approximation to the matrix in (2.21), we denote both of these variants by $CNP(l)$ (i.e., <u>C</u>ontinuation <u>N</u>ewton–<u>P</u>icard). Other variants of the Newton–Picard method for continuation—which we do not call $CNP(l)$ methods—can be derived by deleting the $(2,1)$, $(3,1)$ and $(4,1)$ and/or the $(1,4)$ blocks in (2.21). In all cases, the resulting system is block triangular or even block diagonal and only one Q-system needs to be solved at each step. The method based on neglecting the $(2,1)$, $(3,1)$ and $(4,1)$ blocks and using one step of the Picard iteration (2.12) is basically the equivalent to the continuation variant of the recursive projection method presented in section 7 of [107]. We will denote its extension with l Picard steps as the $CRP(l)$ method which stands for <u>C</u>ontinuation variant of the Recursive <u>P</u>rojection method using <u>l</u> Picard iteration steps. The similarity will be further discussed in chapter 4. The $CRP(l)$ method uses the Jacobian matrix approximation

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1}(V_q^T M V_q)^i\right)^{-1} & 0 & 0 & V_q^T b_\gamma \\ 0 & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ 0 & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ 0 & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix}. \tag{2.37}$$

We first solve the lower right block for the unknown $\Delta\bar{p}$, ΔT and $\Delta\gamma$ and then compute $\Delta\bar{q}$ from the Picard scheme

$$\begin{cases} \Delta\bar{q}^{[0]} = 0, \\ \Delta\bar{q}^{[i]} = V_q^T M V_q \Delta\bar{q}^{[i-1]} + V_q^T(r + b_\gamma\,\Delta\gamma),\ i = 1, \ldots, l, \end{cases} \tag{2.38}$$

or equivalently,

$$\Delta\bar{q} = \Delta\bar{q}^{[l]} = \sum_{i=0}^{l-1} (V_q^T M V_q)^i \, V_q^T (r + b_\gamma \, \Delta\gamma).$$

Since

$$V_q^T (r + b_\gamma \, \Delta\gamma) = V_q^T r(x(0) + \Delta p, T, \gamma + \Delta\gamma) + \text{h.o.t.}, \tag{2.39}$$

this is again essentially a Gauss–Seidel approach, where we first solve the P-system for $\Delta\bar{p}$, ΔT and $\Delta\gamma$ and then use the updated p and γ to compute Δq. Note that since we neglect the term $V_q^T b_T$, which is not zero away from the limit cycle, we cannot state that we also do use the updated period $T + \Delta T$ instead of T in (2.39). Most of our experiments with this method are done using (2.38).

Note that none of the blocks $(2,1)$, $(3,1)$, $(4,1)$ or $(1,4)$ is small in the general case, although special choices of the phase condition and pseudo-arclength condition can lead to small $(3,1)$ and $(4,1)$ blocks. Hence we expect a serious performance degradation from deleting one of the blocks, which is confirmed by our tests in section 2.5. The $CNP(l)$ method has a higher cost per iteration step than variants that omit some terms from the matrix in (2.21) to obtain a block triangular system since it requires the solution of two Q-systems at each step, but is clearly the most robust one and usually needs fewer iteration steps.

The proof of the asymptotic convergence result cannot be generalized to the pseudo-arclength continuation case. The presence of the term $V_q^T b_\gamma$ in (2.21) leads to a nonzero block in the upper right corner of the matrix $\tilde{V}^T G_y(y^*) \tilde{V}$ in the proof of lemma 2.2 and it is impossible to easily read off the eigenvalues of that matrix. In [107], a proof of asymptotic convergence is given for the continuation variant of their method, but the authors do not give all conditions and only state that it is possible to find such conditions and that these conditions are not very practical. From the analysis at the beginning of the next chapter, we will get a feeling of why each of the methods for pseudo-arclength continuation can fail and how we can easily cure this failure in our $CNP(l)$ method.

2.3.2 Computing the tangent vector

In section 1.5, we showed that the tangent vector to the limit cycle curve can be computed from a boundary value problem and used as a predictor in a predictor–corrector continuation code. In this section, we will show how to compute the tangent vector in the Newton–Picard method. Instead of solving the linear boundary value problem (1.56) using single shooting, we will start from (2.16). Note that both approaches result in exactly the same linear system to be solved.

(2.16) defines a curve of points if η is allowed to vary. Each point on the curve represents a periodic solution. By differentiating (2.16), we see that the normalized tangent vector $(\dot{x}(0), \dot{T}, \dot{\gamma}, \dot{\eta} = 1)$ is found from

$$\begin{bmatrix} M - I & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \\ c_n^T & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \dot{x}(0) \\ \dot{T} \\ \dot{\gamma} \end{bmatrix} = - \begin{bmatrix} 0_N \\ 0 \\ n_\eta \end{bmatrix}, \tag{2.40}$$

and is the result of a system with the same matrix as (2.19) but a different right hand side. Because of the special structure of the right hand side of (2.40), we are able to approximate the tangent vector at a low cost. Let $\dot{x}(0) = V_p\dot{\bar{p}} + V_q\dot{\bar{q}}$ with V_p and V_q defined as before. (2.40) can then be rewritten as

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \dot{\bar{q}} \\ \dot{\bar{p}} \\ \dot{T} \\ \dot{\gamma} \end{bmatrix} = - \begin{bmatrix} 0_q \\ 0_p \\ 0 \\ n_\eta \end{bmatrix}. \tag{2.41}$$

If this system is solved using either the Sherman–Morrison-based or Moore–and–Spence-like approach, there are no new Q-systems. One of the Q-systems,

$$\left(V_q^T M V_q - I_q\right) \dot{\bar{q}}_\gamma = \pm V_q^T b_\gamma$$

(where the $+$-sign is for the Sherman–Morrison based approach and the $-$-sign for the Moore–and–Spence-like approach) has already been solved approximately during the last Newton–Picard step. Note that the matrix and right hand side were computed at the end point of the last but one Newton–Picard step and not at the limit cycle. This solution can be further refined with some additional Picard iteration steps, but this will produce only small changes. Hence the further refinement is probably not very useful. The other Q-system in both approaches has a zero right hand side and thus a trivial solution. The matrix in the P-system(s) has also already been computed (if we do not recompute $\dot{\bar{q}}_\gamma$) and we only need to solve the P-system with a different right hand side. This is extremely cheap since the P-system is small and the matrix has already been factored. Note also that in the Sherman–Morrison approach, only one of the P-systems has a new right hand side.

Because of the particular combination of direct and iterative techniques in our approach, computing a good approximation to the tangent vector is as simple and cheap as if a direct method would be used to solve (2.19). In the latter case, one can compute a good approximation to the tangent vector by solving (2.40) using the last computed approximation to the matrix in this system. Hence one only needs to solve a system with a different right hand side but with a matrix that has already been used before. This is very cheap if a direct method is used. With many fully iterative techniques, it is much more expensive to compute the tangent vector. For instance, suppose the GMRES method would be used to solve (2.19) and (2.40) without exploiting the structure of the system, a new Krylov space needs to be built and little or no information on previous solves of (2.19) can be used again to reduce the computation time.

2.4 Implementation aspects

2.4.1 Matrix–vector products

First we will show how matrix–vector products with the matrix $M(x(0), T, \gamma)$ can be implemented efficiently. Note that

$$M(x(0), T, \gamma)\, v = \|v\|_2 \left(\left. \frac{\partial \varphi(x(0), T, \gamma)}{\partial x(0)} \right|_{(x(0), T, \gamma)} \frac{v}{\|v\|_2} \right) \tag{2.42}$$

is just a directional derivative of φ in the direction of v scaled with the norm of v. The matrix–vector product $M(x(0),T,\gamma)\,v$ can be computed by using the variational equations or finite differences. We will now discuss each of these methods in some more detail.

Variational equations. The matrix–vector product $M(x(0),T,\gamma)\,v$ can be computed as the result $v(T)$ of the initial value problem

$$\frac{dv}{dt} = \left.\frac{\partial f(x,\gamma)}{\partial x}\right|_{(\varphi(x(0),t,\gamma),\gamma)} v \text{ with } v(0) = v. \tag{2.43}$$

This is a linear system of N equations if the trajectory $\varphi(x(0),t,\gamma)$ is known in advance. Otherwise (2.43) has to be integrated in conjunction with (1.4). We already pointed out in section 1.4.1 that the variational equations (2.43) with a set of initial conditions can be solved very efficiently in conjunction with (1.4) if *LU* decomposition is used (or another direct method). Let us be a more precise at this point. Suppose (2.43) is solved with the k sets of initial conditions $v_1(0), \ldots, v_k(0)$ in conjunction with (1.4) using the backward Euler time integration scheme. At time step $i+1$, we have to solve the nonlinear system

$$\left\{\begin{array}{l} x^{(i+1)} - \Delta t\, f(x^{(i+1)},\gamma) = x^{(i)}, \\ \bigl(I - \Delta t\, f_x(x^{(i+1)},\gamma)\bigr)\, v_j^{(i+1)} = v_j^{(i)}, \quad j = 1,\ldots,k. \end{array}\right. \tag{2.44}$$

The first N equations in this system do not involve the unknowns $v_j^{(i+1)}$ and are nonlinear in $x^{(i+1)}$. We can first solve $x^{(i+1)}$ from this system using Newton's method. The remaining k sets of N equations are linear in $v_j^{(i+1)}$ and all involve the same matrix $I - \Delta t\, f_x(x^{(i+1)},\gamma)$, so we need to factor this matrix only once and solve the system with k different right hand sides to compute all vectors $v_j^{(i+1)}$, $j = 1,\ldots,k$. Note that we have already computed the *LU* decomposition of the matrix $I - \Delta t\, f_x(\tilde{x}^{(i+1)},\gamma)$, where $\tilde{x}^{(i+1)}$ is the last Newton iterate at which we evaluated f_x. This matrix is very close to $I - \Delta t\, f_x(x^{(i+1)},\gamma)$. We can exploit this by solving

$$\bigl(I - \Delta t\, f_x(x^{(i+1)},\gamma)\bigr)\, v_j^{(i+1)} = v_j^{(i)} \tag{2.45}$$

with the iterative process

$$\left\{\begin{array}{l} v_j^{(i+1)[0]} = 0, \\ \bigl(I - \Delta t\, f_x(\tilde{x}^{(i+1)},\gamma)\bigr)\, \Delta v_j^{(i+1)[\nu]} = v_j^{(i)} - \bigl(I - \Delta t\, f_x(x^{(i+1)},\gamma)\bigr)\, v_j^{(i+1)[\nu]}, \\ v_j^{(i+1)[\nu+1]} = v_j^{(i+1)[\nu]} + \Delta v_j^{(i+1)[\nu]}, \quad \nu = 0,\ldots. \end{array}\right. \tag{2.46}$$

This scheme requires the computation of $I - \Delta t\, f_x(x^{(i+1)},\gamma)$ or matrix–vector products with that matrix, but does not require its factorization. It converges linearly. Note that (2.46) actually corresponds to the *iterative improvement* technique discussed in [44], p. 74–76. The use of the scheme (2.46) is interesting if the variational equations have to be solved with few right hand sides, i.e., if $k \times$ number of Newton steps in (2.46) $\times$ cost of one iteration of (2.46) is lower than the cost of solving (2.45) with k different right hand sides. The discussion presented above was for the backward Euler timestepping scheme, but it is clear that the results hold for many implicit time integration schemes. Note that we only get a large reduction in computation cost if (1.4) can be solved together with (2.43) and if all the vectors v are known at that time. Otherwise we can obtain a similar gain by storing the computed trajectory and the factorization of all matrices $I - \Delta t\, f_x(x^{(i+1)}, \gamma)$ or $I - \Delta t\, f_x(\tilde{x}^{(i+1)}, \gamma)$. This requires however a large amount of memory. If the initial conditions for (2.43) are not known at the moment of integration of (1.4) and if we choose not to store the trajectory and matrix factorizations, we have to integrate (1.4) and (2.43) again for each vector v. If (2.46) is used, the computation of a matrix–vector product $M(x(0)T, \gamma)\, v$ will take approximately the same time as the integration of the nonlinear system (1.4). Note that if Krylov methods or other iterative techniques are use to solve the linear systems, we can still gain from integrating (1.4) in conjunction with (2.43) for all right-hand sides or from storing the trajectory $\varphi(x(0), t, \gamma)$, but the gains will be less since Krylov methods cannot exploit multiple right hand sides very well. However, we can often save on the construction of the preconditioner and it may be worth to store the preconditioner at each time step for later use if not all integrations of (2.43) can occur simultaneously.

Finite differences. The second option for the computation of the matrix–vector product (2.42) is numerical differentiation. A first order approximation to $M(x(0), T, \gamma)\, v$ can be computed using

$$M(x(0), T, \gamma)\, v \approx \frac{\varphi(x(0) + h\, v, T, \gamma) - \varphi(x(0), T, \gamma)}{h}. \tag{2.47}$$

This formula requires one additional integration of (1.4) since $\varphi(x(0), T, \gamma)$ is needed anyway for the construction of the right-hand side of the Newton system. A higher accuracy can be obtained by the use of the second order formula

$$M(x(0), T, \gamma)\, v \approx \frac{\varphi(x(0) + h\, v, T, \gamma) - \varphi(x(0) - h\, v, T, \gamma)}{2h}, \tag{2.48}$$

but this formula requires two integrations of (1.4). The choice of the stepsize h is very difficult. If h is too large, the truncation error of (2.47) or (2.48) gets too large. This can often be cured by the use of a higher order formula or Richardson extrapolation (see, e.g., [78, 85]). On the other hand, if the stepsize is too small, catastrophic cancellation occurs. The roundoff error can be reduced by a more accurate evaluation of φ. The stepsize must be chosen very carefully to balance the truncation error and the roundoff errors.

There is also another problem with the finite difference approach. We derived the Newton–Picard method using the continuous time integration operator $\varphi(x(0), T, \gamma)$. In practise, we only have a numerical approximation $\tilde{\varphi}(x(0), T, \gamma)$ and we should use derivatives of $\tilde{\varphi}$ instead of φ. Since $\tilde{\varphi}(x(0), T, \gamma)$ and $\varphi(x(0), T, \gamma)$ are very close to each other, we usually do not need to take this point into account, although a really rigorous derivation and convergence proof of the method should be based on the approximation $\tilde{\varphi}$. For instance, it does not really matter whether the numerical integration of (2.43) really computes the directional derivative of $\tilde{\varphi}$ or an approximation to the directional derivative of φ. However, in finite difference formulae, it is often important to assure that the same approximation $\tilde{\varphi}$ is used for all time integrations, i.e., the same time grid and, for a variable order method, the same order for corresponding time steps of related integrations. Otherwise the matrix–vector products may be very inaccurate and the Newton–Picard method may fail. This observation is also made in [55].

One of the main advantages of finite differences over the variational equations is the fact that we do not need to make many changes to the time integration code. We need only assure that the above restrictions on time steps and order sequence are met for related integrations. The variational equations require more changes and require at least a routine to compute the matrix–vector products $f_x(x, \gamma)\, v$ accurately. It is sometimes hard to compute these matrix–vector products exactly, e.g., if the PDE also contains some hyperbolic terms that are discretized with upwind schemes. These schemes often lead to complex formulas for the derivatives. However, if $f_x(x, \gamma)\, v$ can be computed analytically, the variational equations are the most robust approach.

Numerical differentiation is often expensive. One matrix–vector product requires at least one time integration of the nonlinear system (1.4), but accurate and robust methods will usually require more time integrations. On the other hand, if direct methods are used, the variational equations can sometimes be solved at very little extra cost. Even in the worst-case scenario, using iterative methods to solve all linear systems that arise in the time integration method and recomputing $\varphi(x(0), T, \gamma)$ for each set of N initial conditions of (2.43), one matrix–vector product usually costs less than two time integrations of (1.4).

Because of the higher robustness and performance, we advise the use of the variational equations if $f_x(x, \gamma)\, v$ can be computed easily and the time integration code can be changed easily for the computation of the variational equations. In all other cases, numerical differentiation of φ should be considered.

2.4.2 Other derivatives

From (1.9) we know

$$\left.\frac{\partial \varphi(x(0), T, \gamma)}{\partial T}\right|_{(x(0),T,\gamma)} = f(\varphi(x(0), T, \gamma)).$$

Hence the computation of b_T is very cheap since $\varphi(x(0), T, \gamma)$ has already been computed during the evaluation of the residual of (2.2) or (2.16). Note that this formula computes the derivatives of the exact time integration operator and that $f(\tilde{\varphi}(x(0), T, \gamma))$ is not necessarily the derivative of $\tilde{\varphi}(x(0), T, \gamma)$ with respect to T. However, this approximation appears to be good enough for our method.

The computation of b_γ in (2.19) can be done using variational equations or finite differences. In the variational equation approach, b_γ is computed as the solution at time T of the initial value problem

$$\frac{dw}{dt} = \left.\frac{\partial f(x, \gamma)}{\partial x}\right|_{(\varphi(x(0),t,\gamma),\gamma)} w + \left.\frac{\partial f(x, \gamma)}{\partial \gamma}\right|_{(\varphi(x(0),t,\gamma),\gamma)} \text{ with } w(0) = 0. \quad (2.49)$$

This expression can be derived by differentiating (1.4) with respect to γ, changing the order of the derivatives with respect to time t and γ, setting $w(t) = \partial x(t)/\partial \gamma$ and noting that $\varphi(x(0), 0, \gamma) = x(0)$ and hence $w(0) = 0$. All remarks on the variational equations and the finite differences approach of section 2.4.1 also hold for the computation of b_γ. The variational equation approach (2.49) usually requires more changes to the time integration code than the finite difference approach, but is more robust and usually faster if $f_x\, v$ and f_γ can be computed analytically.

2.4.3 The Picard scheme

We want to avoid the explicit computation of the matrix V_q since computing and storing V_q is expensive. In our schemes, we never need the low-dimensional update $\Delta\bar{q}$, but only $\Delta q = V_q \Delta\bar{q}$. Hence we should adapt the Picard iteration scheme (2.12) to compute the N-dimensional vectors Δq directly and avoid the use of V_q. This is done by premultiplying each of the steps in (2.12) with V_q. This results in

$$\left\{ \begin{array}{rcl} \Delta q^{[0]} & = & 0, \\ \Delta q^{[i]} & = & V_q V_q^T M \Delta q^{[i-1]} + V_q V_q^T r \\ & = & Q M \Delta q^{[i-1]} + Q r, \quad i = 1, \ldots, l, \\ \Delta q & = & \Delta q^{[l]}. \end{array} \right. \tag{2.50}$$

Since

$$Q = V_q V_q^T = I - V_p V_p^T,$$

we can evaluate (2.50) without first computing V_q. The computation of $\Delta q^{[1]}$ in (2.50) requires no new time integrations. Each of the following steps requires the computation of $M\Delta q^{[i-1]}$. The time integration for this matrix–vector product cannot be done simultaneously with the residual computation of (2.2) or (2.16) since the vector $\Delta q^{[i-1]}$ is the result of the previous Picard iteration step and depends on the residual.

The projection $w = (I - V_p V_p^T)v$ can be computed as

$$w = v - \left(V_p \left(V_p^T v\right)\right), \tag{2.51}$$

which corresponds to the Gramm-Schmidt method without normalization, or by using the scheme

```
$w \leftarrow v$
**for** $i = 1$ **to** $p$ **do**
    $w \leftarrow w - \langle V_p[i], w \rangle V_p[i]$
**end,**
```
(2.52)

which corresponds to the modified Gramm-Schmidt algorithm without normalization. The latter option is numerically more stable, although we have never noticed problems with (2.51) in our tests.

2.4.4 Computation of the projectors

The most important aspect of the Newton–Picard algorithms is the efficient calculation and repeated updating of the low-dimensional subspace $\mathcal{U}$. Recall that $\mathcal{U}$ is defined to be the space spanned by the (generalized) eigenvectors of M corresponding to Floquet multipliers of modulus greater than ρ. Since we are only interested in the dominant eigenvalues and since we are able to compute matrix–vector products with the matrix M, it is natural to use subspace iteration to compute the dominant eigenvalues and eigenvectors. This is also proposed in [56, 57, 58] and [107]. In order to keep the number of (expensive) matrix–vector multiplications low, we will use the most sophisticated version of this algorithm, namely, subspace iteration with projection after each iteration and locking (deflation) (a variant of algorithm 5.4 in [101] with $iter = 1$). Since the details of the locking process are quite technical and would obscure the essential idea of the method, we first outline the procedure ignoring the locking process.

Subspace iteration with projection. Let us first discuss the subspace iteration process without locking but with projection. During the subspace iterations, we keep M fixed. We also assume p is known. We will use p_e additional vectors to accelerate the convergence and to aid the detection of eigenvalues leaving C_ρ. Let

$$V^{[0]} = \left[\begin{array}{ccc} v_1^{[0]} & \cdots & v_p^{[0]} \end{array}\right]$$

be an initial guess for an orthonormal basis for $\mathcal{U}$. We extend this basis to the orthonormal basis

$$V_e^{[0]} = \left[\begin{array}{ccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p+p_e}^{[0]} \end{array}\right]$$

where $v_{p+1}^{[0]}, \ldots, v_{p+p_e}^{[0]}$ are guesses for the next p_e dominant Schur vectors. We rearrange the order of operations in algorithm 5.3 in [101] to further reduce the number of matrix–vector operations and obtain the following algorithm:

```
Algorithm 2.1 Subspace iteration with projection
    Input:
        $V_e^{[0]} = \left[\begin{array}{ccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p+p_e}^{[0]} \end{array}\right]$
        routine to compute $Mv$.
    Output:
        $V_e$, where $\mathrm{span}(V_e[1{:}p])$ is a good approximation for $\mathcal{U}$;
        $W = MV_e$ and $S = V_e^T W$.
    Begin
        $W = V_e^{[0]}$
        Repeat
            $V_e \leftarrow \mathrm{Ortho}(W)$ (Orthonormalize the column vectors of $W$ and store the
            result in $V_e$.)
            Compute $W = MV_e$.
            Compute $U = V_e^T M V_e = V_e^T W$
            Compute the Schur decomposition $UY = YS$ of $U$ and order the Schur
            vectors $Y = \left[\begin{array}{ccc} y_1 & \cdots & y_{p+p_e} \end{array}\right]$ in decreasing order of the modulus of the
            corresponding eigenvalue.
            $V_e \leftarrow V_e Y$, $W \leftarrow WY$.
        until convergence of the first $p$ vectors
    End
```

The matrix $V_p^T M V_p$ in (2.11) and (2.21) is easily recovered from the last iteration step of the algorithm as the upper left $p \times p$ block of the matrix S. The convergence properties for algorithm 2.1 are proven in [101]. The convergence factor for a simple eigenvalue μ_i is $|\mu_{p+p_e+1}/\mu_i|$, where μ_{p+p_e+1} is the next most dominant eigenvalue of M.

Subspace iteration with projection and locking. Note that the more dominant eigenvalues converge faster than the other ones in the subspace iteration method with projection. We exploit this feature by using an adapted version of the subspace iteration algorithm with projection and locking as described in Algorithm 5.4 in [101]. Locking was first introduced in [59]. The basic idea is to not further update vectors that have converged already. This leads to the following algorithm:

Algorithm 2.2 *Subspace iteration with projection and locking.*

```
Input:
    $V_e^{[0]} = \left[ v_1^{[0]} \ \cdots \ v_p^{[0]} \ \cdots \ v_{p+p_e}^{[0]} \right]$
    routine to compute $Mv$.
Output:
    $V_e$, where $\mathrm{Span}(V_e[1{:}p])$ is a good approximation for $\mathcal{U}$;
    $W = MV_e$ and $S = V_e^T W$.
begin
    $p_{eff} \leftarrow 0$
    $W = V_e^{[0]}$
    repeat
        $V_e[p_{eff}+1{:}p+p_e] \leftarrow W[p_{eff}+1{:}p+p_e]$
        Orthonormalize the column vectors of $V_e$ starting at column $p_{eff}+1$.
        Compute $W[p_{eff}+1{:}p+p_e] = MV_e[p_{eff}+1{:}p+p_e]$.
        Compute $U = V_e^T M V_e = V_e^T W$
        Compute the Schur decomposition $UY = YS$ of $U$, order the Schur vectors
        according to decreasing modulus of the corresponding eigenvalue.
        $V_e \leftarrow V_e Y$, $W \leftarrow WY$.
        Determine the number $p_{eff}$ of accurate vectors in $V_e$.
    until $p_{eff} = p$
end
```

In [101] locked vectors are not updated anymore. U is computed as $U = V_e[p_{eff}+1{:}p_e]^T W[p_{eff}+1{:}p+p_e]$ and the step $V_e \leftarrow V_e Y$, $W \leftarrow WY$ becomes $V_e[p_{eff}+1{:}p] \leftarrow V_e[p_{eff}+1{:}p]Y$, $W[p_{eff}+1{:}p] \leftarrow W[p_{eff}+1{:}p]Y$. We have chosen for another approach and allow small updates to already locked vectors based on information gained from the new matrix–vector products of the other basis vectors. This is similar to the procedure used in LOPSI [59]. This approach allows eigenvalues to be reordered in the Schur decomposition and to take into account the (unlikely) case that an eigenvalue is initially underestimated, but converges to a value that is larger than a locked eigenvalue. In this case, the already locked vectors have to change place in the Schur decomposition and p_{eff} is allowed to decrease.

Our convergence criterion is based on [110] and requires no new matrix–vector products with M. Our aim is to compute an invariant subspace $\mathrm{Span}(V_p)$ of M and to ensure that the term $V_q^T M V_p \Delta\bar{p}$ which is assumed 0 in (2.7) and (2.21) remains small. Let $V_k = V_e[1{:}k]$, $W_k = W[1{:}k]$ and $S_k = S[1{:}k, 1{:}k]$. Suppose $S[k+1,k] = 0$, i.e., the (k,k) element of S is not the upper left element of a 2×2-block. We will lock the first k vectors if

$$\zeta_k = \max_{\substack{\|\bar{p}_k\|_2=1 \\ \bar{p}_k \in \mathbb{R}^k}} \left\| \left(I - V_k V_k^T\right) M V_k \bar{p}_k \right\|_2$$

decreases below a user-determined threshold ε_{sub}. ζ_k is the largest singular value of $(I - V_k V_k^T) M V_k$. Let

$$Z_k = (I - V_k V_k^T) M V_k = W_k - V_k V_k^T W_k = W_k - V_k S_k \tag{2.53}$$

Z_k can be computed without new matrix–vector products with M. Note that (2.53) is only valid if $S[k+1,k] = 0$. Under this condition, we also have

$$Z_k = Z_p[1{:}k]. \tag{2.54}$$

p_{eff} is determined as the largest number for which $\|Z_k\|_2 = \sigma_{\max}(Z_k) < \varepsilon_{sub}\ \forall k \in \{i \mid i \leq p_{eff} \text{ and } S[i+1,i] = 0\}$. ε_{sub} is the user-determined threshold for the basis accuracy. Z_k is a $N \times k$-matrix and we wish to avoid doing many computations with it. Furthermore, we wish to determine $\sigma_{\max}(Z_k)$ for different values of k. Therefore we use the fact that $\sigma_{\max}(Z_k)$ is the square root of the largest eigenvalue of the $k \times k$-matrix $Z_k^T Z_k$. Note that

$$Z_k^T Z_k = \left(W_k^T - S_k^T V_k^T\right)\left(W_k - V_k S_k\right) = W_k^T W_k - S_k^T S_k$$

If we first compute

$$A_1 = W^T W,$$

then

$$Z_k^T Z_k = A_1[1{:}k, 1{:}k] - S_k^T S_k$$

and

$$\zeta_k = \sigma_{\max}(Z_k) = \sqrt{\lambda_{\max}(Z_k^T Z_k)}$$

can be computed efficiently for different values of k. Note that computing the singular values of a matrix A by first explicitly computing the matrix $A^T A$ and then computing its eigenvalues is not a very stable algorithm. However, the results are good enough for our purpose and the method does not require the construction of the matrix Z_k. An alternative to this method consists of first computing Z_p, then orthonormalize its columns by computing the QR-decomposition

$$Z_p = \tilde{Z}_p R_p, \quad \tilde{Z}_p \in \mathbb{R}^{N \times p}, \quad R_p \in \mathbb{R}^{p \times p}$$

with R_p an upper triangular matrix and then computing

$$\sigma_{\max}(Z_p) = \sigma_{\max}(R_p).$$

If $S[k+1,k] = 0$, (2.54) holds, $Z_k = \tilde{Z}_p[1{:}k] R_p[1{:}k, 1{:}k]$ is the QR-decomposition of Z_k, and

$$\sigma_{\max}(Z_k) = \sigma_{\max}(R_p[1{:}k, 1{:}k]).$$

The amount of work involved with both approaches is roughly the same. However, the second approach needs the explicit computation of the matrix Z_p and hence requires more storage space.

For small values of l and if a good starting value is available, one subspace iteration step per Newton–Picard step is usually sufficient. For large values of l and small values of p_e, or before the first Newton–Picard steps, more subspace iteration steps are needed. Locking then prevents updating almost converged vectors and is especially useful if some of the Floquet multipliers are large. In the early versions of our code, we kept converged eigenvalues locked between successive Newton–Picard steps once the Newton–Picard procedure had sufficiently converged and some eigenvalues of M did not change much from step to step. This saves some computing time. However, this strategy should be used with care, since preliminary locking of vectors and failing to unlock them again can lead to slow convergence or divergence of the algorithm. Furthermore, there are theoretical objections against this procedure, since M changes between two Newton–Picard iteration steps and W is no longer exactly equal to MV for the locked columns. In later versions of our implementation (based on the methods proposed in the next chapter), we left the user the option to disable this feature since keeping eigenvalues locked from one iteration step to another has a negative influence on the robustness of the method. In any case, all vectors are unlocked to compute accurate values for the Floquet multipliers after the final Newton–Picard step.

The initial basis. If a branch of periodic solutions is computed, the final basis for the periodic solution at the previous continuation point can be used as the initial guess for the basis $V_e^{[0]}$. To improve the robustness of the calculation of the Floquet multipliers, components of $V_e^{[0]}$ are randomly perturbed at the start of each new continuation step. This helps to assure that new eigenvectors can enter the basis rapidly and that all eigenvalues outside of the disk C_ρ are computed.

For the first point on a branch we can either start with random vectors and do some subspace iterations without a Newton–Picard step or derive starting values from the bifurcation point where the branch originates from. For example, at a Hopf point with critical eigenvalues $\pm i\omega$, the Floquet multipliers of the originating periodic orbit are given by

$$\mu_i = e^{\frac{2\pi\lambda_i}{\omega}}$$

where λ_i are the eigenvalues of f_x at the Hopf point. The initial basis is then given by the Schur vectors corresponding to the p rightmost eigenvalues of f_x. These eigenvalues and the corresponding basis can be computed using appropriate iterative techniques, see, e.g., [81].

Basis size. To determine the basis size we try to satisfy

$$|\mu_1| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}| \geq \cdots \geq |\mu_{p+p_e}|$$

with $p_e \geq 2$ to ensure that μ_{p+1} converges (since μ_{p+1} can be a complex or multiple eigenvalue). Since the convergence factor of a simple eigenvalue μ_i is $|\mu_{p+p_e+1}/\mu_i|$, larger values of p_e greatly improve the convergence speed of the p dominant eigenvalues in the subspace iteration procedure.

To assure early detection of growth of the basis, we use a weaker threshold in the convergence criterion for the decision on adding and removing basis vectors than in the criterion for the locking of vectors. Suppose $\mu_1, \ldots, \mu_i$ are accurate enough according to this criterion and suppose n_{vec} is the current number of vectors in the basis. Let p_{large} be the number of eigenvalues in the set $\{\mu_1, \ldots, \mu_i\}$ that are larger than ρ and p_{small} be the number of eigenvalues in that set that are smaller than $r_{hist}\rho$. $r_{hist} < 1$ adds some hysteresis to the criterion to avoid that we quickly add again a vector that has been removed or vice versa. We typically use values between 0.8 and 0.9 in our tests. If $p_{large} + p_e > n_{vec}$, vectors have to be added to the basis (since $p_{large} \leq$ the unknown value of p). Suppose $p_{small} > 0$ and μ_k is the largest eigenvalue in the set $\{\mu_1, \ldots, \mu_i\}$ that is smaller than $r_{hist}\rho$, then vectors should be removed if $(k-1) + p_e < n_{vec}$. This criterion is used in our multiple shooting code. In our (older) single shooting code we used a different criterion and only removed vectors if $p_{small} > p_e$. This is a too conservative criterion and often left too much vectors in the basis, but we mention it here since some of the test results were produced with it.

The combination of the use of extra vectors, random perturbations of the start values and our criterion for adding and removing vectors proved to be reliable in our testcases. We never noticed the problem encountered in the second test example of [107], where a Hopf bifurcation point on the computed branch is detected too late. It is also much simpler than criteria based on monitoring the convergence speed as in [56, 107], since slow convergence can also be caused by an inaccurate basis or bad starting values. The extra cost of the computations with the p_e extra vectors is largely compensated by the higher convergence speed. As we will see in the next chapter, the basis accuracy has a large influence on the performance and robustness of the method and it is important to assure a good basis quality.

2.4.5 The implementation of the $NPGS(l)$ algorithm

We will now sketch the $NPGS(l)$ algorithm as it was implemented to produce some of the test results reported in section 2.5.

Algorithm 2.3 *$NPGS(l)$ and subspace iteration with projection and locking for computing a periodic solution.*

```
    Input:
        Starting value $x^{(0)}(0)$, $T^{(0)}$.
        Starting basis $V_e = \left[ \begin{array}{ccc} v_1 & \cdots & v_{p+p_e} \end{array} \right]$
        Routine to compute $M(x(0),T)\,v$.
    Output:
        FAILURE or $x^*(0)$, $T^*$, final basis $V_e$.
    begin
        $\nu \leftarrow 0$;
        $p_{eff} \leftarrow 0$;
        $W \leftarrow V_e$;
        repeat
            Compute $\varphi \leftarrow \varphi(x(0),T)$; $r \leftarrow \varphi - x(0)$;
            /* Basis computation. */
            for $i = 1$ to $\nu_{sub}$ do
                $V_e[p_{eff} + 1{:}p + p_e] \leftarrow W[p_{eff} + 1{:}p + p_e]$
                Orthonormalize the column vectors of $V_e$ starting at column $p_{eff} + 1$.
                Compute $W[p_{eff} + 1{:}p + p_e] = MV_e[p_{eff} + 1{:}p + p_e]$.
                Compute $U = V_e^T M V_e = V_e^T W$
                Compute the Schur decomposition $UY = YS$ of $U$, order the Schur
                vectors according to decreasing modulus of the corresponding eigen-
                value.
                $V_e \leftarrow V_e Y$, $W \leftarrow WY$.
                Determine the number $p_{eff}$ of accurate vectors in $V_e$.
            end
            /* Picard iterations. */
            $\Delta q \leftarrow \left(I - V_e V_e^T\right) r$
            for $i = 2$ to $l$ do
                $\Delta q \leftarrow \left(I - V_e V_e^T\right) (M\,\Delta q + r)$
            end
            /* Newton correction. */
            Solve
                $\begin{bmatrix} S - I_{p+p_e} & V_e^T f(\varphi) \\ c_s(x(0),T)^T & d_{s,T}(x(0),T) \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = \begin{bmatrix} V_e^T(r + M(x(0),T)\,\Delta q) \\ s(x(0),T) + c_s(x(0),T)^T \Delta q \end{bmatrix}$.
            /* Update the solution. */
            $x(0) \leftarrow x(0) + \Delta q + V_e \Delta\bar{p}$
            $T \leftarrow T + \Delta T$
            $\nu \leftarrow \nu + 1$
            Decide if $p_{eff}$ should be put back to 0.
        until $(\nu \geq \nu_{\max})$ or convergence
        if ( no convergence ) then return FAILURE
        else $x^*(0) \leftarrow x(0)$; $T^* \leftarrow T$.
    end
```

Note that there are some peculiarities in this algorithm. We use all available vectors for the V_p projector in the Newton–Picard algorithm and not only the first p. This is motivated by the fact that the extra vectors usually have a reasonable accuracy and may accelerate the convergence of the Picard scheme, although it should better not be done in an implementation that has robustness as its prime goal instead of efficiency. Furthermore, as noted before, vectors are kept locked between iteration steps. We do a fixed number of subspace steps at the beginning of each Newton–Picard step. In our tests, we generally used $\nu_{sub} = 1$. If the start basis only contains random vectors, we first do some subspace iterations until the basis has sufficiently converged and jump into algorithm 2.3 after the basis computations. Here we do not perform a predetermined number of iterations, but stop the iterations as soon as we are sure that all basis vectors for eigenvalues larger than ρ have been computed with the required accuracy. After the computation of the periodic solution, our code has the option to do some more subspace

method	integration of (1.4)	matrix–vector products with M	computation of b_γ
$NPGS(l)$	1	$\nu_{sub}(p+p_e)+l-1$	0
$NPJ(l)$	1	$\nu_{sub}(p+p_e)+l$	0
$CNP(l)$	1	$\nu_{sub}(p+p_e)+2l$	1
$CRP(l)$	1	$\nu_{sub}(p+p_e)+l-1$	1
full Newton, γ fixed	1	N	0
full Newton, pseudo-arclength	1	N	1

Table 2.1: Work account for the Newton–Picard methods and classical single shooting using full Newton.

iterations to compute the Floquet multipliers accurately. For these iterations, we use the same convergence criterion as for the iterations to refine the random start basis, but with a stronger threshold ε_{sub} on the error norm of the subspace and a different (larger) value of ρ. Note that we do not really test the accuracy of the eigenvalues, but the accuracy of the corresponding maximal dominant invariant subspaces.

The other algorithms can be implemented in a similar way. For the $NPJ(l)$ variant, we must only remove the term $M(x(0),T)\,\Delta q$ and $c(x(0),T)^T\Delta q$ in the right hand side of the system for the Newton correction. For the $CNP(l)$ method, the Picard iterations must be done twice, once using r and once with $b_\gamma(x(0),T)$. The linear system in the Newton correction is also different, and after the computation of the Newton correction we need to reconstruct the Picard update (and for the Sherman–Morrison approach also the other unknowns) from the two contributions.

At this point it is worth emphasizing again that our overall strategy is tuned to utilize good starting values available because of the continuation context, and the algorithm is a balance between efficiency and reliability. In particular, if a failure occurs then the increment in the continuation parameter is halved and the computations restarted.

2.4.6 Work account

Let us first assume locking is not used. Algorithm 2.3 then requires one time integration of the nonlinear system (1.4) to compute the residual per iteration step. Furthermore, each iteration step requires $\nu_{sub}(p+p_e)+l$ matrix–vector products with M: $\nu_{sub}(p+p_e)$ for the subspace iterations, $l-1$ for the Picard scheme and 1 for the term $V_e^T M\Delta q$ in the right hand side of the P-system. All other variants also require one time integration of (1.4) at each step, but a different number of matrix–vector products. The $NPJ(l)$ and $CRP(l)$ methods both require $\nu_{sub}(p+p_e)+l-1$ matrix–vector products while the $CNP(l)$ method requires $\nu_{sub}(p+p_e)+2l$ matrix–vector products because the Picard iteration scheme needs to be executed twice. Furthermore, both the $CNP(l)$ and $CRP(l)$ methods also need the computation of b_γ. The computation cost of this vector is very similar to the cost of a matrix–vector product. The results are summarized in Table 2.1. Note that only the first $p+p_e$ matrix–vector products for the basis iteration and the computation of b_γ in the continuation variants can be done concurrently with the computation of φ. All other matrix–vector products depend on earlier results and can only be done at a later time. Our $NPJ(1)$ and $CRP(1)$ methods are the only variants where the time integrations of (1.4) and all variational equations can be done concurrently. This is a clear disadvantage of our approach compared to the full Newton method. The $NPJ(1)$ and $CRP(1)$ methods are appealing if the simultaneous integration of the nonlinear system and variational equations results in big gains. The full Newton approach may be appealing even for relatively large systems

- if large gains are possible from the simultaneous computation of the residual, b_γ and the monodromy matrix
- and if the $NPJ(1)$ or $CRP(1)$ are not fast and robust enough
- and if it is not possible to store the whole trajectory and matrix factorizations or preconditioner data to obtain similar gains with our better variants.

Note that one should also take into account the faster convergence of the full Newton method in the comparison.

Using a direct solver in the time integration code does not imply that single shooting with full Newton is cheaper than a Newton-Picard method. If $\partial f/\partial x$ has structure that can be exploited by a direct linear system solver, the computation of the N columns of the monodromy matrix is usually much more expensive than the time integration of (1.4). For instance, suppose that $\partial f/\partial x$ has a band structure with bandwidth d. The factorization at each time step is an $O(Nd^2)$ operation and the forward and backward substitutions are $O(Nd)$ operations. The N forward and N backward substitutions needed at each time step to compute the monodromy matrix cost $O(N^2d)$ operations, much more than the factorization of the linear systems at each time step during the integration of (1.4). Usually one factorization at each time step is sufficient. Hence for sufficiently large values of N we can certainly save by using any of our methods if p is small enough, even if we cannot store the information needed to compute the matrix–vector products cheaply without a new integration of (1.4).

The $CNP(l)$ method is considerably more expensive than the $NPGS(l)$ method except if the cost of matrix–vector multiplications is very low compared to the cost of the integration of (1.4). Therefore we advise the use of the $NPGS(l)$ method if branches of periodic solutions are computed and a switch to the more expensive $CNP(l)$ method if a fold point is approached. A fold point occurs at a double $+1$ Floquet multiplier. One can develop a detection criterion based on monitoring the available Floquet multipliers and switching to $CNP(l)$ if a Floquet multiplier (except the trivial 1) approaches 1. Note that a criterion only based on the Floquet multipliers will also switch to $CNP(l)$ in the neighbourhood of a transcritical- or pitchfork bifurcation.

The cost of a Newton–Picard iteration step in terms of the number of matrix–vector products does not depend on the dimension N of (1.4) since p is only determined by the problem. The convergence speed is also only determined by p. Therefore we expect that *the number of time integrations needed for a given reduction of the residual is independent of N*. Since the time integrations are the dominant cost factor in our method—the costs of the various linear algebra operations are at most linear in N and the cost of a time integrator is at least linear in N—*the cost of a Newton–Picard step scales as good or bad as the time integrator*. This is not the case if the full Newton method is used. Here the number of time integrations of the variational equations grows linearly with N and this cost may become much larger than the cost of the time integration of (1.4) if $\partial f/\partial x$ has a structure that can be exploited well. The solution of the linearized system in the single shooting method requires $\mathrm{O}(N^3)$ operations if a direct method is used. Hence the Newton–Picard method is very interesting for problems that require a fine discretization and have a moderate value of p.

A thorough analysis of the cost of different methods should go well beyond a simple comparison of the work per iteration step and also take into account the effects of locking and the different convergence speed of the methods. The cost for a matrix–vector product sometimes depends on whether the computations can be done concurrently with the residual computation or another matrix–vector product or not. The relative cost of the residual computation and the matrix–vector products is also very problem specific. Hence the results of any comparison will be very problem specific. Let us try to give some rough guidelines:

1. A small dimension N or a large value of p compared to N favours the use of the full Newton method. At larger values of N the Newton–Picard method is more interesting.

2. If the gain of the simultaneous integration of (1.4) and the variational equations is very large, Newton's method is interesting even for rather large values of N since it usually requires fewer iteration steps and thus fewer expensive integrations of (1.4). This situation occurs for instance if a direct method is used to solve the linear systems in the time integration code (since the LU factors can be reused) and if there is not enough memory to store all computed points on the trajectory $\varphi(x(0), t, \gamma)$, $t \in [0, T]$, and the associated LU-factors.

3. If we can gain from the simultaneous integration of (1.4) and the variational equations and are not able to store information to obtain similar gains if the integrations cannot occur simultaneously, the full Newton method should be considered for "small" to "moderate" values of N (although it is hard to give precise numbers) and the $NPJ(1)$ and $CRP(1)$ schemes should be considered for larger problems. If the $NPJ(1)$ and $CRP(1)$ are not robust and fast enough, one should consider to switch to the $NPGS(l)$ or $CNP(l)$ variants, but only for very large values of N.

4. If we can gain from the simultaneous integration of the nonlinear system (1.4) and the variational equations, and can store the information needed to get similar gains at later matrix–vector products, full Newton is still interesting for "small" and "moderate" system sizes (because it usually needs fewer iteration steps), while $NPGS(l)$ or $CNP(l)$ variants should be considered for large problems since they generally require less iteration steps than the $NPJ(l)$ and $CRP(l)$ methods. In the next chapter we will show that we can optimize the performance of the $NPGS$ and CNP methods and obtain a result in the same number of steps as the full Newton method (with a possibly growing cost per Newton–Picard step as the iteration

converges). These methods are also interesting for "moderate" system sizes in this scenario.

5. If we gain little or nothing from the simultaneous integration of all initial value problems or if the cost for the matrix–vector products is higher than for the integration of (1.4) (e.g., if higher order finite difference formulas have to be used), the $NPGS(l)$ or $CNP(l)$ schemes are interesting for all problems except if p/N is very large (much larger than 0.1). In the latter case, the full Newton approach is usually the better choice.

2.5 Test results

2.5.1 Test problems

In this section we will discuss two different test problems: the Brusselator model and a model studied by Olmstead et al. in [90]. Later in this text, some other models will be treated.

The Brusselator

The Brusselator model is a reaction–diffusion system and popular as a testcase in bifurcation analysis. Several variants of the model exist, both ODE models (coupled perfectly mixed cells) and PDE models. We will use the variant studied in [53]. The equations were already introduced in section 1.3. We repeat them here for convenience. The variant we study is a one-dimensional model with two reactants X and Y, modeled by the equations

$$\left\{\begin{array}{lcl} \dfrac{\partial X}{\partial t} & = & \dfrac{D_X}{L^2}\dfrac{\partial^2 X}{\partial z^2} + X^2Y - (B+1)X + A, \\[2ex] \dfrac{\partial Y}{\partial t} & = & \dfrac{D_Y}{L^2}\dfrac{\partial^2 Y}{\partial z^2} - X^2Y + BX, \end{array}\right. \tag{2.55}$$

with Dirichlet boundary conditions

$$\left\{\begin{array}{l} X(t,z=0) = X(t,z=1) = A, \\ Y(t,z=0) = Y(t,z=1) = \dfrac{B}{A}. \end{array}\right. \tag{2.56}$$

We use the characteristic length L as the bifurcation parameter while the other parameters are fixed at $A = 2$, $B = 5.45$, $D_X = 0.008$ and $D_Y = 0.004$. For these parameter values, branches of periodic solutions bifurcate from the trivial steady state branch ($X = A$, $Y = B/A$) at Hopf bifurcation points at

$$L_k^H = k\pi\sqrt{\frac{D_X + D_Y}{B - A^2 - 1}} = k\,0.5130$$

[51]. For the reported results, we use an $O(h^2)$ finite difference space discretization with grid size $h = 1/32$, yielding a system of ODEs of dimension $N = 62$. The time integrations

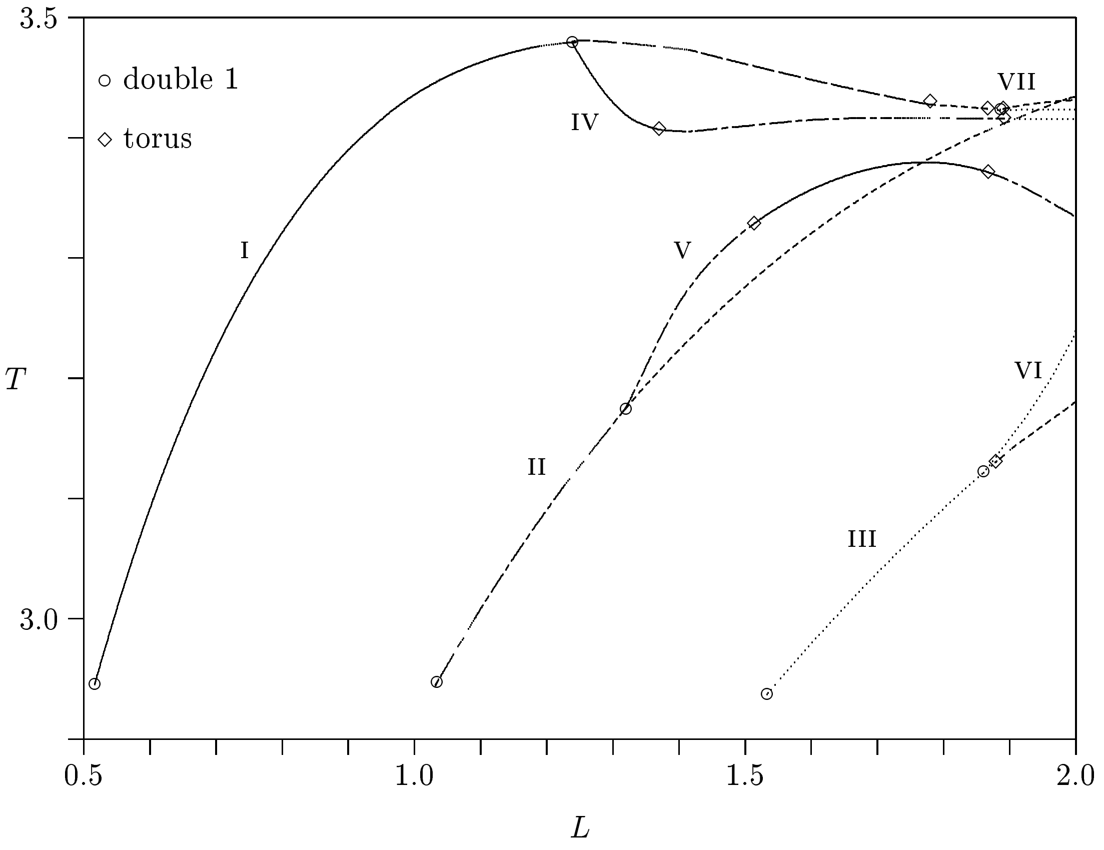


Figure 2.2: Periodic solutions bifurcation diagram for the discretised Brusselator model ($h = \frac{1}{32}$), period T versus the reactor length L. The roman numbers indicate the numbering of the branches used in this section's tables. Double one Floquet multipliers (pitchfork bifurcations and Hopf points) and torus bifurcations are marked with ○ and ◇ respectively. No period doublings occur on the computed branches.

were done using the LSODE package [95]. The bifurcation diagram for the periodic solutions for L between 0.5 and 2 is shown in Figure 2.2. Several torus- and pitchfork bifurcation points were detected on the branches. They were located accurately by taking very small stepsizes from a nearby point on the branch. The new branches emanating from the various pitchfork bifurcations were also computed. Note that on branch I the torus bifurcation at $L \approx 1.867$ is immediately followed by a pitchfork bifurcation at $L \approx 1.887$. On the emanating branch VII there is almost immediately a torus bifurcation at $L \approx 1.8904$. In the next chapter, we will also present a result for the two-dimensional variant of (2.55)

$$\begin{cases} \dfrac{\partial X}{\partial t} = \dfrac{D_X}{L^2}\left(\dfrac{\partial^2 X}{\partial x^2} + \dfrac{\partial^2 X}{\partial y^2}\right) + X^2 Y - (B+1)X + A, \\[2ex] \dfrac{\partial Y}{\partial t} = \dfrac{D_Y}{L^2}\left(\dfrac{\partial^2 Y}{\partial x^2} + \dfrac{\partial^2 Y}{\partial y^2}\right) - X^2 Y + BX \end{cases} \tag{2.57}$$

on the unit square $[0,1] \times [0,1]$ with the Dirichlet boundary conditions (2.56) on all boundaries. Here we used the parameter values $A = 2$, $B = 5.45$, $D_X = 0.004$ and

$D_Y = 0.008$. The first Hopf bifurcation on the trivial branch, leading to branch of stable periodic solutions, occurs at $L \approx 0.72$.

The Olmstead model

The Olmstead model is a simple model for fluid flow with memory [90, 47]. The model is a one-dimensional model given by the equation

$$\frac{\partial u}{\partial t} = \int_{-\infty}^{t} K(t,s)\,\frac{\partial^2 u}{\partial x^2}\,ds + Ru - u^3 \tag{2.58}$$

with Dirichlet boundary conditions

$$u(t, x = 0) = u(t, x = \pi) = 0$$

on the interval $[0, \pi]$ and with either the Maxwell kernel

$$K(t,s) = \frac{1}{\lambda}\exp(-\frac{t-s}{\lambda})$$

or the more general Jeffreys kernel

$$K(t,s) = \frac{1-\delta}{\lambda}\exp(-\frac{t-s}{\lambda}) + 2\delta\,\boldsymbol{\delta}(\mathbf{t}-\mathbf{s}) \tag{2.59}$$

with $\boldsymbol{\delta}(t-s)$ the Dirac delta function. In (2.59), λ is a relaxation time and δ represents the ratio of retardation time to relaxation time. Let

$$v(x,t) = \frac{1}{\lambda}\int_{-\infty}^{t} e^{-\frac{t-s}{\lambda}}\,u(x,s)\,ds, \tag{2.60}$$

then (2.58) with the Jeffreys kernel (2.59) and (2.60) can be rewritten as a system of two coupled PDEs

$$\left\{\begin{array}{rcl} \dfrac{\partial u}{\partial t} & = & \delta\dfrac{\partial^2 u}{\partial x^2} + (1-\delta)\dfrac{\partial^2 v}{\partial x^2} + Ru - u^3, \\ \lambda\dfrac{\partial v}{\partial t} & = & u - v, \end{array}\right. \tag{2.61}$$

with Dirichlet boundary conditions

$$\left\{\begin{array}{l} u(t, x = 0) = u(t, x = \pi) = 0, \\ v(t, x = 0) = u(t, x = \pi) = 0. \end{array}\right. \tag{2.62}$$

We used the Rayleigh number R as the bifurcation parameter. The values of the parameters λ and δ were fixed at $\lambda = 2.0$ and $\delta = 0.1$. The steady-state bifurcation diagram is shown in Figure 2.3. This diagram was computed using an $O(h^2)$ finite difference discretization resulting in a system of 80 ODEs. On the steady state solution branch bifurcating from the trivial branch at $R = 1$, Hopf points arise at $R \approx 1.1007$ and $R \approx 1.2040$. We computed parts of the (initially stable) periodic solution branch, bifurcating from the trivial branch at $R = 0.6$, and of the (unstable) branch emanating at

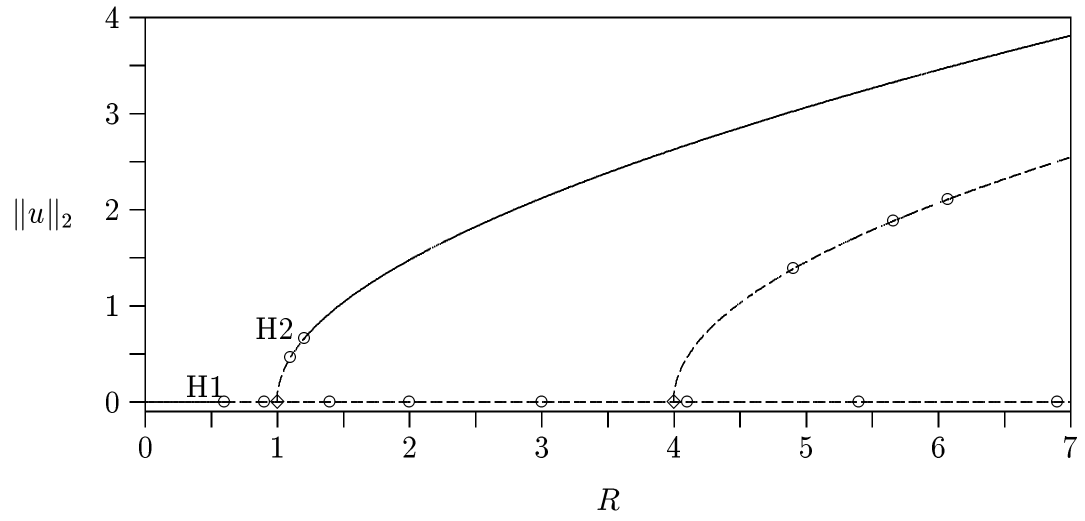


Figure 2.3: Steady-state bifurcation diagram for the Olmstead model, two-norm of the solution profile versus the parameter R. Hopf bifurcation points are marked with $\circ$, pitchfork bifurcations with $\diamond$.

$R = 1.2040$ on the first bifurcated steady state branch. The results are shown in Figure 2.4. To compute this diagram we used the trapezoidal rule for the time integrations and the variational equations for the matrix–vector products. We used some of the improvements explained in the next chapter to obtain this diagram. Without the improvements the code broke down early. Therefore, the results presented in the remainder of this chapter are limited to $R \leq 1.267$ on the first branch and $R \leq 1.24$ on the second. On branch 1, a fold point occurs around $R \approx 1.268930$ and $T \approx 31$. For larger periods, the dominant Floquet multiplier grows quickly and the trivial 1 Floquet multiplier looses some accuracy. At $T = 35.9$, the largest Floquet multiplier is already around 25. On branch 2, there is a torus bifurcation around $R \approx 1.243$ and $T \approx 18.6$. As the period increases, the convergence slows down considerably. The problems are caused by residual components in the P-direction. Since the convergence slows down more and more as the iterations converge, we suspect that inaccuracies in the result of the time integrator cause the problems.

2.5.2 Influence of the basis threshold

To validate that the Newton–Picard procedure will certainly work given an accurate basis and to study the influence of the threshold ρ, free from other influences, we adapted the bifurcation software package LOCBIF of Khibnik et al. [67, 68] to simulate the $CNP(1)$ scheme. In LOCBIF, the matrix M is constructed by numerical differencing and periodic solutions are computed by shooting based on a full Newton iteration. We replaced the matrix $M - I$ in the shooting code with

$$\begin{bmatrix} V_q & V_p \end{bmatrix} \begin{bmatrix} -I & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p \end{bmatrix} \begin{bmatrix} V_q^T \\ V_p^T \end{bmatrix}$$

and b_T with its P-projection. V_p and V_q were computed by applying the QR algorithm on M and are exact bases up to roundoff errors.

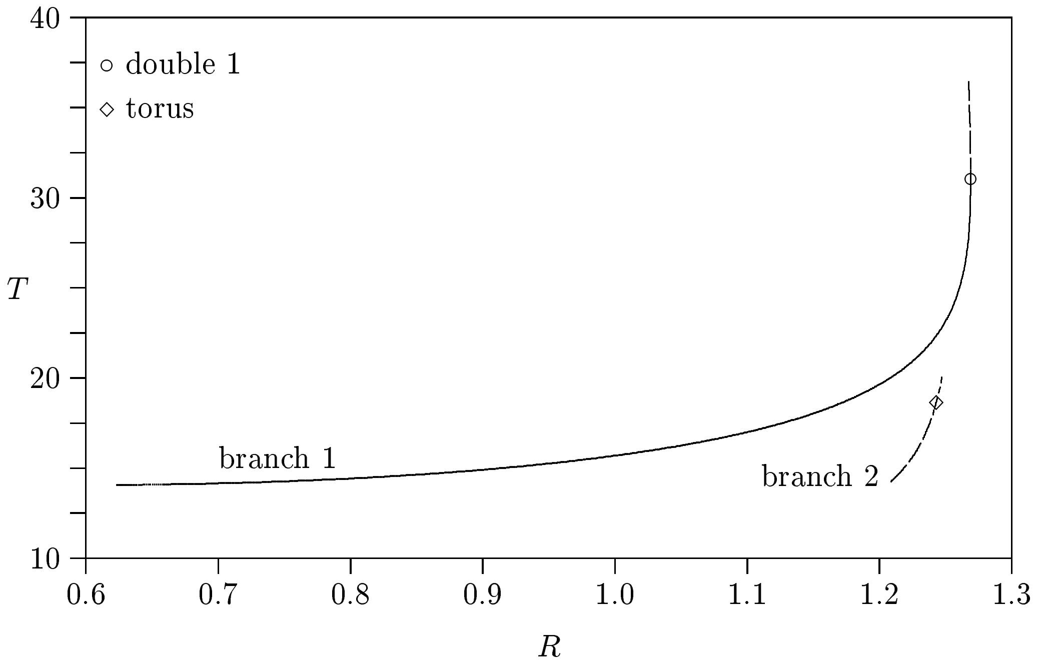


Figure 2.4: 2 (incomplete) periodic solution branches of the Olmstead model, T versus the Rayleigh number R.

We have computed a section on the first branch of periodic solutions for the Brusselator model using threshold values $\rho = 0.75, 0.50, 0.25$ and 0. Note that $\rho = 0$ corresponds to shooting based on a full Newton iteration. The parameters of the continuation procedure in LOCBIF have been set such that the same continuation points are computed during all runs. In Table 2.2 we present for $N = 30$ the number of eigenvalues larger than ρ, and the required number of Newton–Picard iterations in each continuation step.

Although the dominant eigenvalue increases in modulus along the branch, most of the eigenvalues have small modulus. For example, for $L \approx 1$ (continuation point 9), the number of eigenvalues with modulus larger than $\rho = 0.75, 0.50$ and 0.25 is 1, 2 and 4 respectively. For $L \approx 1.5$ (continuation point 15) these numbers are 2, 6 and 6 respectively. If a full Newton iteration is used ($\rho = 0$), three to four iterations per continuation step are sufficient. If a sufficiently small threshold ρ ($\rho \leq 0.5$) is used, the number of Newton–Picard iterations is not much larger. If the threshold is set to $\rho = 0.75$, the number of iterations may grow substantially. This is for example the case for continuation point 15 where an eigenvalue (pair) approaches 0.75 in modulus ($|\mu_3| = |\mu_4| \approx 0.7$). In the next continuation step these eigenvalues become larger than the threshold value. Then p increases and the number of iteration steps is strongly reduced. Clearly, the convergence rate of the Newton–Picard iteration is determined by the largest eigenvalue smaller than the threshold, which in fact determines the convergence rate of the Picard step.

The optimal value of ρ is problem dependent. Values of ρ between 0.5 and 0.25 often gave good results in our tests. However, all of our examples had spectra that were strongly clustered around 0. If the spectrum is less clustered, larger values of ρ may be appropriate. In this case, the slower convergence is compensated by the cheaper iterations

			$\rho = 0.75$		$\rho = 0.5$		$\rho = 0.25$		$\rho = 0.0$	
point	L	T	dim. p	# iter.	dim. p	# iter.	dim. p	# iter.	dim. p	# iter.
0	0.545	3.00	2	/	2	/	2	/	30	/
1	0.588	3.07	1	8	2	4	2	4	30	3
2	0.636	3.14	1	8	2	4	2	4	30	3
3	0.685	3.21	1	8	1	8	4	5	30	3
4	0.734	3.26	1	7	1	7	4	5	30	3
5	0.784	3.31	1	7	1	7	4	5	30	3
6	0.833	3.35	1	7	1	7	3	7	30	3
7	0.884	3.38	1	8	2	8	2	8	30	3
8	0.936	3.41	1	8	2	8	4	5	30	3
9	0.991	3.43	1	9	2	9	4	5	30	3
10	1.05	3.45	2	9	2	9	4	5	30	3
11	1.12	3.47	2	11	2	11	4	5	30	3
12	1.19	3.48	2	13	2	13	4	5	30	3
13	1.29	3.48	2	18	4	5	6	5	30	4
14	1.39	3.47	2	25	4	6	6	6	30	3
15	1.49	3.46	2	32	6	6	6	6	30	3
16	1.59	3.45	4	8	6	8	6	8	30	3
17	1.69	3.43	6	9	6	9	8	5	30	3

Table 2.2: Continuation of the first branch of periodic solutions for the Brusselator model by the Newton-Picard scheme using exact values for p and V_1 (Algorithm 2.1) : dimension p and number of Newton–Picard iterations needed to achieve an accuracy of 10^{-8}.

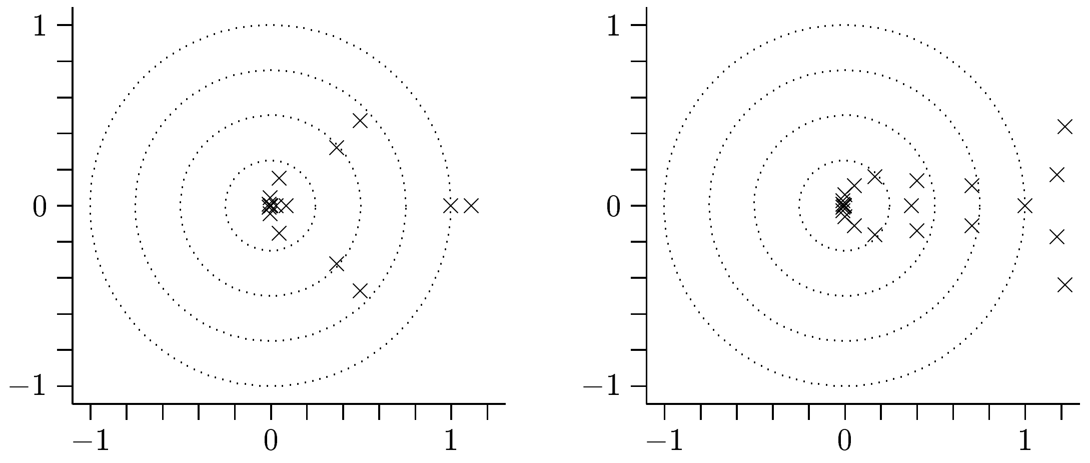


Figure 2.5: Spectra of the Brusselator model, 31 discretisation points, solution at $L = 1.4833$ on branch I and solution at $L = 1.9$ on branch VI.

because the basis size is much smaller.

2.5.3 Comparison of the variants

In this section, we make a comparison of the $NPGS(l)$, $NPJ(l)$, $CNP(l)$ and $CRP(l)$ methods and a single shooting code using the classical Chord-Newton method. We will first discuss some of the parameters used in the algorithm.

For the test results reported in this section, we used $\rho = 0.5$ and four extra vectors (i.e., $p_e = 4$, see section 2.4). Using four extra vectors instead of the minimal number of two, greatly improved the convergence and the robustness of the subspace iterations, and resulted in fewer IVP solves overall. The use of further extra vectors or smaller values of ρ did not result in further improvements in our test examples. This can easily be explained heuristically from the spectral picture of M, see Figure 2.5. By using $\rho = 0.5$ and four extra vectors, we generally capture all basis vectors outside the cluster around 0. The theoretical linear asymptotic convergence factor of the Newton-Picard iterations becomes so small that the asymptotic rate is not reached at all. In fact, in the initial iterations, the Q-projection of the residual decreases faster than the P-projection, which means that the behaviour of the Newton component becomes dominant. However, when multiple eigenvalues can occur, for instance in problems with more space dimensions and geometries with a lot of symmetry, one may need to use more extra vectors.

We used a linear phase condition based on the predicted point $(x_p(0), T_p, \gamma_p)$ for the Newton–Picard iterations:

$$f(x_p(0), \gamma_p)^T (x(0) - x_p(0)) = 0. \tag{2.63}$$

This condition defines a $(N-1)$-dimensional hyperplane (called the Poincaré return map) and should intersect the limit cycle transversally, which requires

$$f(x_p(0), \gamma_p)^T f(x(0)^*, \gamma^*) \neq 0$$

(see, e.g., [37, 106]). (2.63) satisfies this requirement for reasonable starting values.

We computed several branches of periodic solutions using a simple variable stepsize continuation code based on the strategy used in Locbif [67, 68]. A maximum and minimum predictor stepsize is user-imposed. After a convergence failure, the stepsize is halved. The computations are stopped if the minimum stepsize is reached. If a point is computed sufficiently fast, the stepsize is increased by a factor depending on the convergence history for the previous points and the maximum stepsize. We do not use an angular control criterion as in Locbif since such criteria performed badly in experiments with large-scale systems. The variable stepsize strategy allows us to observe not only differences in (asymptotic) convergence speed, but also differences in the size of the attraction domain of the different methods. Some of the reported results use a pseudo-arclength parameterization. Two linear parametrization equations were tested:

$$n_1\left(x(0),T,\lambda\right) = \left(x(0)-x_p(0)\right)^T x_s(0) + \theta_T(T-T_p)T_s + \theta_\gamma(\gamma-\gamma_p)\gamma_s, \tag{2.64}$$

and

$$\begin{aligned} n_2\left(x(0),T,\gamma\right) &= \left(x(0)-x_p(0)\right)^T \frac{f(x_p(0))f(x_p(0))^T}{f^T(x_p(0))\,f(x_p(0))} x_s(0) \\ &\quad +\theta_T(T-T_p)T_s + \theta_\gamma(\gamma-\gamma_p)\gamma_s. \end{aligned} \tag{2.65}$$

$(x_s(0),T_s,\lambda_s)$ is the predictor step along the secant vector and $(x_p(0),T_p,\gamma_p)$ the predicted point. θ_T and θ_γ are two positive scaling parameters. Condition (2.65) is derived from (2.64) by projecting the $x(0)$ -components of the secant vector on the tangent vector of the trajectory starting in the predicted point. Note that condition (2.65) can be dangerous: c_n will lie in the same direction as c_s, so if $\theta_T T_s$ and $\theta_\gamma\gamma_s$ are both zero then the linearized phase condition and parametrizing equation are linearly dependent and the linearized system in the shooting method will be singular.

Several variants of the method were tested, using different values for the parameters. Here we summarize the main conclusions. The main cost in the overall scheme is the number of initial value problem (IVP) solves. In Table 2.3 we use this measure to compare five methods, namely:

1. $NPJ(l)$ (Newton–Picard Jacobi), based on (2.12) and

$$\begin{bmatrix} V_p^T(M-I)V_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = -\begin{bmatrix} V_p^T r \\ s \end{bmatrix}. \tag{2.66}$$

   The parameter γ is fixed during the corrector iterations with this scheme.

2. $NPGS(l)$ (Newton–Picard Gauss–Seidel), based on (2.12) and (2.11). As with the $NPJ(l)$ method, the parameter γ is fixed during the corrector iterations.

3. $CRP(l)$ (Continuation variant of the Recursive Projection method), based on

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = -\begin{bmatrix} V_p^T r \\ s \\ n \end{bmatrix},$$

   (2.38) and parametrizing equation (2.65).

Method	Brusselator model				Olmstead model		Total
Branch	I	II	III	IV	1	2	
Start at $L/R =$	5.55	1.04	1.55	1.30	0.623	1.209	
$T =$	3.017	2.948	2.947	3.433	14.06	14.27	
End at $L/R =$	2.0	2.0	2.0	2.0	1.267	1.24	
$T =$	3.424	3.436	3.197	3.415	27.5	18.0	
NPJ	1045	1611	959	1463	864	385	6327
NPGS	723	721	682	950	832	412	4320
CRP	905	852	746	919	1238	539	5199
CNP	987	932	753	998	1408	662	5740
Average	915	1029	785	1083	1086	500	5397
Chord–Newton	2053	1607	1128	1874	9229	3791	19682

Table 2.3: Number of IVP solves (best results obtained over 12 continuation runs)

4. $CNP(l)$ (Continuation Newton–Picard). For the tests in this section, we use the variant based on the Sherman–Morrison formula explained in (2.22)-(2.25) and parametrizing equation (2.65).

5. Chord–Newton applied to (2.19) directly. We tried several strategies for updating the Jacobian matrix and only report the best result in Table 2.3. Here we used the parametrizing equation (2.64).

All Newton–Picard methods were implemented similarly to algorithm 2.3 with $\nu_{sub} = 1$. We did computations on four branches of the Brusselator model (Figure 2.2) and two branches of the Olmstead model (Figure 2.4). All tests were run using the LSODE package for time integration and finite differences for matrix–vector products and the construction of M. Finite differences were accurate enough in this region, and the lack of control over the sequence of stepsizes and orders in the variable step / variable order time integration method also posed no problems. However, if one tries to compute points further on the branches of the Olmstead model, the use of variational equations and a time integrator with strict control over the stepsizes is essential. All results reported included the accurate computation of all Floquet multipliers larger than 0.7 for the Newton Picard methods and all Floquet multipliers for the Chord–Newton method (using the QR algorithm from LAPACK on the monodromy matrix). After the Newton–Picard iterations (where the basis update is stopped once the Floquet multipliers have two or three accurate digits) we generally needed only one or two subspace iteration steps to compute the dominant Floquet multipliers with four digits of accuracy.

Table 2.3 lists the number of IVP solves for each method on the two problems. To be precise, we computed the branches using one, two or three Picard steps and four values for the maximal steplength of the continuation code, but report only the best result of the twelve combinations in the table. The last column lists the total number of IVP solves. It is seen that $NPGS$ performs best overall and NPJ worst, being about 50% more expensive than $NPGS$. Note however that the difference in performance occurs for the Brusselator example and that for the Olmstead model the performance of both methods

Method	Brusselator model				Olmstead model		Total
Branch	I	II	III	IV	1	2	
NPJ	16	21	11	18	16	7	89
NPGS	11	9	7	12	15	7	61
CRP	13	9	7	11	23	8	71
CNP	13	9	7	11	23	9	72
Average	13	12	8	13	19	8	73
Chord–Newton	9	7	5	8	22	9	60

Table 2.4: Number of continuation points (best results obtained over 12 continuation runs)

Method	Brusselator model				Olmstead model		Average
Branch	I	II	III	IV	1	2	
NPJ	2	1	1	1	1	1	1.17
NPGS	2	2	3	1	1	1	1.67
CRP	1	1	3	2	1	1	1.50
CNP	1	1	2	1	1	1	1.17

Table 2.5: Value for l used to obtain the results presented in the previous tables

is the same. This superiority of *NPGS* is mainly due to the larger domain of attraction and higher convergence performance far from the solution. Close to a solution, the difference in convergence speed between the Gauss–Seidel and Jacobi variant was found to be negligible, in agreement with corollary 2.3. The Gauss–Seidel variant thus allows larger stepsizes (and hence fewer continuation points) in a continuation procedure. As would be expected *CNP*, which uses the rank one update approach, and *CRP* are both more expensive than *NPGS*. Compared to the *CRP* method, the additional IVP solves per iteration needed in the *CNP* method because of the full coupling do not result in improved convergence speed in these testcases. However, the *CNP* method proves to be the most robust method. As expected Chord–Newton is the most expensive method, but it is significant that even for these relatively small problems *NPGS* is over four times faster than Chord–Newton. For larger N the superiority would be even more marked because of the scaling discussion in section 2.4.6.

Table 2.4 compares the same methods with the number of continuation steps taken to compute the branch. Similar conclusions can be drawn from this table.

For the four Newton–Picard methods, the number of Picard steps to compute $\Delta\bar{q}$ in (2.12) is varied from $l = 1$ to $l = 3$. Table 2.5 indicates which value for l provided the fastest performance, the figures for which were used to produce the tables 2.3 and 2.4. For the Olmstead model it is striking that $l = 1$ was best for all runs. For the Brusselator, the results are less clear cut, though the choice $l = 1$ was best in 9 out of 16 runs. There are two possible explanations for this. First, for our choice of parameters the theoretical convergence rate is quite high and is not reached in practice even for $l = 1$. In fact, the transitional behaviour of the Newton part often dominates the convergence.

Second, the convergence speed of the subspace iteration procedure does not change if l is changed. The basis is not accurate enough to get the full theoretical convergence speed. For the Olmstead model the first option is clearly the right one. This model has a spectrum that is extremely clustered around 0. In fact, if four extra vectors are used, the modulus of the smallest computed eigenvalue was often around 10^{-3}. The basis iterations converged extremely fast and the basis was very accurate. For the Picard iterations, we noted a convergence rate of 10^{-3} or better. For the Brusselator model the explanation of the convergence behaviour is not so easy. The behaviour is caused by a combination of rather high convergence speeds of the Picard scheme and basis inaccuracy.

We have also implemented and tested continuation variants that omit the (1,4)-block $V_q^T b_\gamma$ in (2.21). These variants performed considerably worse than the CNP and CRP methods even on these easy branches without fold points. Another option that has been explored is to replace matrix–vector products with time integration of (1.4) from a different starting value:

- The term $r + MV_q\Delta\bar{q}$ in the right hand side of (2.11) is replaced by
$$\varphi(x(0) + V_q\Delta\bar{q}, T, \gamma) - (x(0) + V_q\Delta\bar{q}).$$
- The Picard scheme (2.12) or (2.50) is replaced by
$$\begin{cases} q^{[0]} = Qx(0), \\ q^{[i]} = q^{[i-1]} + Qr(x(0) + q^{[i-1]}, T, \gamma), \quad i = 1, \ldots, l, \\ \Delta q = q^{[l]} - Qx(0). \end{cases}$$
- The Picard scheme (2.38) in the $CRP(l)$ method is replaced by
$$\begin{cases} q^{[0]} = Qx(0), \\ q^{[i]} = q^{[i-1]} + Qr(x(0) + q^{[i-1]}, T, \gamma + \Delta\gamma), \quad i = 1, \ldots, l, \\ \Delta q = q^{[l]} - Qx(0). \end{cases}$$

We have reworked the $NPJ(l)$, $NPGS(l)$ and $CRP(l)$ methods in this way. We did not rework the $CNP(l)$ method since there is no equivalent with time integrations for the Picard scheme for the system
$$\left(V_q^T M V_q - I_q\right)\Delta\bar{q}_\gamma = -b_\gamma.$$
The performance of these variants was very similar to the performance of the corresponding scheme using matrix–vector products. These schemes may be interesting if the computation of matrix–vector products is expensive compared to time integration of (1.4). It is not clear however that these schemes will also work in more difficult situations since we neglect higher order terms if the matrix–vector products are replaced with time integrations. In the next chapter we shall see how we can assure that the $NPGS$ and CNP methods always work if full Newton with a direct solver works. We do not have a similar method for the variants with time integration instead of matrix–vector products.

2.6 Conclusions

This chapter introduced the Newton–Picard single shooting method. This method in its simplest form combines time integration (a kind of Picard iteration) and Newton-based single shooting. The method is inspired by the work of Shroff and Keller [107] and Jarausch and Mackens [56, 57, 58]. Compared to other authors, we take an original starting point. Our method starts from the Newton linearization of the nonlinear single shooting system instead of the nonlinear system itself.

The chapter first concentrated on the computation of a single orbit. The linear system is projected on the low-dimensional generalized eigenspace of the monodromy matrix corresponding to the unstable and weakly stable modes and its high-dimensional orthogonal complement. From the splitting, we also obtain information on the dominant, stability-determining Floquet multipliers. Hence the Newton–Picard method is very well suited to be used for bifurcation analysis. The dimension of the low-dimensional subspace is determined by the problem itself and not by its discretization. The projected linear system is block-triangular. The high-dimensional subsystem is solved using Picard iteration. The first Picard step is basically time integration projected into the high-dimensional subspace. However, we allow to perform more than one Picard step which is an extension to the method of Shroff and Keller. The low-dimensional subsystem has precisely the same structure as the original linearized system and is solved using a direct linear system solver. A method equivalent to the method of Shroff and Keller is obtained by using only one Picard iteration step and neglecting the coupling between the high-dimensional and low-dimensional subsystems. Both methods can be seen as methods that solve a system with an approximation to the Jacobian matrix of the single shooting system exactly. This allowed us to prove an asymptotic convergence result under the assumption that we compute the basis for the low-dimensional subspace exactly. As one would expect, both methods have linear asymptotic convergence and the convergence rate is determined by the Picard scheme and the size of the low-dimensional subspace. Although neglecting the coupling between the high- and low-dimensional subsystems has no influence on the theoretical asymptotic convergence rate, a big performance difference is observed in tests where an imperfect basis is used.

If the method is applied to the single shooting system in a pseudo-arclength continuation code, the projected system is not block triangular anymore. We developed two techniques to deal with this problem: a technique based on the Sherman–Morrison formula and a method similar to a technique used in [83]. The latter method has some advantages over the former since it does preserve more information on bifurcation phenomena in the low-dimensional system. Both techniques require the solution of two high-dimensional subsystems with Picard iterations and can also return an approximation to the tangent vector.

We also discussed various implementation aspects: the computation of matrix–vector products, computation of other derivatives needed, the efficient implementation of the Picard iterations and the computation of the basis for the low-dimensional subspace. For the latter task, we concentrate on robustness in our methods. Hence our strategy differs at various points with previous work. We use the orthogonal subspace iteration method with projection and locking (other authors use ordinary orthogonal subspace iteration) and developed a robust strategy to determine the basis size. The latter is important since we want to detect bifurcation points in a very robust way. It is important to note that the cost for one iteration step is proportional to the cost of one time integration. The amount of matrix–vector products with the monodromy matrix needed in the subspace iterations and the Picard iterations and the convergence rate of the Picard scheme is independent of the number of degrees of freedom of the discretization. Hence, if an efficient time integration code is available, the method is particularly suited for problems that require a fine discretization to capture the behaviour of the physical system correctly and is very well suited to produce quantitative results, which usually require much finer discretizations than are required for the generation of qualitative results only.

We concluded the chapter with test results for two problems that illustrate the various claims made. These test results show that our methods are clearly superior over methods that neglect the coupling between the high- and low-dimensional subsystems such as the method of Shroff and Keller [107].

To the best of our knowledge, only Jarausch [55] has also studied a similar technique to compute periodic solutions using single shooting. We will discuss his method in section 4.5. However, his approach is very different from ours. He does not distinguish between the state, period and system parameter and uses different subspaces to project the system. His choice leads to a decoupling of the high- and low-dimensional systems. However, it is much more difficult to construct the required basis. Jarausch cannot recover as much information on the bifurcations as we can do and, as we shall see, treating the state, period and system parameter in a uniform way also has some other disadvantages. Compared to other work, we took an original starting point (the linearized system) and discovered the coupling between the high- and low-dimensional subsystems and showed the consequences of neglecting the coupling in our test cases. We also extended the method to deal with the bordering coming form the phase condition and pseudo-arclength equation and the corresponding derivatives with respect to the period and system parameter, and also used multiple Picard steps to solve the high-dimensional subsystems. We were also the first to use orthogonal subspace iteration with projection and locking instead of ordinary orthogonal subspace iteration in a method of this type and also took an original approach at determining the size of the low-dimensional subspace, aimed at a higher robustness than previous methods. The work described in this chapter is published in [74, 75, 98].

Chapter 3

Single shooting: advanced methods

3.1 Introduction

In this chapter, we will further develop the $NPGS(l)$ and Moore–and–Spence-like variant of the $CNP(l)$ method. In our test results reported at the end of the previous chapter, the $NPGS(l)$ method was clearly a better performer than the $NPJ(l)$ scheme. The $CNP(l)$ method performed worse than the $CRP(l)$ method in terms of time integrations, but was more robust. We will develop a new and more algebraic view of the method. In the previous chapter, we developed an approximation to the Jacobian matrix of the single shooting method in the $[V_q \;\; V_p]$-basis so that the linear systems can be solved easily. We were able to prove the asymptotic convergence rate of the $NPGS(l)$ method under the assumption of a perfect basis, but could not analyze the effect of basis inaccuracy and couldn't prove any result for the $CNP(l)$ method. In this chapter, we take a different approach. The high-dimensional and hard-to-solve (because of singularities and unstable modes) system is split-up in one or more high-dimensional but more easily solvable Q-subsystems and a low-dimensional P-system that contains the singularities of the original problem. Each of these subsystems will be solved with appropriate iterative or direct techniques. We will be able to analyze the convergence behaviour of the method at any point during the iterations in terms of the performance of the solvers for the subsystems, the basis (in)accuracy, the size of neglected terms and the higher-order effects from the Newton linearization. The analysis will explain the superiority of the $NPGS(l)$ method over the $NPJ(l)$ method in computations, although the theoretical analysis in the previous chapter predicted the same asymptotic convergence rate. The analysis will also indicate the changes needed to the $NPGS(l)$ and $CNP(l)$ schemes to ensure that the methods work in as many cases as possible and will show why it is possible to find cases where the $CRP(l)$ method will fail. In fact, the proposed changes to the $NPGS(l)$ and $CNP(l)$ methods can ensure that the resulting methods almost always converge if the corresponding full Newton method with direct solver works. This is not possible with the $NPJ(l)$ or $CRP(l)$ methods of [107].

The structure of this chapter is as follows. First, we will develop a new algebraic framework. All matrix–vector product based methods developed in the previous chapter fit into this framework, but the framework will allow a more practical convergence analysis and the development of new, more powerful and more robust variants. We will derive a formula that relates the final error of a Newton–Picard iteration step to the errors of the various subsystem solvers, the basis accuracy, size of neglected terms and the higher-order terms from the Newton linearization. In section 3.3 we will discuss why the $NPJ(l)$ and $CRP(l)$ methods sometimes perform badly and illustrate some of our claims with graphs of the convergence. In section 3.4 we will show how the algebraic framework and the convergence relation can be used to improve the robustness and performance of the $NPGS$ and CNP methods. We will discuss iterative methods for the solution of the high-dimensional Q-subsystems in section 3.5. We will consider the Picard iteration scheme and the GMRES method and also discuss the choice of starting values for the iterative methods. In section 3.6 we briefly discuss the solution of the low-dimensional P-system. Although we have not yet done any experiments with integral phase conditions, we will briefly discuss their use in shooting codes in section 3.7 to dispel the prejudice that integral phase conditions cannot be used with shooting codes. In section 3.8 we will present some testcases that illustrate some of the points made in this chapter. The chapter ends with a discussion of the application of the method to ordinary differential equations with one or more discrete delays (further denoted as DDEs or delay differential equations). The latter is joint work with T. Luzyanina (Institute of Mathematical Problems in Biology, Pushchino, Moscow region, Russia) and K. Engelborghs (K.U.Leuven, Department of Computer Science).

3.2 The algebraic framework

3.2.1 Derivation

Suppose we solve (2.16) for the unknowns $x(0)$, T and γ. Newton's method leads to the repetitive solution of (2.19). Under assumption 2.4 (see section 2.3) and the coordinate change (2.5-2.6), (2.19) transforms to

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & V_q^T b_T & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}. \qquad (3.1)$$

The term $V_q^T b_T$ is usually small and was neglected in the previous chapter, but we keep this term here in our analysis. If an exact basis is used, $V_q^T M V_p$ is zero. However, for the approximate basis computed via subspace iteration, this term is close to zero but not exactly zero. We neglect the term $V_q^T M V_p$. In the spirit of the Moore–and–Spence-like approach of section 2.3, (3.1) can be solved approximately by first solving $\Delta\bar{q}_r$, $\Delta\bar{q}_T$ and $\Delta\bar{q}_\gamma$ approximately from

$$\left[V_q^T M V_q - I_q\right] \begin{bmatrix} \Delta\bar{q}_r & \Delta\bar{q}_T & \Delta\bar{q}_\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r & V_q^T b_T & V_q^T b_\gamma \end{bmatrix}, \qquad (3.2)$$

then computing $\Delta\bar{p}$, ΔT and $\Delta\gamma$ from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T(b_T + MV_q\Delta\bar{q}_T) & V_p^T(b_\gamma + MV_q^T\Delta\bar{q}_\gamma) \\ c_s^T V_p & d_{s,T} + c_s^T V_q \Delta\bar{q}_T & d_{s,\gamma} + c_s^T V_q\Delta\bar{q}_\gamma \\ c_n^T V_p & d_{n,T} + c_n^T V_q\Delta\bar{q}_T & d_{n,\gamma} + c_n^T V_q \Delta\bar{q}_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = \\ - \begin{bmatrix} V_p^T(r + MV_q\Delta\bar{q}_r) \\ s + c_s^T V_q \Delta\bar{q}_r \\ n + c_n^T V_q \Delta\bar{q}_r \end{bmatrix} \tag{3.3}$$

and finally computing

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta T\,\Delta\bar{q}_T + \Delta\gamma\,\Delta\bar{q}_\gamma. \tag{3.4}$$

The corresponding scheme for the fixed parameter case (i.e., γ is not considered a variable) is found by omitting the last row and column in (3.3) and the term in $\Delta\bar{q}_\gamma$ in (3.2) and (3.4), i.e., we first solve $\Delta\bar{q}_r$ and $\Delta\bar{q}_T$ from

$$\left[V_q^T M V_q - I_q\right] \left[\begin{array}{cc} \Delta\bar{q}_r & \Delta\bar{q}_T \end{array}\right] = - \left[\begin{array}{cc} V_q^T r & V_q^T b_T \end{array}\right], \tag{3.5}$$

then compute $\Delta\bar{p}$ and ΔT from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T(b_T + MV_q\Delta\bar{q}_T) \\ c_s^T V_p & d_{s,T} + c_s^T V_q \Delta\bar{q}_T \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T(r + MV_q\Delta\bar{q}_r) \\ s + c_s^T V_q\Delta\bar{q}_r \end{bmatrix}, \tag{3.6}$$

and finally compute

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta T\,\Delta\bar{q}_T. \tag{3.7}$$

Note that this algebraic framework allows a different number of Picard iterations or even different solvers for each of the right hand sides of (3.2) and (3.5). Neglecting the $V_q^T b_T$ or $V_q^T b_\gamma$ term in (3.1) corresponds to setting $\Delta\bar{q}_T = 0$ or $\Delta\bar{q}_\gamma = 0$ respectively. Neglecting the $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ terms correspond to omitting all corresponding terms in (3.3). The $CNP(l)$ scheme is recovered by setting $\Delta\bar{q}_T = 0$ and computing $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$ in (3.2) using an l-step Picard iteration. The $NPGS(l)$ scheme is recovered by setting $\Delta\bar{q}_T$ to zero, computing $\Delta\bar{q}_r$ from (3.5) using the l-step Picard scheme (2.12) and computing $\Delta\bar{p}$ and ΔT from (3.6). If we neglect the terms $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ in (3.3), set $\Delta\bar{q}_T = 0$ and compute both $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$ with an l-steps Picard iteration, we recover the $CRP(l)$ scheme. In that case, we can reorganize the computations and first compute $\Delta\bar{p}, \Delta T$ and $\Delta\gamma$ and then immediately compute the vector $\Delta\bar{q} = \Delta q_r + \Delta\gamma\,\Delta\bar{q}_\gamma$ with l steps of the Picard scheme (2.38) instead of computing the contributions $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$ separately. The $NPJ(l)$ scheme corresponds to neglecting all terms $V_p^T M V_q$ and $c_s^T V_q$ in (3.6), setting $\Delta\bar{q}_T = 0$ and computing $\Delta\bar{q}_r$ from the Picard scheme (2.12). Hence we can study all variants developed in the previous chapter in this framework. The frameworks also allows to derive new and more powerful and robust variants.

3.2.2 A practical convergence analysis

In this section, we will derive relationships between the error at the end of a Newton–Picard step on one hand, and the residuals of the Q-systems (3.2), the accuracy of the basis and the higher-order terms on the other hand. This relationship is very practical since it shows us how to control the convergence of the Newton–Picard iterations. Hence the analysis in this section allows us to develop more robust variants of our methods and also allows us to explain heuristically why some of the variants derived in the previous chapter may perform badly or even fail.

Let

$$\left[\begin{array}{ccc} Qr_r & Qr_T & Qr_\gamma \end{array}\right] = \left[\begin{array}{ccc} Qr & Qb_T & Qb_\gamma \end{array}\right] + (QMQ-I)\left[\begin{array}{ccc} \Delta q_r & \Delta q_T & \Delta q_\gamma \end{array}\right] \tag{3.8}$$

(with $\Delta q_r = V_q \Delta\bar{q}_r$ etc.) be the N-dimensional residuals of the Q-systems (3.2) (i.e., $V_q\times$ the residuals of (3.2)) Note we use a rather unusual convention for the sign. From a Taylor expansion around the starting point $(x(0), T, \gamma)$ of a Newton-Picard step we get

$$\begin{aligned} Qr(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = Qr(x(0),T,\gamma) + QM\Delta q - \Delta q \\ +QM\Delta p + \Delta T\, Qb_T + \Delta\gamma\, Qb_\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \tag{3.9}$$

After substitution of (3.4) (in terms of the N-dimensional vectors Δq_* instead of the $(N-p)$-dimensional vectors $\Delta\bar{q}_*$) and (3.8) in (3.9), we get

$$\begin{aligned} Qr(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = \\ Qr_r + \Delta T\, Qr_T + \Delta\gamma\, Qr_\gamma + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \tag{3.10}$$

Similarly,

$$\left\{ \begin{aligned} & V_p^T r(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = V_p^T r(x(0),T,\gamma) + V_p^T M V_q \Delta\bar{q} \\ & \qquad +V_p^T M\Delta p - \Delta p + \Delta T\, V_p^T b_T + \Delta\gamma\, V_p^T b_\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2 \\ & s(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = s(x(0),T,\gamma) + c_s^T V_q\Delta\bar{q} + c_s^T\Delta p \\ & \qquad +d_{s,T}\Delta T + d_{s,\gamma}\Delta\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2 \\ & n(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) = n(x(0),T,\gamma) + c_n^T V_q\Delta\bar{q} + c_n^T\Delta p \\ & \qquad +d_{n,T}\Delta T + d_{n,\gamma}\Delta\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \right.$$

By first substituting (3.4) in the right hand side of this expression, then comparing the resulting expression with (3.3) to eliminate most terms and finally premultiplying the first set of equations with V_p, one can easily show that

$$\left\{ \begin{array}{rcl} Pr(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) & = & \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2 \\ s(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) & = & \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2 \\ n(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma) & = & \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{array} \right. \tag{3.11}$$

Neglecting the coupling terms $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ in (3.1) introduces terms $PM\Delta q$, $c_s^T\Delta q$ and $c_n^T\Delta q$ in (3.11). Neglecting $V_q^T b_T$ corresponds to setting $\Delta\bar{q}_T = 0$ and thus $Qr_T = Qb_T$ and neglecting $V_q^T b_\gamma$ corresponds to $Qr_\gamma = Qb_\gamma$. Note that we supposed that (3.3) is solved accurately. Otherwise the residual

$$\left[\begin{array}{c} V_p^T(r+M\Delta q_r) \\ s + c_s^T\Delta q_r \\ n + c_n^T\Delta q_r \end{array}\right] + \left[\begin{array}{ccc} V_p^T M V_p - I_p & V_p^T(b_T + M\Delta q_T) & V_p^T(b_\gamma + M\Delta q_\gamma) \\ c_s^T V_p & d_{s,T} + c_s^T\Delta q_T & d_{s,\gamma} + c_s^T\Delta q_\gamma \\ c_n^T V_p & d_{n,T} + c_n^T\Delta q_T & d_{n,\gamma} + c_n^T\Delta q_\gamma \end{array}\right] \left[\begin{array}{c} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{array}\right] \tag{3.12}$$

has to be added in the right hand side of (3.11).

The relations (3.10) and (3.11) are very practical in conjunction with our $NPGS$ and CNP schemes. At the end of the Newton–Picard step, all the terms in these relations can be computed without new matrix–vector products or time integrations of (1.4) if we suppose that all time integrations and matrix–vector products were computed with enough accuracy during the iterations (i.e., the error of the result of the time integrations and the errors of the matrix–vector products are negligible compared to each of the terms in (3.10) and (3.11)). The evaluation of $Qr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma)$ and $Pr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma)$ requires a time integration of (1.4), but we need to do that time integration anyway to test the convergence. If we would not test for convergence after every iteration step, we would still need to do the time integration at the beginning of the next Newton–Picard step. The matrix–vector products $M\Delta q_*$ needed for the evaluation of the residuals (3.8) are needed to build the P-system (3.3) and if the terms $V_q^T b_T$ or $V_q^T b_\gamma$ in (3.1) are neglected, the corresponding residual is just Qb_T or Qb_γ and requires no matrix–vector product. From the basis iteration process we can recover MV_p, so

$$QM\Delta p = \left(I - V_p V_p^T\right) MV_p \Delta\bar{p}$$

can also be computed easily. The higher-order terms in (3.10) can then be computed since all other terms are known. The evaluation of (3.12) is also not difficult (if needed). Hence the higher-order terms in (3.11) can also be computed. Note once more that we supposed here that the numerical errors made in the evaluation of each of the terms are much smaller than the size of each of those terms. In practice, both (3.10) and (3.11) contain error terms for the matrix–vector products and we cannot use this relation to determine very small higher-order terms.

Let us now discuss the effect of some of the terms in (3.10) and (3.11) in some more detail. This explanation is mathematically not very rigorous but will give us an intuitive explanation of the performance of the methods in computations and was able to explain the observed behavior in experiments. In the next section, we will then apply this analysis to the $NPGS(l)$, $NPJ(l)$, $CNP(l)$ and $CRP(l)$ schemes developed in the previous chapter and show some convergence graphs that illustrate the results.

- The Picard scheme has a linear asymptotic convergence rate of $\left|\mu_{p+1}\right|$ so we expect that $\|Qr_r\| < \|Qr(x(0), T, \gamma)\|$, $\|Qr_T\| < \|Qb_T\|$ and $\|Qr_\gamma\| < \|Qb_\gamma\|$ if we do some Picard iterations using starting values $\Delta q_* = 0$ (since then, $Qr_r = Qr(x(0), T, \gamma)$, etc., at the beginning of the Picard iterations). This is not necessarily true since M is a nonnormal matrix. The residuals (3.8) may initially grow before converging and this may cause divergence of the Newton–Picard step if the number of Picard iterations is fixed in advance. We have never observed this phenomenon in practice and therefore will not pay much attention to it, but it is clear that a truly robust method will have to use an adaptive number of Picard iteration steps.

- The update $(\Delta\bar{p}, \Delta T, \Delta\gamma)$ does not only depend on the P-projection of the initial residual. If the terms $V_p^T MV_q$, $c_s^T V_q$ and $c_n^T V_q$ in (3.1) are not neglected, terms in Δq_r show up in the right hand side of the P-systems (3.3) and (3.6). Δq_r is usually of similar size as $Qr(x(0), T, \gamma)$. If $Qr(x(0), T, \gamma)$ is much larger than $Pr(x(0), T, \gamma)$, the P-system usually has a large right hand side since $\Delta\bar{q}_r$ and thus $V_p^T MV_q\Delta\bar{q}_r$, $c_s^T V_q\Delta\bar{q}_r$ and $c_n^T V_q\Delta\bar{q}_r$ are large. The update $(\Delta\bar{p}, \Delta T, \Delta\gamma)$ also depends on $Qr(x(0), T, \gamma)$ and will be large if either $Pr(x(0), T, \gamma)$ or $Qr(x(0), T, \gamma)$ are large.

- If we neglect the terms $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ in (3.1), terms $V_p^T M \Delta q$, $c_s^T \Delta q$ and $c_n^T \Delta q$ show up in the right hand side of (3.11). These terms will usually prevent the P-projection to decrease much below the Q-projection of the initial residual. The Newton–Picard scheme will converge with $\|Pr\| \gtrapprox \|Qr\|$ during the iterations. Note that this implies relatively large values of $\Delta \bar{p}$, ΔT and $\Delta \gamma$. Hence we will need to use a more accurate basis and compute $\Delta \bar{q}_T$ and $\Delta \bar{q}_\gamma$ with more accuracy from (3.2) or (3.5) in those variants.

- Neglecting Qb_T in (3.1) and setting $\Delta \bar{q}_T = 0$ usually does not cause much problems as long as we do not aim at getting a very large improvement of the residual at each step. At the limit cycle and with a perfect basis, $Qb_T = 0$. The term is often small enough although we noticed some cases where it significantly slowed down the convergence of the Q-projection of the residual. The latter occurs mainly if we also neglect the lower left blocks in (3.1), since this usually implies a larger right hand side of the P-system and thus a larger value for ΔT.

- Neglecting Qb_γ and setting $\Delta \bar{q}_\gamma = 0$ can have a disastrous effect on the convergence. The term Qb_γ can be quite large. If $\Delta \gamma$ is not small, the Q-projection of the residual will not decrease. Usually, $\Delta \gamma$ will be quite large. We can distinguish two cases.

  1. Suppose the lower left blocks in (3.1) are not neglected. Δq_r has usually approximately the same norm as the Q-projection of the initial residual. Furthermore, M is a nonnormal matrix and the norm of $QM\Delta q_r$ is usually comparable to the norm of Δq_r. The right hand side of the P-system (3.3) will usually be at least as large as the Q-projection of the initial residual because of the coupling terms $V_p^T M \Delta q_r$, $c_s^T \Delta q_r$ and $c_n^T \Delta q_r$ or even larger if the P-projection of the initial residual is large. Hence $\Delta \gamma$ will usually be large.

  2. On the other hand, if we neglect the $(2,1)$, $(3,1)$ and $(4,1)$ blocks in (3.1), we are facing two conflicting terms in the convergence analysis. The terms $V_p^T M \Delta q$, $c_s^T \Delta q$ and $c_n^T \Delta q$ that will appear in the right hand side of (3.11) favour a large P-projection of the residual compared to the Q-projection, and hence a large value of $\Delta \gamma$. However, the term $\Delta \gamma \, Qb_\gamma$ favours a larger Q-projection.

Notice that in our testcases, b_γ usually had large components in the P-space and Qb_γ was not too large. Therefore we were often able to compute solutions with variants that neglect the Qb_γ term. It is clear however that robust methods for continuation should not neglect this term in (3.1).

3.3 Convergence behavior of Newton–Picard methods

In the previous chapter, we introduced the $NPGS(l)$, $NPJ(l)$, $CNP(l)$ and $CRP(l)$ methods. From corollary 2.3, we learned that the $NPGS(l)$ and $NPJ(l)$ schemes have linear asymptotic convergence rate $\left|\mu_{p+1}\right|^l$ (where μ_{p+1} is the largest eigenvalue smaller than the threshold ρ for the basis V_p). This result has several limitations.

1. It assumes a perfect basis V_p.
2. It only gives the asymptotic convergence rate and does not give any information about the transient behavior.
3. The proof does not generalize to the $CNP(l)$ and $CRP(l)$ schemes.

In this section, we will study the observed convergence behavior of these methods using (3.10) and (3.11). We will illustrate the discussion with convergence graphs for some schemes.

3.3.1 Comparison of the $NPGS(l)$ and $NPJ(l)$ methods

We will first study the fixed parameter case (i.e., γ is not a variable) and compare the $NPGS(l)$ and $NPJ(l)$ methods. For both methods, $\Delta\bar{q}_T = 0$ (since the term $V_q^T b_T$ is neglected in (2.7)) and $\Delta\bar{q} = \Delta\bar{q}_r$. For the $NPGS(l)$ method, (3.10) and (3.11) reduce to

$$\left\{\begin{array}{l} Qr(x(0)+\Delta x(0), T+\Delta T) = Qr_r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ Pr(x(0)+\Delta x(0), T+\Delta T) = \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ s(x(0)+\Delta x(0), T+\Delta T) = \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \end{array}\right. \tag{3.13}$$

and for the $NPJ(l)$ method we get

$$\left\{\begin{array}{l} Qr(x(0)+\Delta x(0), T+\Delta T) = Qr_r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ Pr(x(0)+\Delta x(0), T+\Delta T) = PM\Delta q + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ s(x(0)+\Delta x(0), T+\Delta T) = c_s^T\Delta q + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \end{array}\right. \tag{3.14}$$

where

$$Qr_r = (QMQ)^l Qr.$$

If we use enough Picard steps, we can assure that $\|Qr_r\| < \|Qr(x(0), T)\|$. The precise value of l needed is problem dependent and may also vary from step to step. Because of the nonnormality of M, $l = 1$ may not always be enough, although we never experienced a growth of Qr_r during the first Picard step in our tests.

Let us first consider the $NPGS(l)$ method.

- Assume the Q-projection of the initial residual is large compared to the P-projection. The norm of Δq is usually comparable to the norm of the Q-projection of the initial residual, and in our applications, we observed that the norm of the term $V_p^T M \Delta q$ is usually of the same order of magnitude as the norm of Δq. Hence the norm of the right hand side of the P-system (2.11) is of the same order of magnitude as the norm of $Qr(x(0), T)$. One can thus expect that Δp and ΔT will be at most of the same order of magnitude as $Qr(x(0), T)$. The terms $\Delta T\, Qb_T$ and $QM\Delta p$ will usually be much smaller than the initial Q-projection of the residual since Qb_T is usually rather small and since V_p, the space containing the vector Δp, is a near invariant subspace of the matrix M. The terms Qr_r and the higher order terms will dominate the final Q-projection of the residual. If the higher order terms are sufficiently small, both projections of the residual can be easily decreased.

- On the other hand, if initially the P-projection of the residual is much larger than the Q-projection, ΔT and Δp will often be quite large. Hence the terms $\Delta T\, Qb_T$ and $QM\Delta p$ will dominate the Q-projection of the residual. In such a step, $\|Qr\|$ may even grow, but if the higher order terms are not too large, the P-projection and the overall residual will decrease. After one or a few steps, we are back in the first scenario.

Hence we can conclude that an $NPGS(l)$ scheme tries to keep the size of the P-projection of the residual at or below the size of the Q-projection.

This is not the case for the $NPJ(l)$ scheme.

- Suppose the Q-projection of the initial residual is of the same size or larger than the P-projection. The only term in the right hand side of the P-system (2.66) is the P-projection of the initial residual, so we can expect that Δp and ΔT will be small. Since $\Delta T Qb_T$ and $QM\Delta p$ in (3.14) will be small, the Q-projection of the residual can decrease a lot if the higher order terms are small enough and if l is large enough. However, Δq will be large (typically of comparable size as the Q-projection of the initial residual). Hence the term $PM\Delta q$ will usually be large (and depending on the phase condition use, $c_s^T \Delta q$ also) and will limit the decrease of the P-projection of the residual. In fact, the norm of this projection of the residual may even grow.

- On the other hand, if initially the Q-projection of the residual is much smaller than the P-projection, Δq and hence also $PM\Delta q$ and $c_s^T \Delta q$ will be small. The P-projection of the residual will decrease a lot provided the higher order terms are small. However, since Δp and ΔT will be relatively large, the reduction of $\|Qr\|$ will be limited by the size of the terms Qb_T and $QM\Delta p$. In fact, $\|Qr\|$ may even grow.

In general, the $NPJ(l)$ scheme tries to keep $\|Qr\|$ much smaller than $\|Pr\|$, but can only do so if the basis is accurate enough. Otherwise the scheme continuously switches between both scenarios and will alternately reduce $\|Qr\|$ and $\|Pr\|$. Therefore the $NPJ(l)$ needs a more accurate basis than the $NPGS(l)$ method for similar performance in terms of number of iteration steps.

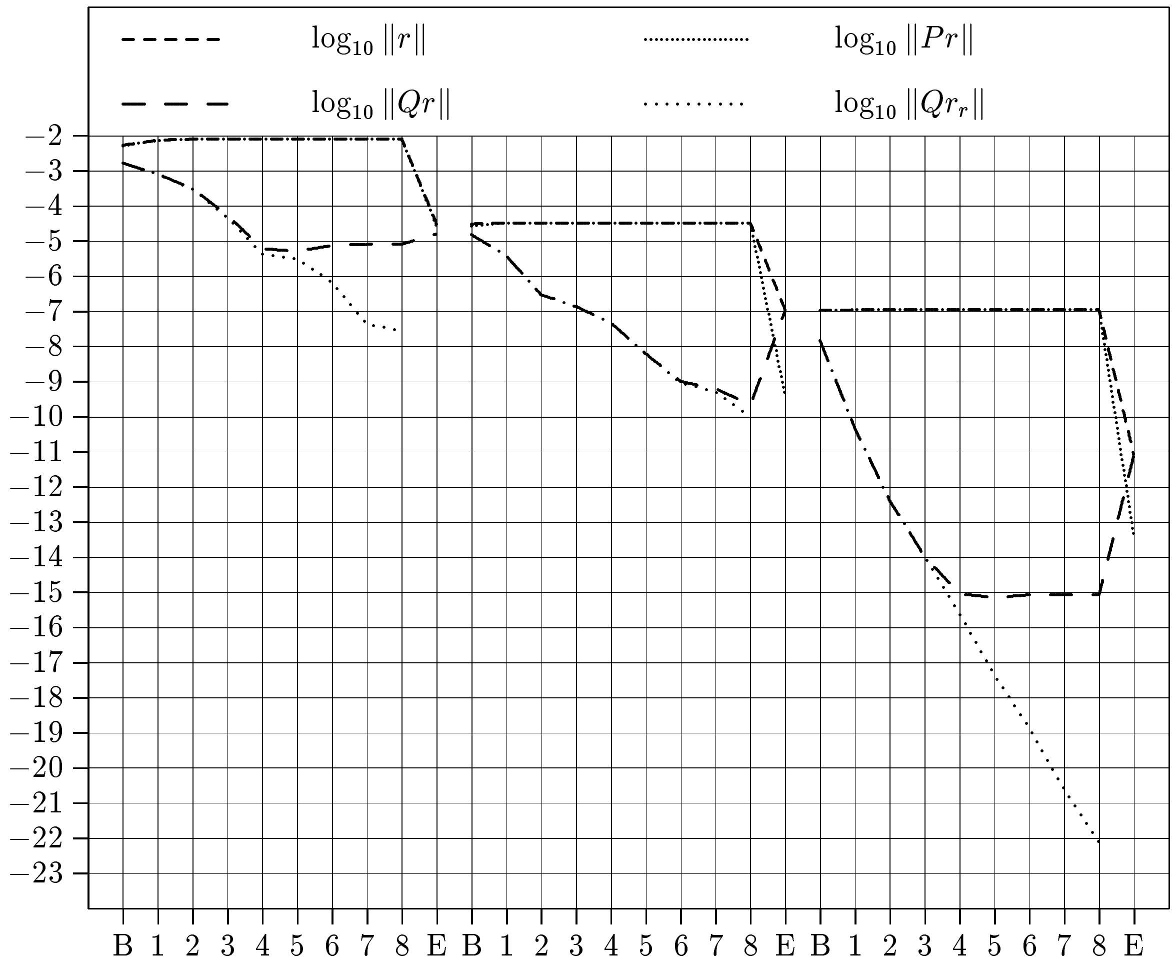


Figure 3.1: Convergence history of the $NPGS(8)$ method. The basis size used for the first step was 2 and for the next steps 3 and 6.

These points are illustrated in Figure 3.1 and 3.2. Figure 3.1 shows the convergence history for the $NPGS(8)$ method and Figure 3.2 for the $NPJ(8)$ method. We took a lot of Picard steps since the differences between both schemes are more pronounced if one tries to get a high convergence speed. Both computations were done for the same periodic orbit—a stable orbit on branch I of the Brusselator model (2.55-2.56) for $L = 0.9158$—and using approximately the same starting value. The letter B marks the beginning of the Newton–Picard step, the numbers 1 to 8 the 8 Picard steps and E the end of the step. We updated the orbit after each Picard step to show how the residual and its projections evolve during the iterations.

In these computations, we allowed the size of the dimension of the P-space to change between every Newton–Picard step. Hence the value of ρ in assumption 2.1 was not fixed during the Newton–Picard iterations. Note also that we only used accurate basis vectors in our basis for the Newton–Picard method and did not use $V_e = V_p$ as in algorithm 2.3. The changes of the basis size explain the jumps of the norms of the projections of the residual between successive Newton–Picard steps.

Notice the approximately linear convergence of the Picard iterations. The convergence factor differs between different Newton–Picard steps since the dimension of the P-space

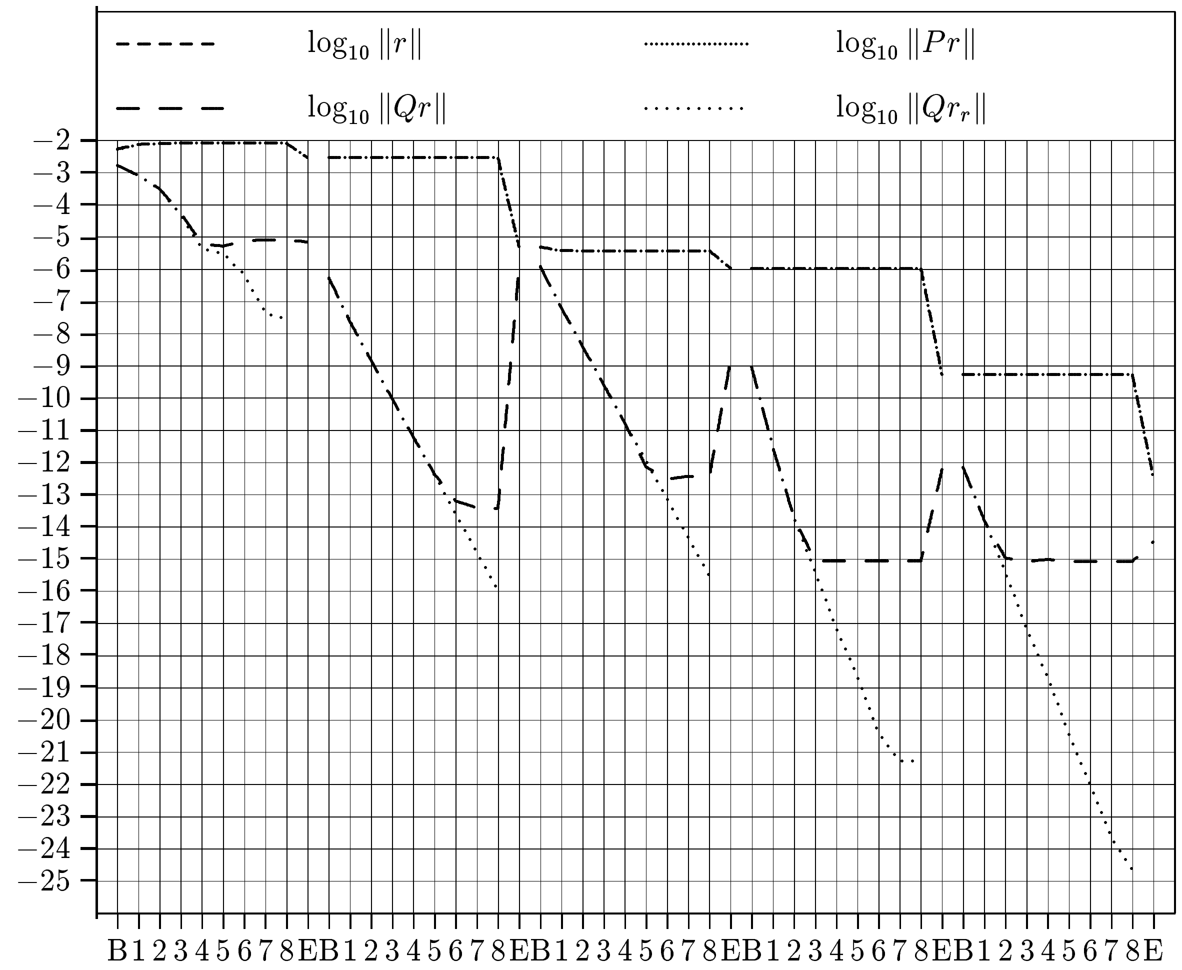


Figure 3.2: Convergence history of the $NPJ(8)$ method. The basis size used for the first step was 2 and for the next steps 5, 5, 6 and 6.

varies. Initially the curves for $\|Qr\|$ and $\|Qr_r\|$ fall together, but after a while, $\|Qr\|$ does not converge any further. This is explained by the higher order terms in the expansion (3.10). In the fourth Newton–Picard step in Figure 3.2 the convergence of $\|Qr_r\|$ breaks off because we have reached the level of roundoff errors of the computation. During the initial Picard steps $\|Pr\|$ changes because of the term $PM\Delta q$. As the Picard iterations converge, Δq settles down to a more or less constant value and so does $\|Pr\|$.

During the Newton step (from 8 to E in the figures), $\|Qr\|$ sometimes increases a lot. In the first Newton–Picard step in Figure 3.1 the higher order terms dominate and no jump is observed. The jump during the second iteration is caused by the term $\Delta T\, Qb_T$ and the jump during the last step by $QM\Delta p$. A similar scenario is observed in Figure 3.2. Initially, higher order terms dominate the Q-projection in (3.14), then $\Delta T\, Qb_T$ and during the final iterations $QM\Delta p$.

In Figure 3.1 we see that the $NPGS(l)$ method converged in three steps. Its convergence speed in this case was clearly limited by the basis accuracy and the size of the term Qb_T. Figure 3.2 for the NPJ method shows a different picture. This scheme needed five steps to converge. We clearly notice the irregular convergence behavior predicted above. During the initial step, the P projection of the residual does hardly decrease because of

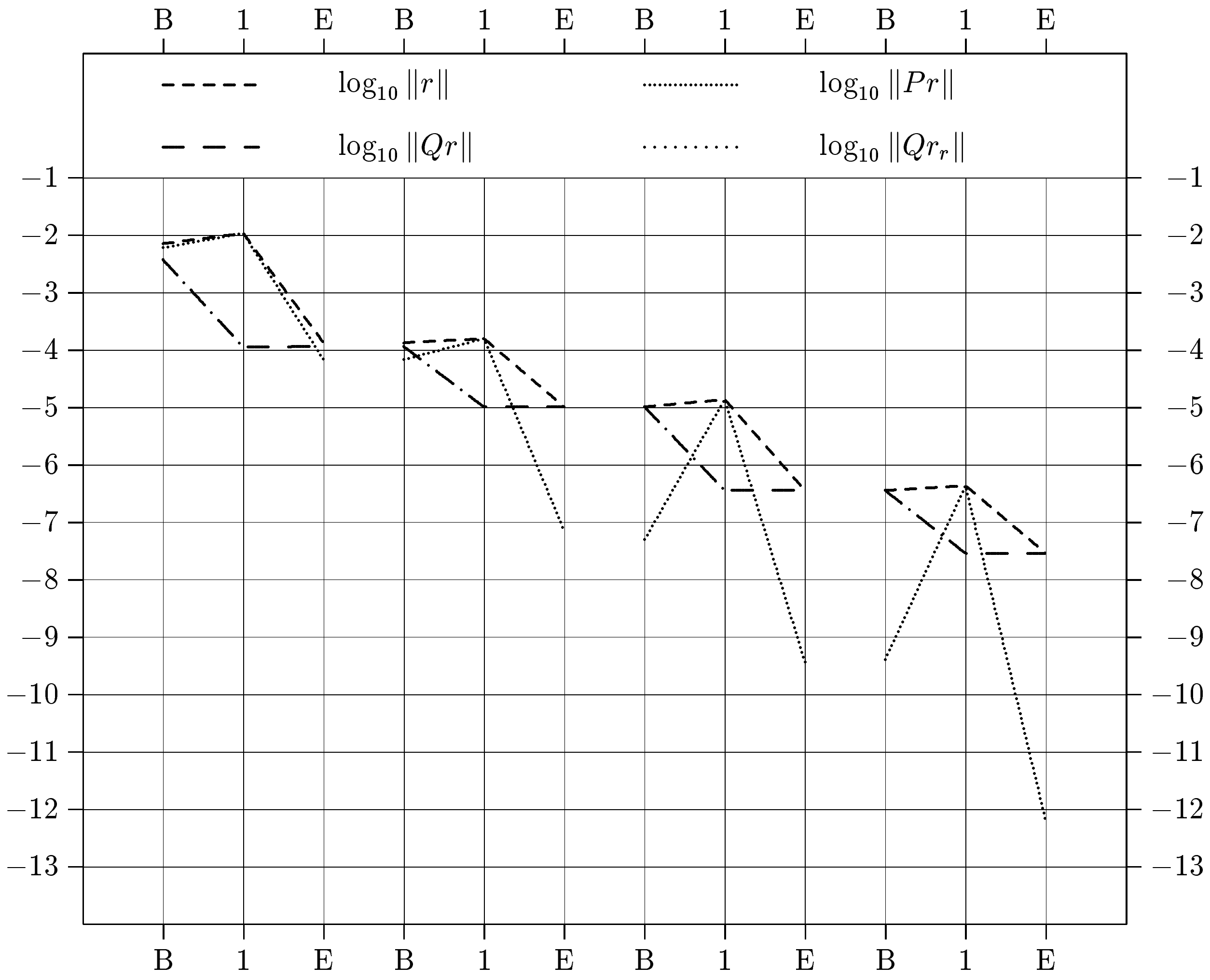


Figure 3.3: Convergence history of the $NPGS(1)$ method. The basis size used was 4 for all steps.

the term $PM\Delta q$, but the Q-projection decreases nicely as it did in the $NPGS$ method. In the second step, $\|\Delta q\|$ is much smaller and the P-residual decreases a lot. However, since Δp and ΔT are large, the terms $\Delta T\, Qb_T$ and $QM\Delta p$ are large and prevent the decrease of $\|Qr\|$. $\|Qr\|$ even grows. The third step shows a similar picture as the first step. By the fourth step, the basis is so accurate and the term Qb_T so small that both projections of the residual can decrease.

Figure 3.3 and 3.4 show results for the $NPGS(1)$ and $NPJ(1)$ method respectively. We again computed a stable solution on branch I of the Brusselator model (2.55-2.56) but now for $L = 0.747$ and $T = 3.273$. (The $NPJ(1)$ method could not compute the solution at $L = 0.9158$.) In Figure 3.3 (for the $NPGS(1)$ method) we notice that the P-projection of the residual is much below the Q-projection. The convergence of the Q-projection is limited by the convergence of the Picard scheme. The convergence of the $NPJ(1)$ method (Figure 3.4) is more regular than for the $NPJ(8)$ method in Figure 3.2. The convergence of the Q-projection is limited by the convergence speed of the Picard iterations, while the convergence of the P-projection is completely governed by

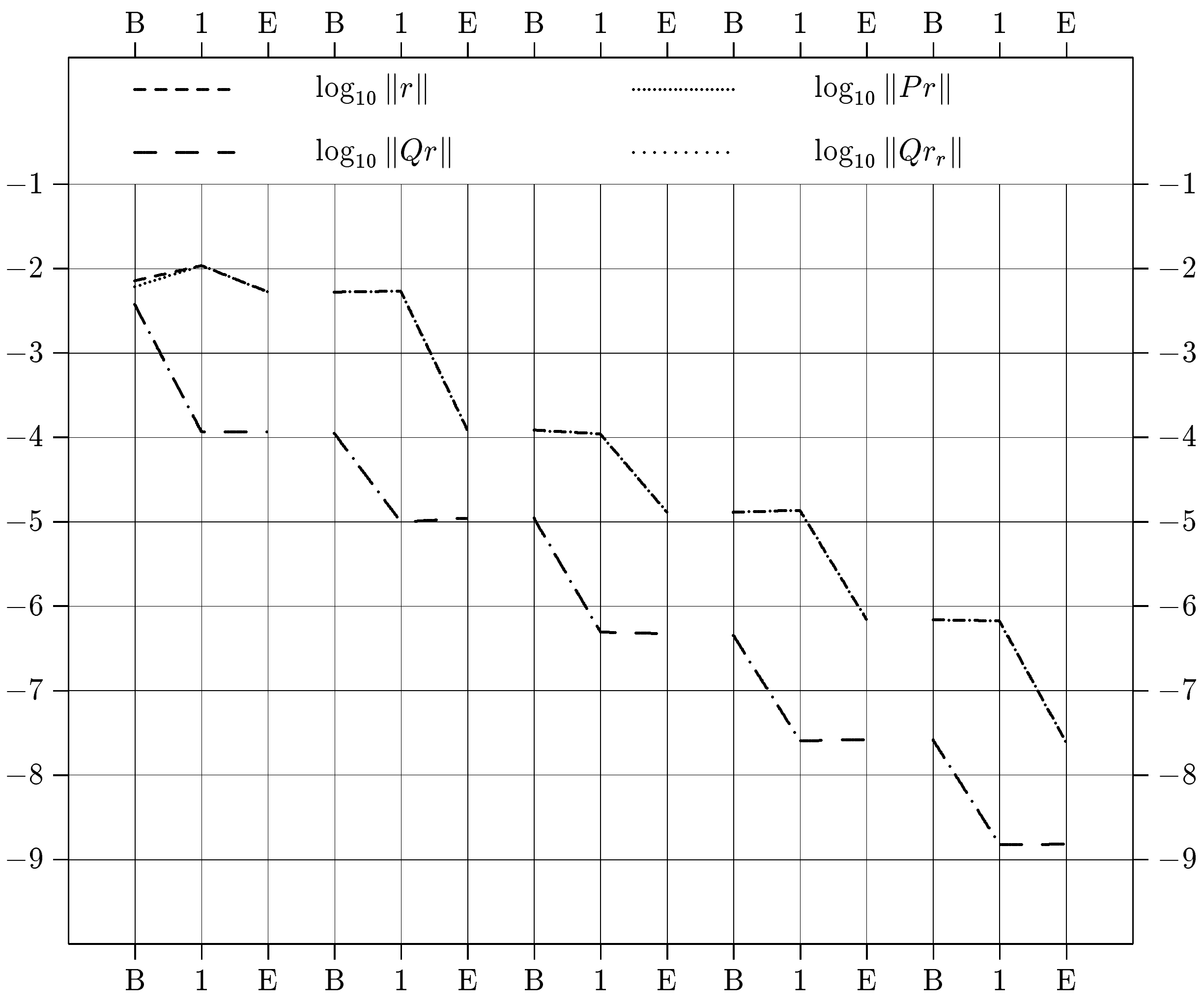


Figure 3.4: Convergence history of the $NPJ(1)$ method. The basis size used was 4 for all steps.

the term $PM\Delta q$. Notice again that the Q-projection of the residual is kept well below the P-projection. Remark also that the $NPJ(1)$ scheme needed six iteration steps while the $NPGS(1)$ scheme could compute the solution in five steps.

Figures 3.1 to 3.4 confirm our predictions. In the $NPGS(l)$ method, $\|Pr\| \lessapprox \|Qr\|$, while the $NPJ(l)$ method tries to keep the Q-projection much below the P-projection, which it can only do if the basis is accurate enough. As we have already seen in the tests at the end of the previous section, the $NPGS(l)$ scheme will profit more from a larger l than the $NPJ(l)$ scheme. The $NPJ(l)$ method needs a more accurate basis before it can profit from the larger value of l.

3.3.2 The $CNP(l)$ and $CRP(l)$ methods

The picture for the continuation variants is not so different. For the CNP scheme we get

$$\left\{\begin{array}{rcl} Qr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& Qr_r+\Delta T\,Qb_T+\Delta\gamma\,Qr_\gamma+ \\ && QM\Delta p+\mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ Pr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ s(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ n(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \end{array}\right. \tag{3.15}$$

and for the CRP method

$$\left\{\begin{array}{rcl} Qr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& Qr_r+\Delta T\,Qb_T+\Delta\gamma\,Qr_\gamma+ \\ && QM\Delta p+\mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ Pr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& PM\Delta q+\mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ s(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& c_s^T\Delta q+\mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ n(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) &=& c_n^T\Delta q+\mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \end{array}\right. \tag{3.16}$$

where

$$\begin{array}{rcl} Qr_r &=& (QMQ)^l Qr, \\ Qr_\gamma &=& (QMQ)^l Qb_\gamma. \end{array}$$

The $CNP(l)$ method will converge keeping $\|Pr\|$ at or below $\|Qr\|$, while the $CRP(l)$ method will keep $\|Qr\|$ below $\|Pr\|$ and converge irregularly if the basis is not accurate enough.

There is however an important difference in the behaviour of the fixed parameter and continuation variants if the nonnormality causes initial growth of the residuals Qr_r or Qr_γ. Note that the residual (3.8) will only grow until it has turned almost completely in the direction of the dominant (generalized) eigenvector(s) of QMQ. From then on, it will decrease again as the Picard iterations proceed.

- Let us first consider the fixed parameter case and suppose l is chosen too small and Qr_r grows in the initial step. Assume that the other terms in (3.13) or (3.14) do not become too large. The residual Qr will turn into the direction of the dominant (generalized) eigenvector(s) of QMQ and after a few Newton–Picard steps the method will start converging. The method will still work for a starting value sufficiently close to the limit cycle.

- This is not necessarily true for the continuation variants. If $Qr_\gamma = (QMQ)^l Qb_\gamma$ is small and only $Qr_r = (QMQ)^l Qb_r$ is large, the method will still converge for a starting value sufficiently close to the limit cycle. If $Qr_\gamma = (QMQ)^l Qb_\gamma$ is large, it will push the residual in the direction of the more dominant (generalized) eigenvector(s) of QMQ. However, the term Qb_γ doesn't change much and so $Qr_\gamma = (QMQ)^l Qb_\gamma$ will remain large in every Newton–Picard step and will slow down or prohibit the convergence of the Q-projection of the residual even if the Picard iterations for $\Delta\bar{q}_r$ converge. We cannot hope that a decrease of $\Delta\gamma$ will help to reduce the term

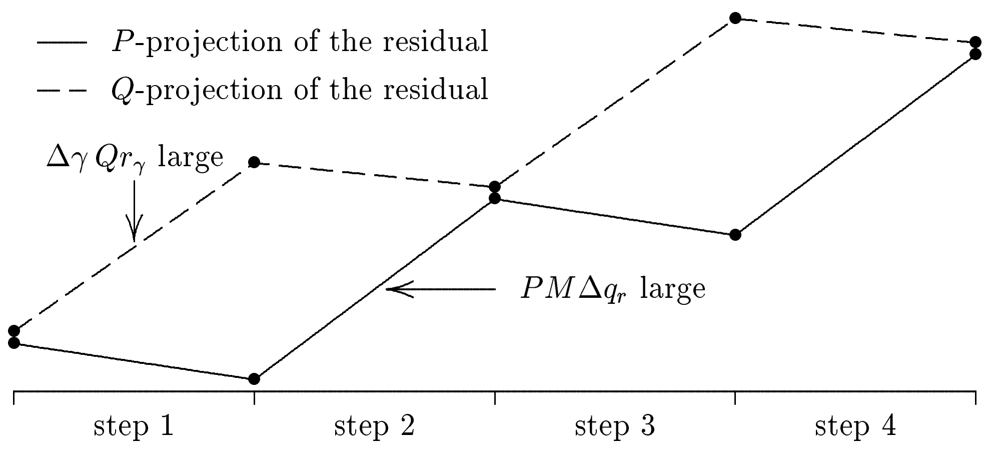

—— P-projection of the residual
— — Q-projection of the residual
$\Delta\gamma\, Qr_\gamma$ large
$PM\Delta q_r$ large
step 1
step 2
step 3
step 4

Figure 3.5: Possible bad behavior of the $CRP(l)$ scheme if $(QMQ)^l Qr_\gamma$ is very large.

$\Delta\gamma\, Qr_\gamma$ in (3.15) and (3.16). Both in the $CRP(l)$ and the $CNP(l)$ scheme, one can get caught in a spiral of ever increasing residuals. This is further explained below.

In the $CRP(l)$ scheme, the P-projection of the residual will usually grow if the Q-projection of the residual at the beginning of the step is sufficiently larger than its P-projection. Hence in the next Newton–Picard step, the right hand side of the P-system will be larger and $\Delta\gamma$ will increase. The Q-projection of the residual will also increase in that step because of the large size of the term $\Delta\gamma\, Qr_\gamma$. This behavior is illustrated in Figure 3.5. Note that this figure is an artificial figure and does not show results computed by our code. In the first step, the Q-projection grows considerably because of the term $\Delta\gamma\, Qr_\gamma$. However, the P-projection of the residual might even decrease a little if $V_p^T M V_q$ is not too large. In the second step, Δq_r is large because the Q–projection at the beginning of the step is very large. Hence the norm of the P-projection will grow significantly during that step. However, since the right hand side of the P-system (which is just the P-projection of the initial residual) is even a little smaller than at the beginning of the first step, $\Delta\gamma$ is smaller than during the first Newton–Picard step and the norm of the Q-projection of the residual decreases slightly. The third step looks again precisely the same as the first one, and so on.

Let us now look at the $CNP(l)$ scheme. If $Qr(x(0), T, \gamma)$ is large, Δq_r and $V_p^T M \Delta q_r$ are usually also large and hence the right hand side of the P-system (3.3) is also large. $Qr(x(0), T, \gamma)$ remains large as long as the right hand side of the P-system and hence $\Delta\gamma$ are large. This is shown in Figure 3.6. This figure is also artificial. In the first step, the Q-projection of the residual increases because the term $\Delta\gamma\, Qr_\gamma$ is large. Since the Q-projection of the initial residual is larger in the second step than in the first step, on can expect that Δq_r will be larger. Hence the right hand side of the P-system is also larger and $\Delta\gamma$ also grows in the second step, and the term $\Delta\gamma\, Qr_\gamma$ becomes even larger.

From the above discussion, we can conclude that both continuation variants can fail for an arbitrary fixed value of l even for a starting value arbitrary close to the limit cycle.

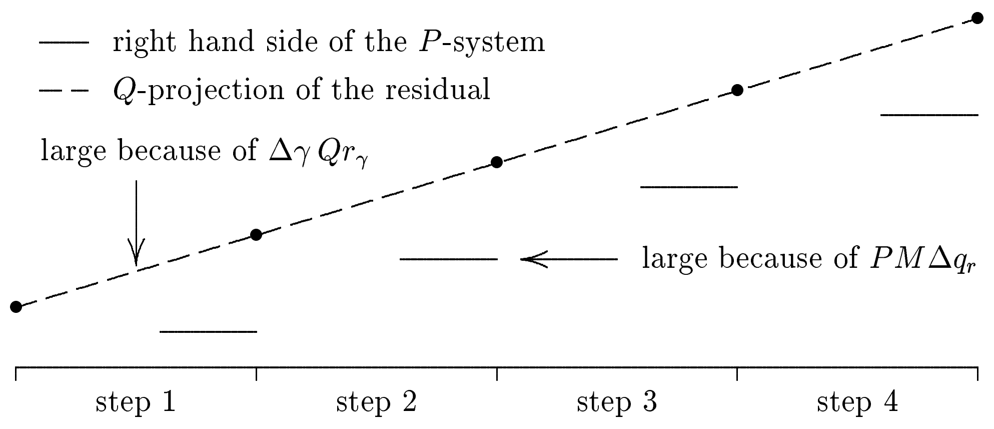

right hand side of the P-system
Q-projection of the residual
large because of $\Delta\gamma\, Qr_\gamma$
large because of $PM\Delta q_r$
step 1
step 2
step 3
step 4

Figure 3.6: Possible bad behavior of the $CNP(l)$ scheme if $(QMQ)^l Qr_\gamma$ is very large.

This indicates that it might be impossible to proof an asymptotic convergence result for the $CNP(l)$ and $CRP(l)$ schemes without additional conditions on the matrix M and/or the vector b_γ. We stress once more that this is a rather hypothetical scenario; we have never observed growth of the residuals (3.8) during the initial Picard steps in our test examples. However, we learn from the above discussion that to obtain a robust code, l should be allowed to vary and one should test for the size of the term Qr_γ.

3.4 Towards robust and powerful methods

In this section, we will first recapitulate the different factors in (3.10) and (3.11) that limit the convergence. Next we will discuss how convergence problems caused by each of those factors can be treated. We will show that information computed from (3.10) and (3.11) at the end of a Newton–Picard step can be used to detect the cause of convergence problems, and that (3.10) and (3.11) can also be used at the beginning of a Newton–Picard step to set convergence goals for the iterative solvers for each of the vectors $\Delta\bar{q}_*$ in (3.2) and for the basis iterations. Exploiting the information contained in (3.10) and (3.11) thus leads to more robust methods.

Factors limiting convergence. The algebraic framework of section 3.2 opens the door to new and powerful extensions of the $NPGS$ and CNP methods. From (3.10) and (3.11) it is clear that 5 factors limit the improvement of the residual that can be obtained from a given Newton–Picard step

1. Higher-order terms neglected by the Newton linearization from which our method departs put limits on both the Q- and the P-projection of the final residual.

2. Inaccuracy of the basis (the term $QM\Delta p$) limits the Q-projection of the final residual.

3. The accuracy of the solutions of the Q-systems (3.2) or (3.5) influences the Q-projection of the final residual.

4. If the P-system (3.3) or (3.6) is solved inaccurately, the term (3.12) shows up in (3.11) and limits the convergence of Pr, s and n.
5. Numerical errors, especially in the evaluation of the matrix–vector products and derivatives, but also in other operations, can also limit the obtainable improvement.

Curing convergence problems. Robust methods will have to control each of these elements.

1. Higher-order terms can be controlled using damping strategies (see, e.g., [24]). Note that this is not very important if the solver is used inside a continuation code. If the higher order terms are too large, the solver fails and the predictor decreases the steplength and generates a new and better starting value. Controlling the higher order terms by damping is important to compute a single solution or to start the continuation run if good starting values are not available.
2. Influence of the basis inaccuracy. An inaccurate basis can limit the improvement of the Q-projection of the residual. The strategy used in algorithm 2.3—using the complete basis V_e used in the subspace iterations without testing the accuracy of the additional vectors to construct the projectors for the Newton–Picard step–was not a very good idea. Instead we should select an accurate dominant subset of the basis V_e. The error criterion discussed in section 2.4.4 is an important tool in controlling this term since it measures the worst case for $||QM\Delta p||_2$ given Δp has unit norm.
3. The accuracy of the solution of the Q-subsystems. In the algebraic framework that is the basis for our extensions, there is no need to use the same iterative method or the same number of iteration steps for each of the right hand sides of (3.2). In fact, to avoid problems with initial growth of the residuals (3.8), we suggest to use a variable number of Picard steps and to test for the convergence after each iteration step. The convergence test is very cheap, since the result of the matrix–vector product that is needed for the test can be used in the next Picard step or to build the P-system. One can also consider to solve the Q-systems using more advanced iterative methods. We will briefly discuss the use of the GMRES method as an alternative to Picard iterations. We will also discuss the choice of good starting values for the iterative methods.
4. The accuracy of the solution of the P-system. As an alternative to Gauss elimination, we propose the use of the least-squares method based on the singular value decomposition (SVD). This method is more expensive than Gauss elimination, but the solution of the P-system represents only a negligible part of the total iteration cost. The robustness of our method increased significantly when we switched to the SVD-based approach, particularly in the neighbourhood of pitchfork bifurcations. In the SVD approach, we can omit the parametrizing equation and even the phase condition. This simplifies the implementation. However, we sometimes noticed problems with some DDE problems if the phase condition was omitted and also noticed problems at larger continuation stepsizes if the parametrizing equation was omitted.

5. Numerical errors can only be controlled by using high-quality time integration techniques and robust, stable numerical methods at each step in the algorithm.

Relation (3.10) and (3.11) have a double use. They can be used a priori to determine convergence thresholds for the basis and the various iterative solvers for the Q-systems and after the iteration step to analyze the convergence behaviour.

A posteriori use of (3.10) and (3.11). At the end of a Newton–Picard iteration step, the new residual is computed. If the convergence goal for the iteration step is not reached, (3.10) and (3.11) can be used to analyze what went wrong and to take the appropriate action.

- If the higher order terms caused divergence, a damping strategy should be used or the Newton–Picard solver should return with an error condition and rely on the continuation code to generate a better starting value (by decreasing the stepsize).
- If the size of $QM\Delta p$ causes problems, we should consider to improve the basis quality and then check the solution of the Q-systems. Refining the basis is expensive and should only be done if the Newton–Picard step diverged or if a sufficient additional decrease of the residual is possible.
- If one of the residuals (3.8) caused problems, we can do some more iteration steps with the iterative method we used.

Remark that we can already test the size of the various residuals (3.8) and $QM\Delta p$ in (3.10) before the computation of the new residual $r(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma)$ to check whether these terms have decreased enough. The early detection of problems can avoid a superfluous integration of (1.4). This is particularly important if the integration of (1.4) is expensive compared to the matrix–vector products.

A priori use of (3.10) and (3.11). At the beginning of a Newton–Picard iteration step, we can use (3.10) and (3.11) to determine the convergence thresholds for the basis computation process and the iterative solvers for (3.2) and (3.5). Based on the initial residual and estimates for the higher order terms, the goal for the residual at the end of the Newton–Picard step is determined. Then the size of Δp, ΔT and $\Delta\gamma$ is estimated. Based on these estimates and the goal for the step, it is possible to derive convergence thresholds for the iterative solver and basis computation process from (3.10). We will now discuss two elements from this strategy in a little more detail. First, we will briefly discuss the estimation of Δp, ΔT and $\Delta\gamma$. Next we will propose a strategy to determine the goal for the residual at the end of a Newton–Picard step.

Estimating the size of Δp, ΔT and $\Delta\gamma$. Several strategies can be developed to estimate the size of Δp, ΔT and $\Delta\gamma$.

A first option is to solve the P-system of the previous step with the P-projection of the initial residual of the current step as its right hand side. The reasoning behind this strategy is that the matrix of the low-dimensional subsystem usually does not change a lot between two successive iterations (at least if the basis size does not change). This strategy cannot be used for the first Newton–Picard step. Furthermore, the strategy only makes sense if the Q-projection of the initial residual and thus Δq_r is sufficiently low compared to the P-projection since this strategy neglects the term $V_p^T M \Delta q_r$ in the right hand side of the P-system. (Remind that in our experiments, the size of the Q-projection of the initial residual, the size of Δq_r and the size of the term $V_p^T M \Delta q_r$ were usually of the same order of magnitude.) The strategy also does not make much sense if the size of the basis is changed in the Newton–Picard step since this changes the relative size of the P- and Q-projections of the residual. Hence this strategy is hard to use and sometimes unreliable.

An alternative is to estimate Δp, ΔT and $\Delta\gamma$ based on the vectors b_T and b_γ and the residual r. A very rough estimate can be obtained from looking at the system (2.19). The size of Δp and the total update $\Delta x(0)$ is usually of the same order of magnitude, even if the Q-projection of the residual at the beginning of the step is much larger than its P-projection. Because of the term $V_p^T M \Delta q_r$ in the right hand side of the P-system (3.3), the initial residual of the Newton–Picard step and the right hand side of the P-system have usually about the same size. We estimate that the norm of Δp is of the same order of magnitude as the norm of the initial residual, i.e.,

$$\|\Delta p\| \approx \|r\| . \tag{3.17}$$

This estimate may be too pessimistic (i.e., overestimate the norm of $\Delta x(0)$) if the limit cycle is very unstable. In the latter case, a small error in the point $x(0)$ can cause a large residual. The estimates for ΔT and $\Delta\gamma$ are derived from comparing the right hand side of (2.19) with the columns corresponding to ΔT and $\Delta\gamma$, e.g.,

$$\begin{aligned} \Delta T &\approx \frac{\|r\|}{\|b_T\|} \\ \Delta\gamma &\approx \frac{\|r\|}{\|b_\gamma\|}. \end{aligned} \tag{3.18}$$

We stress once more that these estimates are very crude and more work needs to be done to develop better estimates.

Setting the convergence goal. Setting the convergence goal for the Newton–Picard step is even harder. We determine a factor ρ_{goal} expressing the desired improvement of the residual, i.e., at the end of the Newton–Picard step we wish to obtain

$$\|r(x(0)+\Delta x(0), T+\Delta T, \gamma+\Delta\gamma)\| \approx \rho_{goal} \|r(x(0), T, \gamma)\| .$$

To determine ρ_{goal}, we proceed as follows. First, we try to estimate the higher order terms in (3.10) and (3.11) based on the results of the previous Newton–Picard steps. If the higher order terms are larger than the initial residual, one has to use damping. In this case we require that the linearized residual should improve with a factor ρ_{req}. ρ_{req} is a user-determined parameter. In fact, we always assure $\rho_{goal} \le \rho_{req}$, even if the higher order terms are small. If the estimated higher order terms are smaller than $\rho_{req} \|r(x(0), T, \gamma)\|$

we have more options. By changing the strategy to determine ρ_{goal}, the Newton–Picard iterations can show either linear or quadratic convergence behaviour (or any behaviour in between). Quadratic convergence can be obtained by reducing $\|r\|$ at each step to the level of the estimated higher order terms. This requires a lot of work at each iteration step, but reduces the number of Newton–Picard iteration steps. This strategy makes sense if the matrix–vector products are much cheaper than the integration of the nonlinear system (1.4). We may also settle for linear convergence of the Newton–Picard iterations, or something in between linear and quadratic convergence, say

$$\rho_{goal} = \rho_{req}^{1-\omega}\xi^{\omega}, \quad 0 \leq \omega \leq 1,$$

where ξ is the ratio of the estimated higher order terms to the initial residual. $\omega = 0$ corresponds to linear convergence with convergence rate ρ_{req} and $\omega = 1$ corresponds to quadratic convergence. Once we have determined ρ_{goal}, we set our goal for the Q-projection of the final residual and for the individual residuals (3.8) and the basis iterations. We also have to put a lower limit on ρ_{goal} since the obtainable improvement is limited by the errors in the computation of the matrix–vector products and not only by the higher order terms, and since a wrong estimate for the higher order terms may lead to an unfeasible goal. Hence we limit

$$\rho_{best} \leq \rho_{goal} \leq \rho_{req}$$

where ρ_{best} is a user-determined parameter. Furthermore, we should avoid to do unnecessary work and make sure that the residual is not decreased much below the convergence requirement for the Newton–Picard procedure.

Minimizing the total cost. There is another important parameter in the Newton–Picard methods that needs to be determined, namely the parameter ρ in assumption 2.1 and 2.4. Lower values of ρ result in more basis vectors and thus more work to construct the basis, but a faster convergence of the Picard iterations for the various $\Delta\bar{q}_{\cdot}$'s. The ideal strategy for the choice of ρ and ρ_{goal} should minimize the total cost for the computation of the periodic orbit. However, it is extremely difficult or even impossible to predict the total cost of the iterations in advance in terms of the various parameters. A relaxation of this goal is to try to minimize the ratio of the computation cost to the improvement of the residual at each step. This goal is also too ambitious since we cannot determine in advance the precise cost of the basis iterations and the Picard iterations. This cost depends on unknown information, the full spectrum of eigenvalues and eigenvectors of M. One can only try to develop reasonable estimates to approximate this goal, and this is an area in which much work remains to be done.

Many of the aspects discussed in this section have been implemented in our multiple shooting code (see chapter 6). We used a simple strategy for the convergence goal where ρ, ρ_{req}, ρ_{best} and ω are set by the user and did not try to minimize the computation cost automatically. Some of the ideas in this section have also been implemented in our single shooting code to test if our claims are valid.

3.5 Advanced methods for solving the Q-system

From the discussion in the previous sections, we can deduce the following requirements for the Q-system solver:

1. The Q-system solver must be able to use a variable number of steps and to test for convergence.
2. The solver must be able to start with a nonzero starting value. We will see that it is possible to derive good starting values for $\Delta\bar{q}_r$ in (3.2).
3. The solver should also output the terms $M\Delta q_*$ or $V_p^T M\Delta q_*$ for the construction of the P-system. These terms can be recovered easily from the Q-system solver. We save a matrix–vector product for the construction of the P-system in doing so.
4. During the a posteriori analysis we may decide that the solution of one of the Q-systems needs further refinement. Therefore the Q-system solver should generate the necessary information to make the restart as cheap as possible.

We will first discuss the necessary changes to the Picard iteration (2.50). Then we will briefly explain why we think that the GMRES method [103] is often a better alternative to the Picard iteration scheme. At the end of this section we will discuss the choice of the starting values for the various Q-systems (3.2) or (3.5).

3.5.1 The Picard scheme

The Picard iteration is given by

$$\begin{cases} \Delta q \leftarrow \Delta q^{[0]}, \\ \Delta q \leftarrow QM\Delta q + Qr_* \ \textbf{until} \text{ convergence} \end{cases} \tag{3.19}$$

where r_* is the residual r or one of the vectors b_T or b_γ. The scheme (3.19) has converged if

$$\|Q\,(r_* + (M - I)\Delta q)\| \le \varepsilon,$$

where ε is the convergence threshold. This test is basically free since the result of the matrix–vector product $M\Delta q$ can be used in the next Picard step or to construct the P-system. Our algorithm must take into account two different start scenarios:

1. A starting vector Δq is given, but $M\Delta q$ is not yet known.
2. Both Δq and $M\Delta q$ are given. This occurs if the starting value is 0 since then $M\Delta q = 0$ and at a restart.

Incorporating these remarks in (3.19) leads to the following algorithm:

Algorithm 3.1 *Advanced Picard iteration for single shooting.*

```
    Input
        Starting value $\Delta q^{[0]}$.
        Optional: $M\Delta q^{[0]}$.
        Convergence threshold $\varepsilon$.
        Routine to compute $Mv$.
        Basis $V_p$.
        Right hand side $r$.
    Output
        $\Delta q = V_q \Delta\bar{q}$ with $\Delta\bar{q}$ an approximate solution to $\left(V_q^T M V_q - I_q\right)\Delta\bar{q} = -V_q^T r$.
        $M\Delta q$.
    begin
        if $\Delta q^{[0]} = 0$ then
            $\Delta q \leftarrow 0$; $M\Delta q \leftarrow 0$;
        else if $M\Delta q^{[0]}$ is known
            $\Delta q \leftarrow \Delta q^{[0]}$; Set $M\Delta q = M\Delta q^{[0]}$.
        else
            $\Delta q \leftarrow \Delta q^{[0]}$;
            Compute $M\Delta q$.
        endif
        $RES \leftarrow \left(I - V_p V_p^T\right)(r + M\Delta q - \Delta q)$
        while $\|RES\| > \varepsilon$ do
            $\Delta q \leftarrow \left(I - V_p V_p^T\right)(M\Delta q + r)$
            Compute $M\Delta q$.
            $RES \leftarrow \left(I - V_p V_p^T\right)(r + M\Delta q - \Delta q)$
        end while
    end
```

Note that this algorithm may return without actually updating Δq if the initial solution is accurate enough. This is important if Δq_T is being computed since the term Qb_T is often small enough so $\Delta q_T = 0$ is sufficient. In the actual implementation, we also stop the iterations if a predetermined maximum number of iterations is exceeded to protect against infinite loops if the Picard iteration cannot reach the desired residual norm. l steps of this algorithm require l matrix–vector products if $M\Delta q^{[0]}$ is known and $l+1$ products otherwise. This is not in contradiction with our analysis in the previous chapter where we stated that the l-step Picard iteration (2.50) required $l-1$ matrix–vector products since (2.50) supposes that the starting value $\Delta q^{[0]} = 0$ (and thus $M\Delta q^{[0]}$ is known) and does not compute $M\Delta q$ at the end, requiring an additional matrix–vector product.

3.5.2 The GMRES method

Consider the system

$$\left(V_q^T M V_q - I_q\right)\Delta\bar{q}_* = -V_q^T r_*, \tag{3.20}$$

where r_* is one of r, b_T or b_γ. The l-step Picard iteration constructs the approximation

$$\Delta\bar{q}_* = \left(\sum_{i=0}^{l-1}\left(V_q^T M V_q\right)^i\right) V_q^T r_*.$$

Hence $\Delta\bar{q}_* \in \mathcal{K}_l\left(V_q^TMV_q, -V_q^Tr_*\right)$, the l-dimensional Krylov subspace of $V_q^TMV_q$ based on $-V_q^Tr_*$. Krylov subspaces are shift-invariant, so

$$\mathcal{K}_l\left(V_q^TMV_q, -V_q^Tr_*\right) = \mathcal{K}_l\left(V_q^TMV_q - I_q, -V_q^Tr_*\right).$$

The l-step Picard iteration uses a fixed linear combination of powers of $V_q^TMV_q$ (or $V_q^TMV_q - I_q$) to construct $\Delta\bar{q}_*$, independent of the right hand side of (3.20). In fact, we can write

$$\Delta\bar{q}_* = \psi_{P,l}\left(V_q^TMV_q - I_q\right)\left(-V_q^Tr_*\right),$$

with $\psi_{P,l}$ the polynomial of degree $l-1$ given by

$$\psi_{P,l}(x) = -\sum_{i=0}^{l-1}(x+1)^l.$$

In other words, the l-step Picard iteration uses the fixed approximation

$$\psi_{P,l}\left(V_q^TMV_q - I_q\right)$$

to the inverse of $\left(V_q^TMV_q - I_q\right)$. The GMRES($l$) method for (3.20) will compute a different polynomial $\psi_{G,l}(x)$ and

$$\Delta\bar{q}_* = \psi_{G,l}\left(V_q^TMV_q - I_q\right)\left(-V_q^Tr_*\right) \in \mathcal{K}_l\left(V_q^TMV_q - I_q, -V_q^Tr_*\right).$$

The approximation $\psi_{G,l}\left(V_q^TMV_q - I_q\right)$ to the inverse of $\left(V_q^TMV_q - I_q\right)$ is in some sense fine-tuned for the right hand side $-V_q^Tr_*$: the GMRES(l) method will minimize the residual of (3.20) over the search space $\mathcal{K}_l\left(V_q^TMV_q - I_q, -V_q^Tr_*\right)$. Hence the GMRES($l$) method will always produce an approximate solution to (3.20) with a residual that is lower or at most equal to the residual for the solution generated by the l-steps Picard scheme and is a good replacement for the Picard iterations in our method. The main disadvantage of the method is the need to store a basis for the $(l+1)$-dimensional Krylov subspace $\mathcal{K}_{l+1}\left(V_q^TMV_q - I_q, -V_q^Tr_*\right)$. If ρ is large, the convergence can be slow and we may need to use restarted GMRES instead.

The GMRES(l) method for (3.20) will first compute an orthonormal basis $\bar{V}_{l+1} = \left[\begin{array}{ccc}\bar{v}_1 & \cdots & \bar{v}_{l+1}\end{array}\right]$ for the Krylov subspace $\mathcal{K}_{l+1}\left(V_q^TMV_q - I_q, -V_q^Tr_*\right)$ and an $(l+1)\times l$ upper Hessenberg matrix $\bar{H}_l$ satisfying

$$\bar{v}_1 = -\frac{1}{\left\|V_q^Tr_*\right\|}V_q^Tr_*$$

and

$$\left(V_q^TMV_q - I_q\right)\bar{V}_l = \bar{V}_{l+1}\bar{H}_l. \tag{3.21}$$

Then the least-squares solution of

$$\bar{H}_ly = \left\|V_q^Tr_*\right\|e_1 \tag{3.22}$$

(with e_1 the $(l+1)$-dimensional vector with a one on the first position and zeros on the others) is computed and finally

$$\Delta\bar{q}_* = \bar{V}_ly.$$

If a nonzero starting value $\Delta\bar{q}_*^{[0]}$ is given, we instead construct a basis for the Krylov subspace

$$\mathcal{K}_{l+1}\left(V_q^T M V_q - I_q, -V_q^T r_* - \left(V_q^T M V_q - I_q\right)\Delta\bar{q}_*^{[0]}\right)$$

and compute $\Delta\bar{q}_*$ from

$$\Delta\bar{q}_* = \Delta\bar{q}_*^{[0]} + \bar{V}_l y.$$

We will make a small but important modification to the classical implementation of the GMRES method. The reason for this modification will become clear when we discuss GMRES for the multiple shooting method in chapter 6. Instead of computing the relationship (3.21), we will compute an orthonormal basis $\bar{V}_{l+1}$ for the Krylov subspace $\mathcal{K}_{l+1}\left(V_q^T M V_q, -V_q^T r_*\right)$ (or $\mathcal{K}_{l+1}\left(V_q^T M V_q, -V_q^T r_* - \left(V_q^T M V_q - I_q\right)\Delta\bar{q}_*^{[0]}\right)$) and an $(l+1)\times l$ upper Hessenberg matrix satisfying

$$V_q^T M V_q \bar{V}_l = \bar{V}_{l+1}\bar{H}_l.$$

Note that

$$\left(V_q^T M V_q - I_q\right)\bar{V}_l = \bar{V}_{l+1}\bar{H}_l - \bar{V}_l = \bar{V}_{l+1}\bar{H}_l - \bar{V}_{l+1}\bar{I}_l = \bar{V}_{l+1}(\bar{H}_l - \bar{I}_l),$$

where $\bar{I}_l$ is the $(l+1)\times l$ matrix with ones on its main diagonal and all other elements equal to 0. The vector y is computed as the least-squares solution of

$$(\bar{H}_l - \bar{I}_l)y = \beta e_1 \text{ with } \beta = \left\|V_q^T r_*\right\|$$

(or $\beta = \left\|-V_q^T r_* - \left(V_q^T M V_q - I_q\right)\Delta\bar{q}_*^{[0]}\right\|$ for the nonzero starting value $\Delta\bar{q}_*^{[0]}$). There are several methods to solve this least-squares problem, e.g., the method used in [103] based on Givens rotations or the normal equations

$$(\bar{H}_l - \bar{I}_l)^T(\bar{H}_l - \bar{I}_l) = \beta(\bar{H}_l - \bar{I}_l)^T e_1$$

which can be solved using Gauss elimination. The approach based on Givens rotations is preferable from a numeric point of view. In the normal equations approach the condition number of the linear system is the square of the condition number of $(\bar{H}_l - \bar{I}_l)$.

In terms of $(N-p)$-dimensional vectors, the GMRES(l) algorithm is given by:

Algorithm 3.2 *GMRES(l) for (3.20).*

```
Input
    Starting value $\Delta\bar{q}^{[0]}$. Optionally, $MV_q\Delta\bar{q}^{[0]}$.
    The basis $V_q$, a routine to compute $Mv$ and the right hand side $r$.
Output
    An approximates solution $\Delta\bar{q}$ to $\left(V_q^T M V_q - I_q\right)\Delta\bar{q} = -V_q^T r$.
    $V_q^T M V_q \Delta\bar{q}$.
begin
    $\bar{r} \leftarrow -V_q^T r - \left(V_q^T M V_q - I_q\right)\Delta\bar{q}^{[0]}$
    $\beta \leftarrow \|\bar{r}\|$
    $v_1 \leftarrow \bar{r}/\beta$
    for $i = 1$ to $l$ do
```

```
            $\bar{v}_{i+1} \leftarrow V_q^T M V_q \, \bar{v}_i$
            **for** $j = 1$ **to** $i$ **do**
                $h_{j,i} \leftarrow <\bar{v}_j, \bar{v}_{i+1}>$
                $\bar{v}_{i+1} \leftarrow \bar{v}_{i+1} - h_{j,i}\bar{v}_j$
            **end for**
            $h_{i+1,i} \leftarrow \|\bar{v}_{i+1}\|$
            $\bar{v}_{i+1} \leftarrow \bar{v}_{i+1}/h_{i+1,i}$
        **end for**
        Compute the least-squares solution of $(\bar{H}_l - \bar{I}_l)y = \beta e_1$.
        $\Delta\bar{q} \leftarrow \Delta\bar{q}^{[0]} + \bar{V}_l y$
        $V_q^T M V_q \Delta q \leftarrow V_q^T M V_q \Delta\bar{q}^{[0]} + \bar{V}_{l+1}\bar{H}_l y$
    **end**
```

This algorithm requires l matrix–vector products if $V_q^T M V_q \Delta q^{[0]}$ is known and $l+1$ otherwise, precisely the same as the l-step Picard iteration. To be practical, we need to rewrite algorithm 3.2 using only N-dimensional vectors. This is easily done by premultiplying each of the lines containing $(N-p)$-dimensional vectors in algorithm 3.2 with V_q and setting $v_i = V_q\bar{v}_i$ and $V_i = V_q\bar{V}_i$. To be able to recover $M\Delta q$ we need to compute and store the terms $w_i = V_p^T M v_i$ during the basis construction iterations. This requires little memory. We can then compute

$$V_p^T M \Delta q = V_p^T \left(M\Delta q^{[0]}\right) + V_p^T M V_l y = \left[\begin{array}{ccc} w_1 & \cdots & w_l \end{array}\right] y$$

and

$$\begin{aligned} M\Delta q &= M\Delta q^{[0]} + V_q V_q^T M V_l y + V_p V_p^T M V_l y \\ &= M\Delta q^{[0]} + V_{l+1}\bar{H}_l y + V_p \left[\begin{array}{ccc} w_1 & \cdots & w_l \end{array}\right] y. \end{aligned}$$

Instead of using a fixed number of iterations, we should stop the iterations as soon as the residual has decreased below a user-determined level. Note that

$$\begin{aligned} \left\|-V_q^T r_* - \left(V_q^T M V_q - I_q\right)\Delta\bar{q}_*\right\|_2 &= \left\|\beta\bar{v}_1 - \bar{V}_{l+1}\left(\bar{H}_l - \bar{I}_l\right)y\right\|_2 \\ &= \left\|\bar{V}_{l+1}\left(\beta e_1 - \left(\bar{H}_l - \bar{I}_l\right)y\right)\right\|_2 \\ &= \left\|\beta e_1 - \left(\bar{H}_l - \bar{I}_l\right)y\right\|_2, \end{aligned}$$

so checking the convergence is easy and cheap. With the above modifications, algorithm 3.2 turns into

Algorithm 3.3 *Advanced GMRES(l) method for single shooting.*

```
    **Input**
        Starting value $\Delta q^{[0]}$.
        Optional: $M\Delta q^{[0]}$.
        Convergence threshold $\varepsilon$.
        Routine to compute $Mv$.
        Basis $V_p$.
        Right hand side $r$.
```

```
Output
    $\Delta q = V_q \Delta\bar{q}$ with $\Delta\bar{q}$ an approximate solution to $\left(V_q^T M V_q - I_q\right)\Delta\bar{q} = -V_q^T r$.
    $M\Delta q$.
begin
    if $\Delta q^{[0]} = 0$ then
        $\Delta q \leftarrow 0$; $M\Delta q \leftarrow 0$;
    else if $M\Delta q^{[0]}$ is known
        $\Delta q \leftarrow \Delta q^{[0]}$; Set $M\Delta q = M\Delta q^{[0]}$.
    else
        $\Delta q \leftarrow \Delta q^{[0]}$;
        Compute $M\Delta q$.
    endif
    $\tilde{r} \leftarrow -\left(I - V_p V_p^T\right)(r + M\Delta q - \Delta q)$
    $\beta \leftarrow \|\tilde{r}\|$
    $v_1 \leftarrow \tilde{r}/\beta$
    if $\beta < \varepsilon$ then return
    $i = 0$
    repeat
        $i \leftarrow i + 1$
        Compute $w_i \leftarrow V_p^T M v_i$ and $v_{i+1} \leftarrow (I - V_p V_p^T) M v_i$.
        for $j = 1$ to $i$ do
            $h_{j,i} \leftarrow <v_j, v_{i+1}>$
            $v_{i+1} \leftarrow v_{i+1} - h_{j,i} v_j$
        end for
        $h_{i+1,i} \leftarrow \|v_{i+1}\|_2$
        $v_{i+1} \leftarrow v_{i+1}/h_{i+1,i}$
        Compute the least squares solution of $(\bar{H}_l - \bar{I}_l)y = \beta e_1$
        $RES \leftarrow \beta e_1 - (\bar{H}_l - \bar{I}_l)y$
    until $\|RES\|_2 \le \varepsilon$ or maximum basis size reached
    $\Delta q \leftarrow \Delta q^{[0]} + V_i y$
    $M\Delta q \leftarrow M\Delta q^{[0]} + V_{i+1}\bar{H}_l y + V_p \left[\begin{array}{ccc} w_1 & \cdots & w_i \end{array}\right] y$
end
```

If the algorithm ends because the maximum basis size is reached, we can restart it using the result as the new starting value. At the restart, we know $M\Delta q^{[0]}$. Hence ν GMRES(l) steps require νl or $\nu l + 1$ matrix–vector products with the monodromy matrix, depending on whether $M\Delta q^{[0]}$ is known or not for the first GMRES(l) step. This is precisely the same amount of work as for νl Picard steps. The GMRES(l) method is guaranteed to produce a residual that is not worse than the residual obtained by l Picard steps (measured in the 2-norm). However, this is not necessarily true if we use restarted GMRES.

If we decide to restart the algorithm during the a posteriori analysis, we must restart from the beginning using the previous result as the starting value. Algorithm 3.3 does not store the basis to start again precisely where the algorithm ended. This is less efficient with respect to the number of iterations and matrix–vector products, but is easier to implement since less data has to be stored in between calls of the routine with the same right hand side. Furthermore, it allows us to use the same interface to the routine as for the Picard iterations.

We have not yet implemented this algorithm in our code. However, we believe that it may offer some advantages if ρ is large and more than two or three Picard steps have to be done to obtain the desired result.

3.5.3 Starting values

We need to derive starting values for two situations: at the beginning of the Newton–Picard step and for a restart after the a posteriori analysis.

Starting values for a restart. If the basis V_p has not changed, we can restart algorithm 3.1 or 3.3 using the values of Δq_* and $M\Delta q_*$ returned after the previous run for the same right hand side. The situation is different if the basis has been refined. We need to assure that Δq_* lies in the new subspace $\mathcal{U}^\perp$. Therefore we will reproject Δq_* and at the same time make the corresponding updates to $M\Delta q_*$:

$$\begin{cases} \Delta q_* \leftarrow \left(I - V_p V_p^T\right)\Delta q_*, \\ M\Delta q_* \leftarrow M\Delta q_* - (MV_p)V_p^T \Delta q_*. \end{cases} \tag{3.23}$$

Note that MV_p is known from the subspace iterations. Formula (3.23) produces a good starting value if vectors that were contained in V_p have not changed too much and if the basis size has not decreased (i.e., no new vectors added to V_q). To cope with a decrease of the dimension of $\mathcal{U}$ we suggest to extend Δq_* with components in the direction of the vectors of V_p for the least dominant eigenvalues before the restart of the basis iterations and to remove the unnecessary components afterwards using (3.23). Let $V_p = \left[\begin{array}{cc} V_{p,1} & V_{p,2} \end{array}\right]$ where $V_{p,1}$ is a basis for the maximal invariant subspace of M for all Floquet multipliers larger than or equal to 1 in modulus or some value $\tilde{\rho}$ slightly smaller than 1 to avoid a too ill-conditioned linear system later in the procedure. Let $V_{p,1} \in \mathbb{R}^{p_1}$ and $V_{p,2} \in \mathbb{R}^{p_2}$. If we would use the basis $V_{p,1}$ instead of V_p for the splitting, we would end up with the Q-system

$$\begin{bmatrix} V_{p,2}^T M V_{p,2} - I_{p_2} & V_{p,2}^T M V_q \\ 0 & V_q^T M V_q - I_q \end{bmatrix} \begin{bmatrix} \Delta \bar{q}_{p,*} \\ \Delta \bar{q}_{q,*} \end{bmatrix} = - \begin{bmatrix} V_{p,2}^T r_* \\ V_q^T r_* \end{bmatrix} \tag{3.24}$$

with $\Delta q_* = V_{p,2}\Delta\bar{q}_{p,*} + V_q \Delta\bar{q}_{q,*}$ and r_* equal to r, b_T or b_γ. The lower right block is the Q-system that has to be solved if the basis V_p is used and $\Delta\bar{q}_{p,*}$ are the additional components needed if we would use the smaller P-space spanned by the basis $V_{p,1}$ instead. These components are computed by solving $\Delta\bar{q}_{p,*}$ from

$$\left(V_{p,2}^T M V_{p,2} - I_{p_2}\right)\Delta\bar{q}_{p,*} = -V_{p,2}^T(r_* + MV_q\Delta\bar{q}_{q,*}) \tag{3.25}$$

where $V_q\Delta\bar{q}_{q,*}$ is the solution computed by the Picard iterations or GMRES method using the basis V_p. System (3.25) is small and is easily constructed. $MV_{p,2}$ and $V_{p,2}^T M V_{p,2}$ have already been computed during the basis computation process. $MV_q\Delta\bar{q}_{q,*}$ is recovered from the Picard iterations or GMRES iterations. Before the basis computations, we will add the components $V_{p,2}\Delta\bar{q}_{p,*}$ to Δq_* and $MV_{p,2}\Delta\bar{q}_{p,*}$ to the computed matrix–vector product $M\Delta q_*$ and after the basis iterations unnecessary components are easily removed again using (3.23).

Simple starting value at the beginning of new Newton–Picard step. The Q-projection of the single shooting residual r changes from step to step. At the limit cycle, r and thus Δq_r are 0. Therefore 0 is a reasonable starting value for the Q-system with right hand side Qr.

Although the vector b_T does not change much during the iterations, the right hand side Qb_T also changes considerably since b_T^* is an eigenvector of M^* for the eigenvalue 1 (at least if we neglect the effect of time discretization errors). At the limit cycle, $Qb_T^* = 0$ for a perfect basis and hence $\Delta q_T = 0$. $\Delta q_T^{[0]} = 0$ is a very reasonable choice. Since a nonzero solution of the Q-system at the limit cycle is caused by time discretization errors and errors in the basis computation process and since Qb_T should converge to 0 if an exact basis were used, we expect that Qb_T and thus Δq_T will often change a lot from step to step. Hence it is unreasonable to attempt to use the value of the previous iteration step as a starting value.

Except for changes caused by a change of the basis size, Qb_γ is fairly constant. Here we suggest to use the value of Δq_γ computed in the previous iteration step as a starting value and to reproject Δq_γ into $\mathcal{U}^\perp$ using (3.23). Note that one should only use the value Δq_γ and not the product $M\Delta q_\gamma$ from the previous step since M has changed. Using the old M reduces the reliability of the convergence criterion and the a posteriori analysis. To cope with the effect of a decrease of the basis size, we suggest the same procedure as for the restart after a basis change. At the end of the Newton–Picard step, we add additional components to Δq_γ using the procedure outlined above. The projection (3.23) at the beginning of the next Newton–Picard step again removes unnecessary components.

Constructing better starting values at the beginning of new Newton–Picard step. It is sometimes possible to derive an alternative starting value from components of V_e in the subspace algorithm 2.2 that were rejected from the basis V_p. Let $V_q = \begin{bmatrix} V_{q,1} & V_{q,2} \end{bmatrix}$ where $V_{q,1}$ contains the vectors of V_e that are not used in the basis V_p, ordered according to decreasing modulus of the corresponding eigenvalue. Let $V_{q,1} \in \mathbb{R}^{q_1}$ and $V_{q,2} \in \mathbb{R}^{q_2}$ and $\Delta q_* = V_{q,1}\Delta\bar{q}_{1,*} + V_{q,2}\Delta\bar{q}_{2,*}$. The Q-system can be rewritten as

$$\begin{bmatrix} V_{q,1}^T M V_{q,1} - I_{q_1} & V_{q,1}^T M V_{q,2} \\ V_{q,2}^T M V_{q,1} & V_{q,2}^T M V_{q,2} - I_{q_2} \end{bmatrix} \begin{bmatrix} \Delta\bar{q}_{1,*} \\ \Delta\bar{q}_{2,*} \end{bmatrix} = - \begin{bmatrix} V_{q,1}^T r_* \\ V_{q,2}^T r_* \end{bmatrix}. \tag{3.26}$$

If subspace iteration with projection is used (which we always do) $V_{q,1}^T M V_{q,1} - I_{q,1}$ is an upper block triangular matrix with 1×1 and 2×2-blocks on the diagonal. If the vectors of $V_{q,1}$ would be accurate approximations to the next most dominant Schur vectors of M not contained in the basis V_p, the term $V_{q,2}^T M V_{q,1}$ would be 0. In practise, the more dominant vectors in $V_{q,1}$ are more accurate and the first columns of $V_{q,2}^T M V_{q,1}$ usually have smaller entries than the other columns. Suppose $\Delta\bar{q}_{1,*} \gg \Delta\bar{q}_{2,*}$. This assumption is certainly reasonable for the right hand side r if (3.10) is dominated by the term Qr_r. Furthermore assume $V_{q,2}^T M V_{q,1}$ is not too large. The solution to (3.26) can be approximated by setting $\Delta\bar{q}_{2,*} = 0$ and solving

$$\left[V_{q,1}^T M V_{q,1} - I_{q_1}\right] \Delta\bar{q}_{1,*} = -V_{q,1}^T r_*.$$

Hence we can use $\Delta q_*^{[0]} = V_{q,1}\Delta\bar{q}_{1,*}$ as a starting value. Note that $M\Delta q_*^{[0]} = (MV_{q,1})\Delta\bar{q}_{1,*}$ is also known (since $MV_{q,1}$ can be recovered from the subspace iterations), so we can easily check whether $\Delta q_*^{[0]} = V_{q,1}\Delta\bar{q}_{1,*}$ is a better starting value than $\Delta q_*^{[0]} = 0$ by comparing the norm of

$$\left(I - V_pV_p^T\right)\left(r_* + MV_{q,1}\Delta\bar{q}_{1,*} - V_{q,1}\Delta\bar{q}_{1,*}\right) = \left(I - V_pV_p^T\right)\left(r_* + MV_{q,1}\Delta\bar{q}_{1,*}\right)$$

with the norm of Qr_*. Instead of using all components, we can also try to use only the first k components of $\Delta\bar{q}_{1,*}$, i.e.,

$$\Delta q_*^{[0]} = \sum_{i=1}^{k} V_{q,1}[i]\,\Delta\bar{q}_{1,*}[i]$$

and test which value of k results in the lowest residual. We expect that this procedure can often reduce the error components in the more dominant directions of V_q. This is important if Picard iterations are used, since these error components have the slowest convergence. The procedure is cheap since it requires no new matrix–vector products with the monodromy matrix and since it does not only give a starting value $\Delta q_*^{[0]}$ but also the corresponding $M\Delta q_*^{[0]}$ so we do not need an extra matrix–vector product at the beginning of algorithm 3.1 or 3.3. Further testing of this procedure is necessary.

3.6 Advanced methods for solving the P-system

The P-systems (3.3) or (3.6) are small systems. The cost of solving these systems is only a negligible fraction of the total cost of a Newton–Picard step. Hence we can afford to use expensive but stable methods for their solution. Note that the P-system (3.3) is always singular at a transcritical or pitchfork bifurcation point (and (3.6) even at a fold point) and is very ill-conditioned in the neighbourhood of these points. In chapter 2 we suggested to use Gauss elimination with pivoting to compute a solution of (3.3) or (3.6). A numerically very stable option is to use the least-squares method based on the singular value decomposition instead. We refer to [44] for a thorough discussion of the least-squares method and its relationship to the singular value decomposition; we just mention some of the properties without proof.

Suppose we wish to solve $Ax = b$ where $A \in \mathbb{R}^{m\times n}$ and suppose $\text{Rank}(A) = r$. There exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that

$$A = U\Sigma V^T \tag{3.27}$$

where $\Sigma \in \mathbb{R}^{m\times n}$ has only r nonzero entries: the first r entries on the main diagonal $\sigma_1, \ldots, \sigma_r$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. The columns of V are the *right singular vectors* of A and the columns of U the *left singular vectors*. Note that

$$\begin{aligned} \text{Ker}(A) &= \text{Span}\left(V[r+1:n]\right) \\ \text{Range}(A) &= \text{Span}\left(U[1:r]\right). \end{aligned} \tag{3.28}$$

Let

$$\alpha = \min_{x\in\mathbb{R}^n} \|b - Ax\|$$

and

$$X = \{x \mid \|b - Ax\| = \alpha\}.$$

If $r = \min\{m, n\}$ then X is a singleton. The least squares solution of $Ax = b$ is the unique element x_{LS} of X with minimal 2-norm. Let $\Sigma^+ \in \mathbb{R}^{n\times m}$ be the matrix with all elements equal to zero except for the first r elements on the main diagonal which are $\sigma_1^{-1}, \dots, \sigma_r^{-1}$ (in this particular order).

$$A^+ = V\Sigma^+U^T$$

is known as the pseudo-inverse (or Moore–Penrose inverse) of A. It can be shown that

$$x_{LS} = A^+b.$$

This solution satisfies

$$\begin{array}{l} U[1:r]^T(b - Ax_{LS}) = 0 \\ V[r+1:n]^T x_{LS} = 0. \end{array} \tag{3.29}$$

Implementations of this method usually have a parameter that sets the threshold for the smallest singular value. All singular values below this threshold are set to zero. A careful choice of this parameter is very important, otherwise the detected rank of the matrix may be wrong and this may lead to undesired results. Suppose we approximate Σ^+ with

$$\Sigma^+ = \operatorname{diag}(\sigma_1^{-1}, \cdots, \sigma_s^{-1}, 0, \dots, 0) \in \mathbb{R}^{n\times m}.$$

with $s \le r$. Let $U = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix}$ with $U_1 \in \mathbb{R}^{m\times s}$, $U_2 \in \mathbb{R}^{m\times(r-s)}$ and $U_3 \in \mathbb{R}^{m\times(m-r)}$. The least squares solution $x = V\Sigma^+U^Tb$ then produces the residual

$$r = b - U\Sigma V^T V\Sigma^+U^Tb = U\left(I - \Sigma\Sigma^+\right)U^Tb = U_2U_2^Tb + U_3U_3^Tb.$$

If $b \in \text{Range}(A)$ then $U_3^Tb = 0$. However, we are left with the residual components in $\text{Span}(U_2)$.

If we use the least-squares method, we can omit the explicit parametrizing equation and phase condition. The SVD-based least-squares approach automatically selects good conditions. We will now motivate this.

Suppose $A \in \mathbb{R}^{m\times n}$ with $n > m$ and $\text{Rank}(A) = m$. From (3.29) follows that the least-squares solution of $Ax = b$ also solves the square system

$$\begin{bmatrix} A \\ V[m+1:n]^T \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Moreover, if A has full rank, this system is nonsingular and has a unique solution. Let us first assume we only omit the parametrizing equation. This corresponds to $n = m+1$ and the least-squares solution of $Ax = b$ is precisely the solution of

$$\begin{bmatrix} A \\ v_{m+1}^T \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

This resembles closely the approach used in CONTENT [72] for the continuation of steady-state solutions of $f(x, \gamma) = 0$. Let

$$A = \frac{\partial f(x,\gamma)}{\partial(x,\gamma)}.$$

Furthermore, assume $v^{(0)}$ is given. In CONTENT, $v^{(0)}$ is the predictor direction. At step ν of the Newton process, CONTENT solves the system

$$\begin{bmatrix} A(x^{(\nu-1)},\gamma^{(\nu-1)}) \\ v^{(\nu-1)T} \end{bmatrix} \begin{bmatrix} \Delta x^{(\nu-1)} & \Delta v^{(\nu-1)} \\ \Delta\gamma^{(\nu-1)} & \end{bmatrix} = - \begin{bmatrix} f(x^{(\nu-1)},\gamma^{(\nu-1)}) & Av^{(\nu-1)} \\ 0 & 0 \end{bmatrix} \quad (3.30)$$

and computes

$$\begin{aligned} \begin{bmatrix} x^{(\nu)} \\ \gamma^{(\nu)} \end{bmatrix} &\leftarrow \begin{bmatrix} x^{(\nu-1)} \\ \gamma^{(\nu-1)} \end{bmatrix} + \begin{bmatrix} \Delta x^{(\nu-1)} \\ \Delta\gamma^{(\nu-1)} \end{bmatrix}, \\ w^{(\nu)} &\leftarrow v^{(\nu-1)} + \Delta v^{(\nu-1)}, \\ v^{(\nu)} &\leftarrow \frac{w^{(\nu)}}{\|w^{(\nu)}\|}. \end{aligned}$$

Let $\left\|w^{(\nu)}\right\| = h^{(\nu)}$. We have

$$A^{(\nu-1)}v^{(\nu)} = h^{(\nu)}\left(Av^{(\nu-1)} + A\,\Delta v^{(\nu-1)}\right) = 0.$$

Hence $v^{(\nu)}$ is a vector in $\mathrm{Ker}(A^{(\nu-1)})$. If $A^{(\nu-1)}$ has full rank, then $\dim\left(\mathrm{Ker}(A^{(\nu-1)})\right) = 1$ and $v^{(\nu)}$ is the last vector of V in the SVD (3.27) of $A(x^{(\nu-1)},\gamma^{(\nu-1)})$ (since (3.28)). In iteration step $\nu > 1$, the last right singular vector of $A(x^{(\nu-2)},\gamma^{(\nu-2)})$ is used in the system (3.30), while in the least-squares approach the corresponding vector of $A(x^{(\nu-1)},\gamma^{(\nu-1)})$ is used.

The least-squares method can also replace both the phase condition and the parametrizing equation and will automatically choose quite a good phase condition. Let

$$A = \begin{bmatrix} A_0 & b_0 & b_1 \end{bmatrix}, \quad A_0 \in \mathbb{R}^{m\times m}, \quad A_0 b_0 = 0,$$

and suppose 0 is a simple eigenvalue of A_0. The least-squares solution x_{LS} of $Ax = b$ satisfies

$$V[m+1:m+2]^T x_{LS} = 0.$$

Since $\mathrm{Ker}(A) = \mathrm{Span}\{v_{m+1}, v_{m+2}\}$ and since

$$A\begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix} = 0,$$

we also have

$$\begin{bmatrix} b_0^T & 0 & 0 \end{bmatrix} x_{LS} = 0. \quad (3.31)$$

If A_0 is not exactly singular and if b_0 is close to the smallest eigenvector of A_0, then (3.31) is not exactly satisfied, but still $\begin{bmatrix} b_0^T & 0 & 0 \end{bmatrix} x_{LS} \approx 0$. The P-system (3.3) has precisely this property. Suppose V_p is an exact basis, then at the limit cycle

$$V_p^T(b_T + M\,\Delta q_T) = V_p^T b_T$$

is the eigenvector of $V_p^T M V_p - I_p$ for the trivial Floquet multiplier 1. The SVD based least-squares approach implicitly adds a condition which is almost equivalent to the phase conditions

$$b_T^T V_p \Delta\bar{p} = 0 \quad (3.32)$$

or

$$b_T^T \Delta x(0) = 0 \tag{3.33}$$

(since $b_T^T \Delta x(0) = b_T^T V_p \Delta \bar{p} + b_T^T V_q \Delta \bar{q} \approx b_T^T V_p \Delta \bar{p}$). The least-squares approach automatically selects a close to optimal condition, although conditions like (3.32) or (3.33) with zero right hand side cannot eliminate the phase shift introduced by the predictor in the continuation code. If this is a problem, an explicit phase condition—probably an integral phase condition—should be used.

We have implemented the SVD-based least-squares approach in our code together with the option to omit the parametrizing equation and the phase condition and have computed various branches with this routine. If the parameter to detect the rank is well chosen, this method performs very well and is more robust than our previous routine using Gauss elimination with partial pivoting near pitchfork bifurcation points. In some rare cases there were some problems if the parametrizing equation and phase condition were omitted, particularly in the neighbourhood of the steady-state Hopf bifurcation at the end of a periodic solution branch. We also noticed some problems if a very large stepsize was used. The method did converge, but the distance to the previous solution was much smaller than expected.

3.7 An integral phase condition

It is often said that shooting precludes the use of an integral phase condition such as (1.38). (1.38) is better at avoiding phase shifts and is interesting if time-adaptive meshes are used since it reduces the need for remeshing. Although we have done no experiments with an integral phase condition, we wish to show that it is possible in principle to use an integral phase condition (and an integral pseudo-arclength condition), although an efficient implementation may require more memory in certain cases. We will discuss the use of (1.38) as an example.

Let $\tilde{x}(t)$ be a periodic reference solution with period $\tilde{T}$. (1.38) is easily transformed to

$$g_1(x(0), T, \gamma) = \int_0^1 \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \tilde{T}s} \right)^T \varphi(x(0), Ts, \gamma)\, ds = 0 \tag{3.34}$$

where we omitted the constant factor $\tilde{T}$. Setting $u = Ts$, (3.34) is transformed to

$$g_2(x(0), T, \gamma) = \int_0^T \frac{1}{T} \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \frac{\tilde{T}}{T}u} \right)^T \varphi(x(0), u, \gamma)\, du = 0. \tag{3.35}$$

g_1 and g_2 are discretized with a quadrature formula that uses the time meshpoints of the numerical integrator for φ. The efficient evaluation of g_1 or g_2 requires storing the orbits $\tilde{x}(t)$ and $\varphi(x(0), t, \gamma)$.

To compute the derivatives, formula (3.34) is easier to use than the equivalent formula (3.35). Let us first consider the derivative with respect to the initial state of g_1. We get

$$c_s^T = \frac{\partial g_1}{\partial x(0)} = \int_0^1 \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \tilde{T}s} \right)^T \left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial x(0)} \right|_{(x(0), Ts, \gamma)} ds. \tag{3.36}$$

(3.36) is expensive to compute, but in (3.3), we only need selected scalar products of c_s with the columns of V_p and the various vectors Δq_*. These can be computed from

$$c_s^T v = \frac{\partial g_1}{\partial x(0)} v = \int_0^1 \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \tilde{T}s} \right)^T \left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial x(0)} \right|_{(x(0), Ts, \gamma)} v \, ds. \tag{3.37}$$

Note that

$$\left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial x(0)} \right|_{(x(0), Ts, \gamma)} v = v(Ts)$$

where $v(t)$ is given by the variational equations (2.43). It is also possible to compute this term via finite differences. For instance, for the first-order formula (2.47), one should compute $\varphi(x(0) + hv, t, \gamma)$ and set

$$\left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial x(0)} \right|_{(x(0), Ts, \gamma)} v \approx \frac{\varphi(x(0) + hv, Ts, \gamma) - \varphi(x(0), Ts, \gamma)}{h}.$$

To compute (3.37) efficiently, one needs to store the functions $v(t)$ or $\varphi(x(0) + hv, t, \gamma)$. The value at time T of these functions is also needed for the corresponding matrix–vector products with M. Otherwise one should combine the computation of (3.37) with the computation of the corresponding matrix–vector products.

It is possible to derive formulas for the other derivatives in a similar way. The derivative with respect to T is given by

$$d_{s,T} = \frac{\partial g_1}{\partial T} = \int_0^1 \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \tilde{T}s} \right)^T f\left(\varphi(x(0), Ts, \gamma), \gamma\right) s \, ds \tag{3.38}$$

and the derivative with respect to the parameter γ is

$$d_{s,\gamma} = \frac{\partial g_1}{\partial \gamma} = \int_0^1 \left(\left. \frac{d\tilde{x}(t)}{dt} \right|_{t = \tilde{T}s} \right)^T \left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial \gamma} \right|_{(x(0), Ts, \gamma)} ds. \tag{3.39}$$

The term

$$\left. \frac{\partial \varphi(x(0), t, \gamma)}{\partial \gamma} \right|_{(x(0), Ts, \gamma)}$$

is computed using the variational equation (2.49) or finite differences. The computation of the scalar term $d_{s,\gamma}$ can be implemented in a similar way as the scalar products $c_s^T v$: either we store $w(t)$ in (2.49) or $\varphi(x(0), t, \gamma + \Delta\gamma)$ or we combine the evaluation of (3.39) with the computation of b_γ.

The approach can be extended to integral parametrizing equations such as (1.54) or (1.55). It is clear that integral conditions for shooting are complicated. However, they may make sense in conjunction with time-adaptive meshes or space-time methods that store the orbit to allow for a more efficient parallelization (e.g., the waveform relaxation method, see, e.g., [114]).

	branch I	branch II	branch III	branch IV	total
NPGS	996	1102	831	1304	4233
CNP	1194	1132	1051	1361	4738

Table 3.1: Number of IVP solves for the *NPGS* and *CNP* methods, 1-dimensional Brusselator model.

3.8 Testcases

1-dimensional Brusselator

We consider again the Brusselator model (2.55-2.56) studied in section 2.5. To show the efficiency of the *NPGS* and *CNP* method derived in this chapter, we computed the same 4 branches as in section 2.5.3 with both methods. In Table 2.3, we reported the number of initial value problem (IVP) solves for the methods of the previous chapter. We counted each matrix–vector product for one IVP solve. Each of the results reported was the best out of 12 runs with different parameters. We heavily optimized the parameters. Furthermore, the method often failed and this is not reported in the tables since we only reported the best results.

We changed the implementation such that only the accurate basis vectors were used. The P-system was solved using the least-squares approach. We omitted the phase condition and the pseudo-arclength equation. We did not implement the a priori/a posteriori convergence analysis and used a fixed number of Picard steps to solve the Q-systems with the right-hand sides Qr and Qb_γ. We set $\Delta q_T = 0$. We did a number of runs using the *CNP* method to compute branch I and used the best settings to compute the branches II, III, and IV and to compute the 4 branches using the *NPGS* method. Hence the parameters were not as much optimized as before. For the basis iterations, we used 2 extra vectors. The threshold to add vectors to the basis was 0.55, the threshold to delete vectors 0.45. We switched to a Newton–Picard step as soon as the Schur basis corresponding to all eigenvalues larger than 0.8 satisfied $\sigma_{max}(Z_k) \leq 10^{-2}$ with Z_k given by (2.53). To solve the Q-system, we used 2 Picard steps with a zero starting value. The iterations were stopped once the residual and the updates Δp, ΔT and $\Delta\gamma$ were smaller than 10^{-6}. After the iterations, we computed all Floquet multipliers larger than 0.7 until the corresponding basis satisfied $\sigma_{max}(Z_k) \leq 10^{-4}$. We also required that the changes between corresponding eigenvalues in 2 successive subspace iteration steps were smaller than 10^{-4}. The results are reported in table 3.1. For the corresponding branches, we needed 3076 IVP solves using the $NPGS(l)$ method and 3670 IVP solves using the $CNP(l)$ method of the previous chapter. Although we needed 30% to 40% more IVP solves, we think that the results are acceptable since we did not optimize the parameters as much as for the tests in section 2.5.3 and since the new methods are much more robust. In fact, using the methods of chapter 2, the continuation procedure fails much more often. Also, the methods from chapter 3 are more sensitive to the parameter settings than the methods introduced in this chapter.

We did observe some failures during our tests with the *NPGS* and *CNP* method. Most of the failures were caused by the omission of the phase condition. The least-squares based solution technique cannot remove a phase shift introduced by the predictor while phase conditions using the previous orbit as a reference solution can do that. In some cases, the phase shift between 2 successive orbits grew as the continuation proceeded and the predictor generated bad starting values. The continuation method failed because the corrector needed too many iterations and the continuation stepsize had to be decreased too much or because the corrector produced a new solution at the wrong side of the previous one and returned along the already computed part of the branch. We conclude that it is not a good idea to omit the phase condition. Instead one should use a phase condition like (2.63), preferably using the previous periodic solution instead of the result of the predictor as in (2.63).

Olmstead model

The bifurcation diagram (Figure 2.4) of the Olmstead model (2.61-2.62) was computed using the CNP method also used for the 1-dimensional Brusselator. The $CNP(l)$ method presented in the previous chapter hardly managed to pass the fold point on branch 1 and the other methods had even more trouble. This already demonstrates the increased robustness. Towards the end of both computed branches, the unstable Floquet multipliers start to grow rapidly while the other ones (except the trivial 1) go to zero quickly. We expect that both branches evolve to a homoclinic orbit.

We made an implementation of the model using the trapezoidal rule for time integration and finite differences for the matrix–vector products and one that uses the variational equations instead. Using finite differences, we are able to compute branch 1 until $T \approx 35.88$. The largest computed Floquet multiplier at that point is 26.63. The computed Floquet multiplier closest to 1 is 0.872. This indicates that the time integration and matrix–vector products are not very accurate (otherwise we should have a Floquet multiplier very close to 1). Using the variational equations instead, we still obtain three correct digits for the trivial 1 Floquet multiplier and we can compute the branch until $T \approx 36.54$ and $\mu_1 = 40.88$. Even then, the error on the computed approximation to the trivial Floquet multiplier is only around 0.001. On branch 2, we can to compute a solution with $T = 22.69$ and $\mu_1 = 153.5$. However, we have to lower our requirements for the norm of the final residual to 10^{-4} and the computations proceed slowly using a very small stepsize in the continuation code. The inaccurate results of the finite difference based method and the failure to decrease the P-projection of the residual below 10^{-4} indicate that in this model, the time integration code causes trouble if the orbit becomes very unstable or higher order terms are large. It is very likely that both occur since the method converges until the size of the residual is 10^{-4}, but does so very slowly. It is impossible to distinguish from our convergence analysis between higher order terms and numerical errors. Note that the Q-projection of the residual converges well with both implementations. The problems are limited to the unstable subspace. The Q-projection of the residual is several orders of magnitude smaller than the P-projection and its norm easily decreases to 10^{-7}-10^{-8}.

2-dimensional Brusselator model

We also computed a part of a branch for the 2-dimensional Brusselator model (2.57). We used $D_X = 0.008$ and $D_Y = 0.004$ as in [99]. We set $A = 2$ and $B = 5.45$. At these parameter values, the trivial steady-state solution $X \equiv A$ and $Y \equiv B/A$ has a first Hopf bifurcation at $L \approx 0.72$. We computed part of the stable branch of periodic solutions that appears at this point, from $L = 0.78$ to $L \approx 3$. We used a second order spatial finite difference discretization with 10×10 and 20×20 discretization points, resulting in ODE systems of dimension 200 and 800 respectively. The time integration was done using the trapezoidal rule. However, the linear system solver did not fully exploit the structure of the linear systems in this time integration code and this made it impossible to compute solutions on a finer grid because of excessive computation time.

Using the 20×20 grid, the continuation code computes 17 orbits on the branch. Initially, at low values of L, everything goes fine and the code works efficiently, requiring around 200 IVP solves per orbit. For higher values of L however, a cluster of eigenvalues develops around -0.75. In the last computed point on the 20×20-grid, at $L = 3.12$, there are over 40 computed Floquet multipliers close to -0.75. The basis used during the subspace iterations grows from 12 vectors in the start point to 60 vectors in the last point. The number of matrix–vector products grows excessively towards the end of the computed branch. The accurate computation of the Floquet multipliers larger than 0.75 in modulus gets very expensive. In total, we needed 4295 time integrations (an average of 253) to compute the orbits. However, 9848 matrix–vector products had to be computed to compute the dominant Floquet multipliers with the desired accuracy. Still, the computation of the orbits themselves was quite efficient and the code did not fail in the presence of a large cluster of eigenvalues and managed to increase its basis correctly as the computations proceeded.

We believe that the results which we obtained are not a correct representation of the infinite-dimensional system. We observed that for the values of D_X and D_Y used, the solution develops strong spatial gradients near the boundaries during part of the period and we can not capture the gradients well enough with our rather coarse discretization. The gradients get stronger as L gets larger. We suspect that the inaccurate representation of the physical model causes the cluster of eigenvalues and the cluster might disappear on a finer grid. To test our implementation, we made an independent implementation in Matlab and did some time integrations using one of the built-in methods from Matlab. At the given parameter values, we obtained similar results as with our Newton–Picard code. Hence we assume that our implementation of the system is correct. Using $D_X = 0.008$ and $D_Y = 0.004$, the gradients are less pronounced. We were not able to do tests on a finer grid to verify the solution because our time integration code is too inefficient for this 2-dimensional problem.

Model of Elezgaray and Arneodo

We also computed a branch of a reaction-diffusion model studied by Elezgaray and Arneodo in [35]:

$$\begin{cases} \dfrac{\partial u}{\partial t} = D\dfrac{\partial^2 u}{\partial^2 x} + \dfrac{1}{\epsilon}(v - (u^2 + u^3)), \\[2ex] \dfrac{\partial v}{\partial t} = D\dfrac{\partial^2 v}{\partial^2 x} + \alpha - u, \end{cases} \tag{3.40}$$

with Dirichlet boundary conditions

$$\begin{cases} u(0,t) \equiv u(1,t) \equiv -2, \\ v(0,t) \equiv v(1,t) \equiv -4. \end{cases} \tag{3.41}$$

D is used as the bifurcation parameter. ϵ and α are both fixed at 0.1. The bifurcation diagram can be found in [46], with a different scaling of the equations however. (3.40-3.41) has a branch of periodic solutions that emanates from a steady-state Hopf bifurcation at $D \approx 0.02630$ and disappears in another steady-state Hopf bifurcation around $D \approx 0.03230$. On this branch, there are period doubling bifurcations around $D \approx 0.03208$ and $D \approx 0.03227$. In between the 2 period doubling bifurcations, the branch is unstable and there are various chaotic regimes limited by period doubling cascades at the 2 period doubling points. We used second order finite differences for the space discretization, the trapezoidal rule for time integration and variational equations for the matrix–vector products. The solution develops strong spatial gradients and a very fine space discretization is needed to compute the branch accurately. We did computations using 63, 255 and 1023 discretization points and managed to compute the complete branch, also the unstable part in the chaotic region. In this region, one of the Floquet multipliers grows to values around -190. A graph of the modulus of the computed dominant Floquet multipliers on part of the branch using 1023 discretization points is given in Figure 3.7. We requested the computation of all Floquet multipliers larger than 0.75 in modulus. There is little difference between the Floquet multipliers for 1023 and for 255 discretization points. However, the bifurcation points have shifted a little compared to the results for 63 discretization points. 87 orbits were computed to construct Figure 3.7 There were 31 failures at which the stepsize was decreased. 13254 time integrations were required (4083 for failed points and 9171 for successful points), or an average of 112.3 time integrations per successfully computed continuation point. We never needed more than 213 time integrations for a single orbit. Note that we used conservative settings and did not try to optimize the result. We managed to compute highly unstable solutions in a chaotic region without too much difficulties and with a very good efficiency.

3.9 An application to delay equations

Together with T. Luzyanina (Institute of Mathematical Problems in Biology, Pushchino, Moscow Region, Russia) and K. Engelborghs (K.U.Leuven), we studied the application of the Newton–Picard idea to the computation of periodic solutions of delay differential equations with one or more finite discrete delays. We adapted our single shooting based code to compute branches of periodic solutions of delay equations. Problems met during the tests triggered the development and implementation of some of the improvements proposed in this chapter. Although our contribution to this work was limited to making some changes in the code, assisting in the interpretation of the convergence behaviour of the Newton–Picard solver and advice on the use of the code, we wish to mention the basic ideas and present some results in this text to illustrate that the ideas behind our method are also applicable to other infinite dimensional problems.

We first give a brief introduction on periodic solutions of delay equations. Then we will show how the shooting procedure can be adapted to compute periodic solutions of

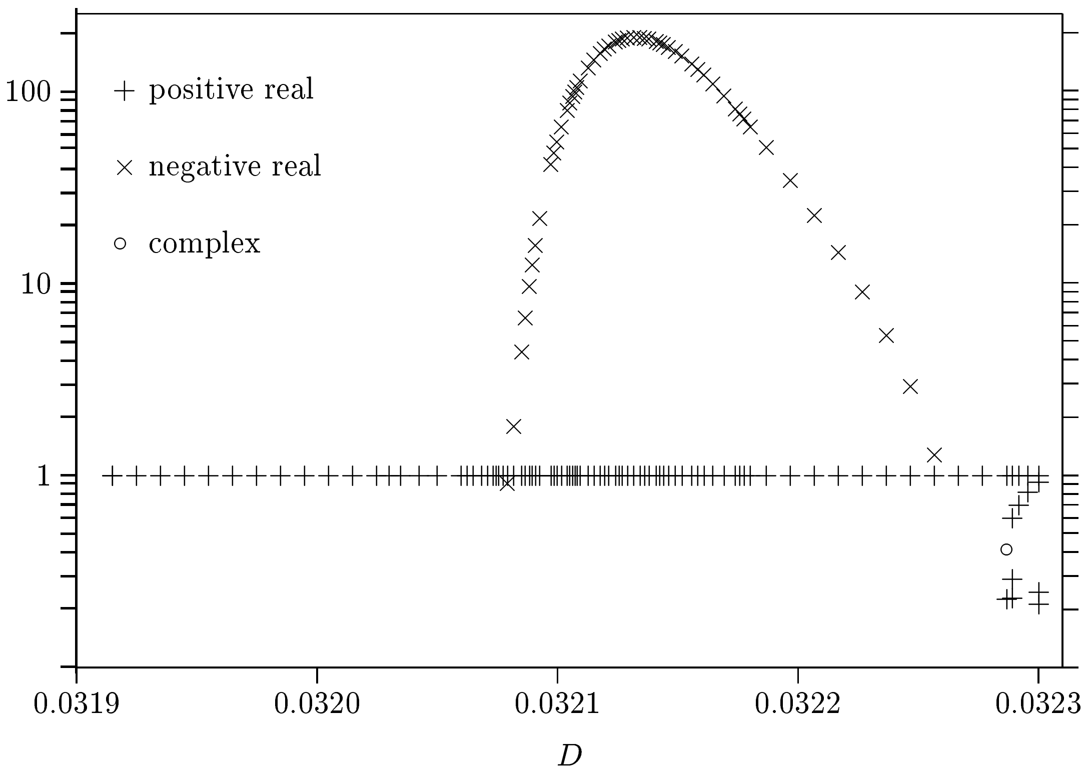


Figure 3.7: Modulus of the computed Floquet multipliers for the model of Elezgaray and Arneodo in terms of the parameter D, 1023 discretization points. Threshold in the code: 0.75. The plusses stand for positive real Floquet multipliers, the stars for negative real floquet multipliers, and the circles stand for pairs of complex conjugate Floquet multipliers.

DDEs and how the Newton–Picard idea can be applied. We will end this section with some test results. For a more complete discussion, the reader is referred to [76].

3.9.1 Periodic solutions of DDEs

We study the system

$$\frac{dx}{dt} = f(x(t), x(t-\tau_1), \ldots, x(t-\tau_k), \gamma) \tag{3.42}$$

of DDEs with k discrete, finite and strictly positive delays $\tau_1, \ldots, \tau_k$. We wish to study branches of periodic solutions of (3.42) were the parameter γ is allowed to vary.

$$f : \mathbb{R}^n \times \mathbb{R}^n \times \cdots \times \mathbb{R}^n \times \mathbb{R} = \mathbb{R}^{(k+1)\times n+1} \mapsto \mathbb{R}^n$$

is a vector-valued function. Note that we do not assume that n is large. In fact, n can be as small as 1. Let

$$\tau = \max_i \tau_i$$

be the largest delay. To uniquely identify a trajectory of (3.42), an initial condition on the interval $[-\tau, 0]$ must be specified. A single point $x(0)$ as in the ODE case is not sufficient. Since an initial function is needed to uniquely determine a trajectory, DDE problems are infinite-dimensional. We will denote the initial condition by ϕ, i.e.,

$$\phi : [-\tau, 0] \mapsto \mathbb{R}^n \tag{3.43}$$

is a continuous vector-valued function defined on the interval $[-\tau, 0]$. Let $\varphi(\phi, t, \gamma)$ denote the solution at time t of (3.42) with initial condition (3.43). Furthermore, let

$$x_T(\phi, \gamma) : [-\tau, 0] \mapsto \mathbb{R}^n : x_T(\phi, \gamma)(\theta) = \varphi(\phi, T + \theta, \gamma)$$

denote a segment of length τ of the solution of (3.42) shifted from time T to 0. $x_T(\phi, \gamma)$ is a function on an interval of length τ, parametrized by the initial condition ϕ and the scalar system parameter γ.

A non-constant solution $x(t) = \varphi(\phi, t, \gamma)$ of (3.42) is a periodic solution if there exists a strictly positive number T such that

$$x_T(\phi, \gamma) - \phi = 0. \tag{3.44}$$

The smallest strictly positive number T satisfying (3.44) is the period of the periodic solution. Note that (3.44) expresses an equality of functions on the interval $[-\tau, 0]$ and not of vectors as in the ODE case! The *linearized Poincaré operator* M $(M = \partial x_T(\phi, \gamma)/\partial \phi)$ is defined by the equality

$$M\psi = y_T(\psi), \tag{3.45}$$

where $y_T(\psi)$ is the solution on $[T - \tau, T]$ of the variational equation

$$\frac{dy}{dt} = \sum_{i=0}^{k} \left. \frac{\partial f(x_0, \ldots, x_k, \gamma)}{\partial x_i} \right|_{(\varphi(\phi, t-\tau_0, \gamma), \ldots, \varphi(\phi, t-\tau_k, \gamma), \gamma)} y(t - \tau_i) \tag{3.46}$$

$(\tau_0 := 0)$ for the initial condition ψ. The variational equation is also a system of delay equations with the same delays as the original system (3.42). The eigenvalues of the operator M are the Floquet multipliers of the periodic solution $\varphi(\phi, t, \gamma)$. For DDEs with finite delays, the linearized Poincaré operator is compact (see [50]). The Floquet multipliers have finite multiplicity and zero is their only cluster point. Note that $\mu = 1$ is always a Floquet multiplier, just as in the ODE case.

3.9.2 Single shooting for periodic solutions of DDEs

Until now, most authors used a Fourier series approximation to compute periodic solutions of DDEs [15, 32, 109]. In [49], a shooting approach is used. The Fourier approach is quite efficient, but it is hard to obtain stability information. The shooting approach constructs a discrete version of the linearized Poincaré operator so it is easy to recover the stability information.

The single shooting method solves the system

$$\left\{ \begin{array}{rcl} r(\phi, T) & = & x_T(\phi) - \phi = 0, \\ s(\phi, T) & = & 0 \end{array} \right. \tag{3.47}$$

for the initial function ϕ and the period T. In (3.47), $s(\phi, T) = 0$ is a phase condition to fix the initial function along the limit cycle. We can solve (3.47) using Newton's method, requiring the repetitive solution of (infinite-dimensional) linear systems

$$\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} \\ c_s^{T^{(\nu)}} & d_{s,T}^{(\nu)} \end{bmatrix} \begin{bmatrix} \Delta\phi^{(\nu)} \\ \Delta T^{(\nu)} \end{bmatrix} = - \begin{bmatrix} r(\phi^{(\nu)}, T^{(\nu)}) \\ s(\phi^{(\nu)}, T^{(\nu)}) \end{bmatrix} \tag{3.48}$$

with

$$\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} \\ c_s^{T^{(\nu)}} & d_{s,T}^{(\nu)} \end{bmatrix} = \left. \frac{\partial(r,s)}{\partial(\phi,T)} \right|_{(\phi^{(\nu)}, T^{(\nu)})} .$$

We will again omit the superscript ν from our notations. At the periodic solution, the operator M^* has an eigenvalue 1 and the corresponding eigenvector is b_T^*, i.e.,

$$M^* b_T^* = b_T^*. \tag{3.49}$$

To compute the solution numerically, we need to discretize the initial function ϕ. We choose a time mesh of m points and replace the function ϕ with a $N = nm$ dimensional vector representing the n-dimensional function ϕ in the m meshpoints. Note that we will not change our notation; from now on, ϕ and $x_T(\phi)$ will denote vectors. $x_T(\phi)$ represents the solution on the interval $[T - \tau, T]$ using the same time mesh as ϕ but shifted to $[T - \tau, T]$ and can be computed using a special time integration code for delay equations. This code should return the N-dimensional vector $x_T(\phi)$ and not just the n-dimensional vector at time T. M is now a $N \times N$-matrix . Its dominant eigenvalues are good approximations to those of the linearized Poincaré operator and do not change much if the time mesh is refined. The smaller eigenvalues of M may be less accurate approximations to the corresponding eigenvalues of the linearized Poincaré operator, but the approximation will improve if the time mesh is refined. The new eigenvalues that appear if the time mesh is refined will be very close to zero. The spectral picture is very similar to Figure 1.2. M can be computed column by column using a discrete version of (3.46) or using finite differences. Note that we do not have to integrate an N-dimensional system, but an n-dimensional system where n is usually small. If n is small, it is not too expensive to store the trajectory and information on the various derivatives of f in (3.46) and the variational equations can be integrated efficiently. Although n is small, $N = nm$ is large and we wish to avoid computing, storing and factoring the full $N \times N$-matrix M.

Note that because of the time discretization of the initial condition, 1 is not an eigenvalue of M^* anymore and b_T^* is not an eigenvector. However, M^* will have an eigenvalue very close to 1 with a corresponding eigenvector close to b_T^*. (3.48) in terms of matrices and vectors instead of operators and functions is equivalent to (2.3) and the Newton–Picard approach is easily generalized.

3.9.3 The Newton–Picard method for DDEs

We will now briefly discuss the Newton–Picard procedure for DDEs and stress the main differences with the ODE case. We will use the algebraic framework proposed in this chapter and derive the method for pseudo-arclength continuation.

The initial state vector ϕ, period T and parameter γ are solved from

$$\left\{\begin{array}{lcl} r(\phi,T,\gamma) & = & x_T(\phi,\gamma)-\phi=0, \\ s(\phi,T,\gamma) & = & 0, \\ n(\phi,T,\gamma;\eta) & = & 0. \end{array}\right. \tag{3.50}$$

Newton's method leads to the linearized system

$$\left[\begin{array}{ccc} M-I & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \\ c_n^T & d_{n,T} & d_{n,\gamma} \end{array}\right]\left[\begin{array}{c} \Delta\phi \\ \Delta T \\ \Delta\gamma \end{array}\right] = -\left[\begin{array}{c} r \\ s \\ n \end{array}\right]. \tag{3.51}$$

We have to adapt assumption 2.4 to take into account that 1 is not an exact eigenvalue of M^*.

Assumption 3.1 *Let $y^*=(\phi^*,T^*,\gamma^*)$ denote an isolated solution to (3.50), and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M(y)=\frac{\partial x_T(\phi,\gamma)}{\partial\phi}(y)$ for $y\in\mathcal{B}$ and denote its eigenvalues by μ_i, $i=1,\ldots,N$. Assume that for all $y\in\mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho=\{|z|\le\rho\},\quad 0<\rho<1$$

and that no eigenvalue has modulus ρ; i.e., for all $y\in\mathcal{B}$

$$|\mu_1|\ge|\mu_2|\ge\cdots\ge|\mu_p|>\rho>|\mu_{p+1}|\ge\ldots\ge|\mu_N|.$$

Furthermore, ρ should be chosen small enough such that the eigenvalue of $M^*=M(y^*)$ corresponding to the trivial Floquet multiplier 1 of the linearized Poincaré operator (3.45) at the limit cycle is larger than ρ in modulus.

The further derivation of the method is straightforward. We briefly recall the important steps. The bases V_p and V_q are chosen as before. Let

$$\phi=V_p\bar\phi_p+V_q\bar\phi_q=\phi_p+\phi_q,\ \phi_p=V_p\bar\phi_p,\ \phi_q=V_q\bar\phi_q \tag{3.52}$$

with $\bar\phi_p\in\mathbb{R}^p$ and $\bar\phi_q\in\mathbb{R}^q=\mathbb{R}^{N-p}$. We insert (3.52) in (3.51) and also multiply the first N rows of (3.51) at the left hand side with $\left[\begin{array}{cc} V_q & V_p\end{array}\right]^T$ and obtain

$$\left[\begin{array}{cccc} V_q^TMV_q-I_q & 0 & V_q^Tb_T & V_q^Tb_\gamma \\ V_p^TMV_q & V_p^TMV_p-I_p & V_p^Tb_T & V_p^Tb_\gamma \\ c_s^TV_q & c_s^TV_p & d_{s,T} & d_{s,\gamma} \\ c_n^TV_q & c_n^TV_p & d_{n,T} & d_{n,\gamma} \end{array}\right]\left[\begin{array}{c} \Delta\bar\phi_q \\ \Delta\bar\phi_p \\ \Delta T \\ \Delta\gamma \end{array}\right] = -\left[\begin{array}{c} V_q^Tr \\ V_p^Tr \\ s \\ n \end{array}\right].$$

We assumed a perfect basis, so $V_q^TMV_p=0$. This system can be solved by first computing $\Delta\bar\phi_{q,r}$, $\Delta\bar\phi_{q,T}$ and $\Delta\bar\phi_{q,\gamma}$ from

$$[V_q^TMV_q-I_q]\left[\begin{array}{ccc} \Delta\bar\phi_{q,r} & \Delta\bar\phi_{q,T} & \Delta\bar\phi_{q\gamma r} \end{array}\right] = -\left[\begin{array}{ccc} V_q^Tr & V_q^Tb_T & V_q^Tb_\gamma \end{array}\right], \tag{3.53}$$

then solving $\Delta\bar{\phi}_p$, ΔT and $\Delta\gamma$ from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T\left(b_T + M\,\Delta\phi_{q,T}\right) & V_p^T\left(b_\gamma + M\,\Delta\phi_{q,\gamma}\right) \\ c_s^T V_p & d_{s,T} + c_s^T\Delta\phi_{q,T} & d_{s,\gamma} + c_s^T\Delta\phi_{q,\gamma} \\ c_n^T V_p & d_{n,T} + c_n^T\Delta\phi_{q,T} & d_{n,\gamma} + c_n^T\Delta\phi_{q,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{\phi}_p \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T\left(r + M\,\Delta\phi_{q,r}\right) \\ s + c_s^T\Delta\phi_{q,r} \\ n + c_n^T\Delta\phi_{q,r} \end{bmatrix} \tag{3.54}$$

(with $\Delta\phi_{q,r} = V_q\Delta\bar{\phi}_{q,r}$, etc.) and computing

$$\Delta\phi_q = \Delta\phi_{q,r} + \Delta T\,\Delta\phi_{q,T} + \Delta\gamma\,\Delta\phi_{q,\gamma}.$$

Finally, we update

$$\phi \leftarrow \phi + \Delta\phi_q + V_p\Delta\bar{\phi}_p;\; T \leftarrow T + \Delta T;\; \gamma \leftarrow \gamma + \Delta\gamma.$$

The subsystems (3.53) and (3.54) are solved as before and the convergence analysis of section 3.2.2 can still be used. Note that the computation of the vector b_T is a little different from the ODE case. b_T is evaluated by computing f along the trajectory at the time mesh points of ϕ translated to $[T-\tau, T]$. Matrix–vector products Mv are computed using a discrete version of (3.46) or finite differences. It is also possible to derive a variational equation for the computation of b_γ. As an alternative, finite differences can be used.

3.9.4 Testcases

We will present results for two test problems. The solutions were computed using the continuation variant of the Newton–Picard method. $\Delta\phi_{q,r}$ and $\Delta\phi_{q,\gamma}$ are solved from (3.53) using a fixed number of Picard iterations. We set $\Delta\phi_{q,T} = 0$. The subspace $\mathcal{U}$ was constructed using only sufficiently converged eigenvectors. The P-system (3.54) was solved using a least-squares routine based on the singular value decomposition as described in section 3.6. We used an explicit phase condition but omitted the parametrizing equation. The time integration was done using a fourth-order Runge–Kutta method with adaptive step size and a fourth-order interpolation scheme. The matrix–vector products were computed using first order finite differences. The initial condition was discretized on a uniform grid.

Testcase 1

To validate the method, we used an equation that has been studied analytically (see, e.g., [62, 89, 116]):

$$\frac{dx}{dt} = -\alpha\,x(t-1)\,\frac{1 + x^2(t-1)}{1 + x^4(t-1)}. \tag{3.55}$$

It was shown analytically that a branch of periodic solutions bifurcates from the zero solution at $\alpha = \pi/2$, that solutions of period $T = 4$ exist for $\alpha > \pi/2$ and that a fold bifurcation occurs on the branch of periodic solutions at a value of α “close to” $\pi/2$.

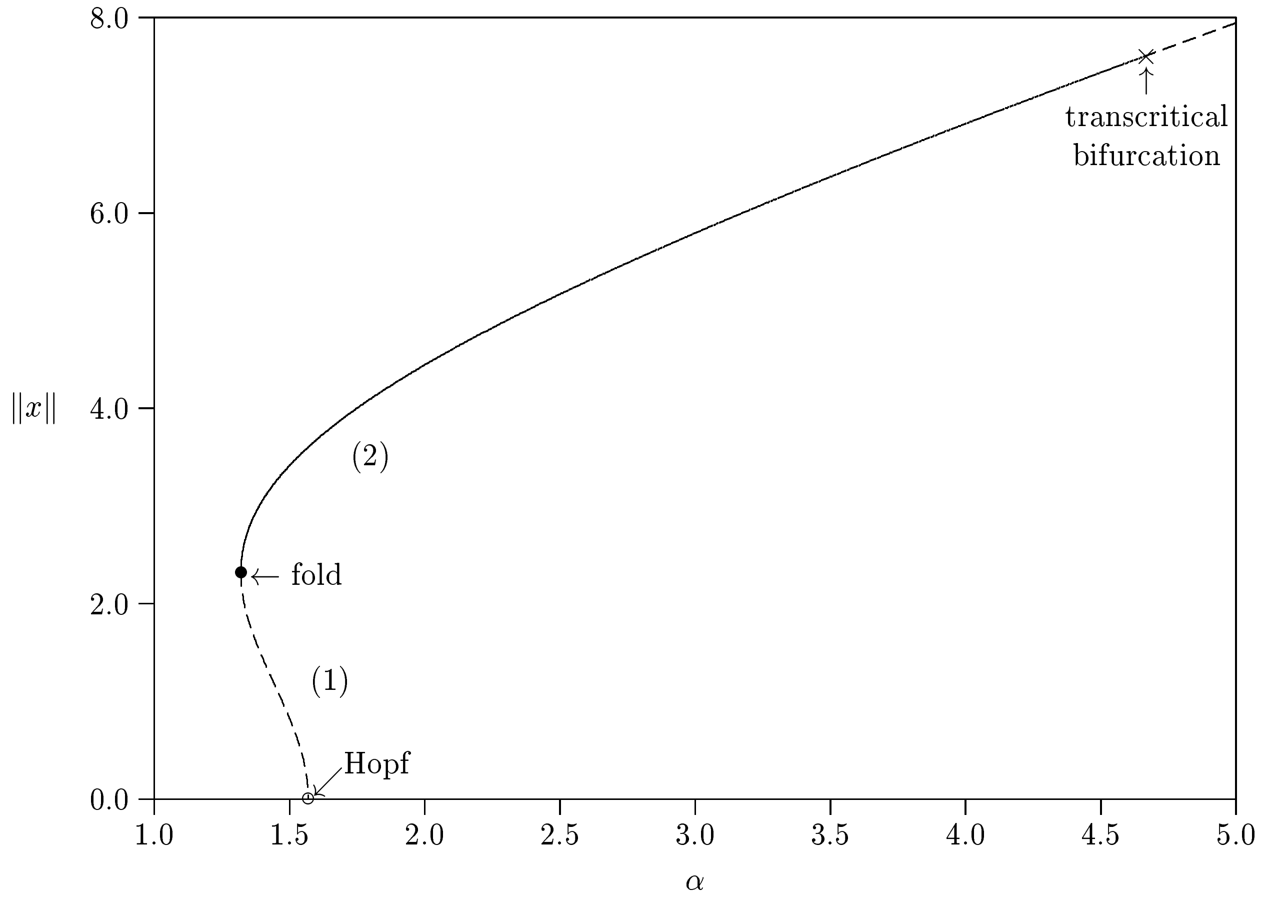


Figure 3.8: Branch of periodic solutions of the testcase 1 model originating at the Hopf bifurcation for $\alpha = \pi/2$.

We computed the branch of periodic solutions that bifurcates at the Hopf point $\alpha = \pi/2$. The bifurcation diagram is given in Figure 3.8. The computations were done using $N = 21$ discretization points on the delay interval and using $\rho = 0.3$ and two extra vectors. The evolution of the Floquet multipliers on this branch is shown in Figure 3.9. Note that the Floquet multipliers are extremely clustered around 0: nowhere along the branch are more than two Floquet multipliers larger than 0.3. The Floquet multipliers $\mu_4, \ldots, \mu_{21}$ all remain smaller than 0.0044 along the branch. Three bifurcation points were detected. The branch starts at a Hopf bifurcation of the trivial steady state at $\alpha \approx 1.57 \approx \pi/2$ and becomes stable at a fold point at $\alpha \approx 1.32$. The stability is lost in the subsequent transcritical bifurcation at $\alpha \approx 4.67$. We did not compute the branch intersecting at this point.

To check the accuracy of the obtained results, we computed a point on the branch for different values of N ($N = 11$, 21, 41 and 81) and checked the convergence of the trivial Floquet multiplier to 1 and of the period to 4. These results are shown in table 3.2. The dominant Floquet multiplier and the period converge quickly to the correct values. There is little difference between the results for $N = 41$ and $N = 81$. As one would expect, the less dominant eigenvalues converge slower. Table 3.3 shows the number of initial value problems (IVP) solves per point for 12 points computed in the segment $3.37 \leq \alpha \leq 4.38$ We used a fixed stepsize continuation code with fixed steps in the parameter direction

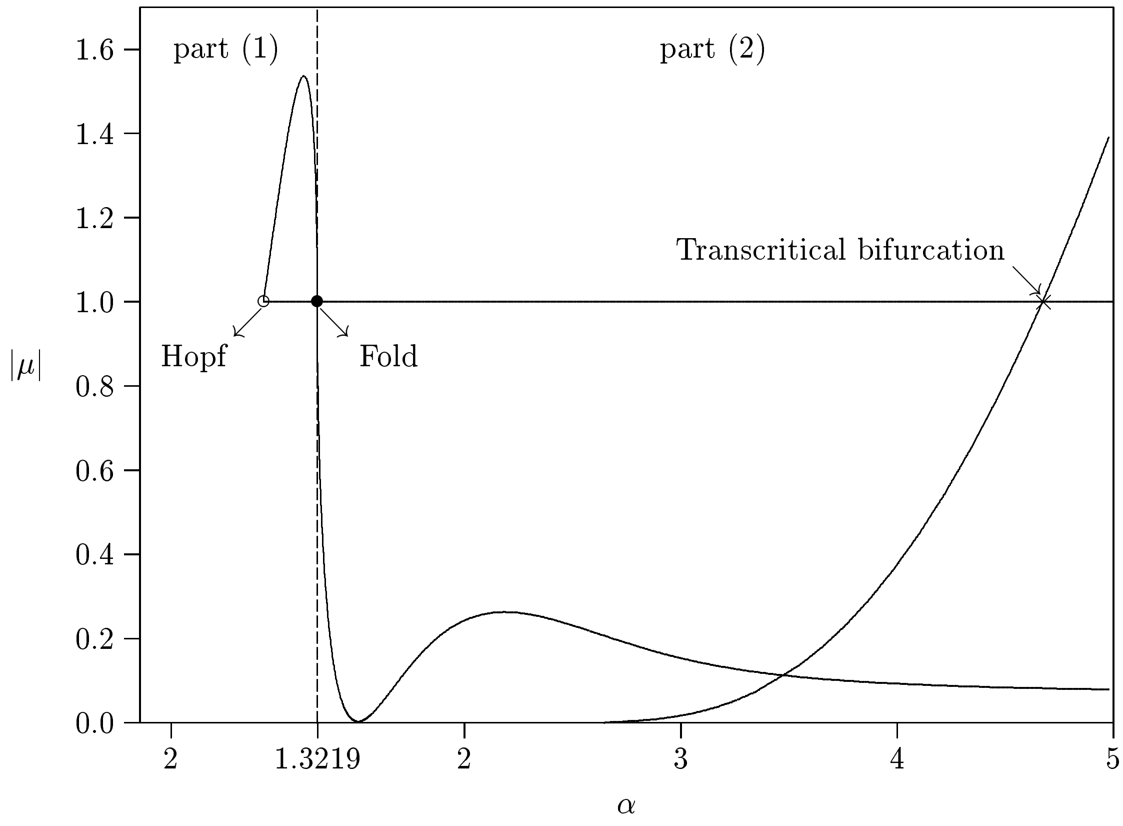


Figure 3.9: Testcase 1: the dominant Floquet multipliers along the branch in the previous figure. Notice that the parameter initially decreases and then again grows along the horizontal axis.

to generate solutions at identical values of the parameter for each of the values of N. Each matrix–vector product counts for one IVP solve. There is little difference between the cases $N = 21$, $N = 41$ and $N = 81$, which is what we expect from the theory since the expected number of iteration steps and matrix–vector products during each step is determined by the dynamics of the system and not the discretization. For $N = 11$ we needed more IVP solves. At this value of N, the discrete approximation of the continuous model is not so good. Note that for $N = 81$, the full Newton method requires 81 IVP solves to construct the Jacobian matrix in the first step. Another 81 IVP solves are needed to compute the Floquet multipliers from the monodromy matrix at the end of the iterations using the QR-algorithm. Furthermore, one often needs to recompute the Jacobian matrix at every Newton step in the beginning of the iterations. Although our testcase is a rather small problem, we can conclude that our approach is already faster than the full Newton approach.

Testcase 2: the Olmstead model.

To obtain a larger testcase, we adapted the Olmstead model (2.58). The integral expression was discretized with a 3-point quadrature formula and the spatial derivative with a second order central difference scheme with 8 discretization points. In this way, we obtain a system of 8 DDEs with two delays. Note that this model is not a good approxima-

	$\lvert\mu_1\rvert$	$\lvert\mu_2\rvert$	$\lvert\mu_3\rvert$	$\lvert\mu_4\rvert$	T
$N = 11$	1.00695464	0.76687110	0.08439711	0.00271824	3.9986426
$N = 21$	0.99999327	0.77952649	0.08396869	0.00302678	3.9999036
$N = 41$	0.99999556	0.77999131	0.08395139	0.00291003	3.9999935
$N = 81$	0.99999958	0.78002082	0.08394678	0.00287782	3.9999996

Table 3.2: Testcase 1: dominant Floquet multipliers and period for the solution at $\alpha = 4.4745$.

Point	1	2	3	4	5	6	7	8	9	10	11	12	total
$N = 11$	86	73	56	58	64	86	109	84	90	88	82	78	854
$N = 21$	78	77	53	61	53	55	49	61	68	66	63	60	744
$N = 41$	62	61	67	60	60	55	55	59	55	60	60	57	721
$N = 81$	62	69	69	63	59	55	55	55	59	60	57	57	720

Table 3.3: Testcase 1: number of IVP solves per continuation step for 12 periodic solutions on the interval $\alpha \in [3.37, 4.38]$.

tion to the continuous model (2.58); therefore one needs to use much more discretization points, both in the spatial direction and for the integral. We used the same parameters as in section 2.5.1: $\delta = 0.1$ and $\lambda = 2$. R was used as the continuation parameter. The steady-state bifurcation diagram of this model is studied in [77]. We computed the branch emanating at the first Hopf bifurcation on the trivial steady-state solution branch $u(x, t, R) = 0$. This branch corresponds to “branch 1” in Figure 2.4. The delay interval was discretized using $N = 21$ equidistant points. We used $\rho = 0.2$. The computed branch is shown in Figure 3.10. The Floquet multipliers on the rightmost part of the branch (starting at $R = 1.25$) are shown in Figure 3.11 in terms of the pseudo-arclength. We detected three fold points and one torus bifurcation point. At the end of the computed part, the largest Floquet multiplier was larger than 100. The trivial 1 Floquet multiplier started to loose accuracy at that point and the computations failed. Note also that in the neighbourhood of the fold bifurcation points, the trivial 1 Floquet multiplier interacts with the Floquet multiplier that crosses the unit circle and two complex Floquet multipliers close to 1 are computed instead. At the fold point, M has a double 1 eigenvalue with geometric multiplicity 1. This double eigenvalue is ill-conditioned and this explains the problems. In [37, 38] the authors propose a procedure to force the trivial Floquet multiplier to be real leading to higher accuracy. We dit not yet implement an equivalent feature in our code. In Table 3.4 we show the computed values for the 6 most dominant

	$\lvert\mu_1\rvert$	$\lvert\mu_2\rvert = \lvert\mu_3\rvert$	$\lvert\mu_4\rvert$	$\lvert\mu_5\rvert = \lvert\mu_6\rvert$	T
$N = 11$	1.00047136	0.49008132	0.28830116	0.26154925	15.609848
$N = 21$	0.99998339	0.49035024	0.28834113	0.26183869	15.609690
$N = 41$	0.99999923	0.49037802	0.28834387	0.26186098	15.609658

Table 3.4: Moduli of the six most dominant Floquet multipliers and the period for the Olmstead DDE problem, $R = 0.8$.

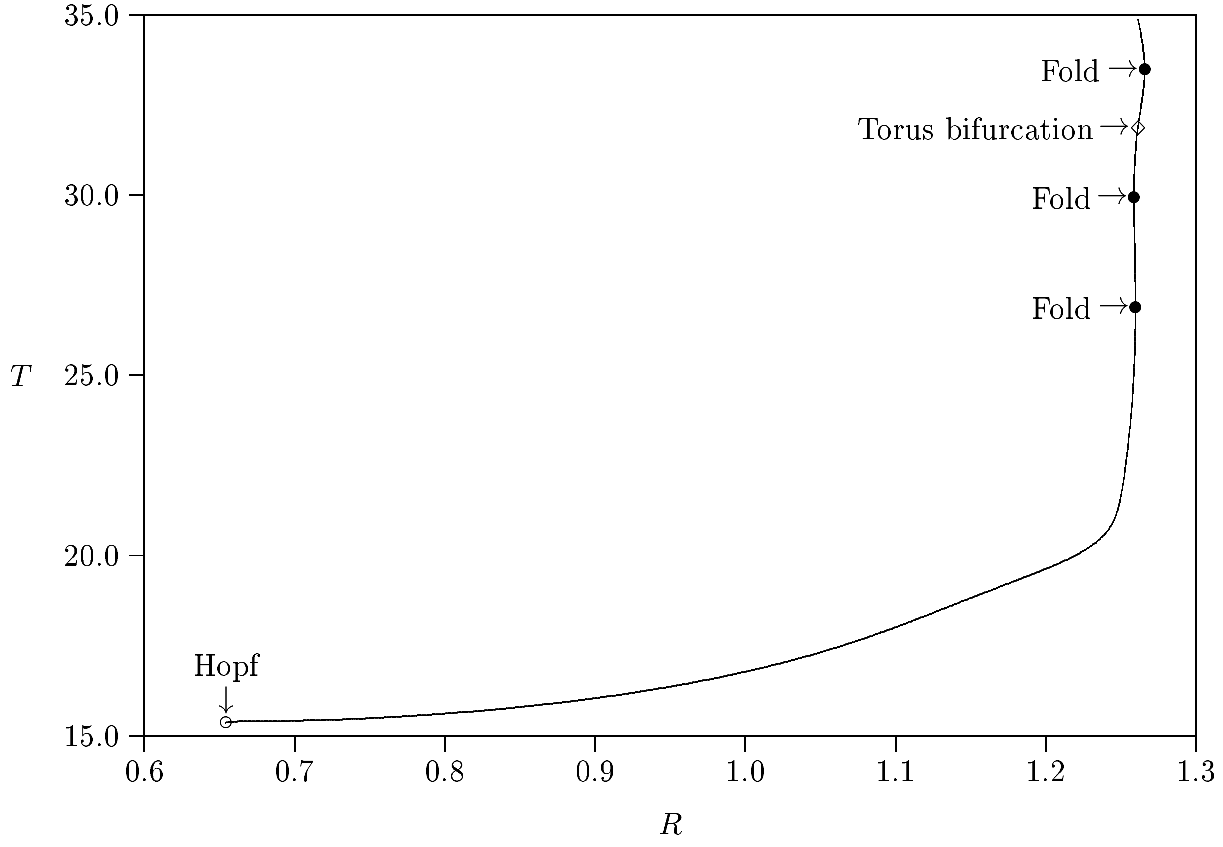


Figure 3.10: Branch of periodic solutions of the Olmstead DDE model originating at the Hopf bifurcation for $R \approx 0.6544$. (8 discretization points in the space direction, 2 delays, $N = 41$.)

Floquet multipliers and the period at $R = 0.8$ for $N = 11$, 21 and 41. Notice the excellent convergence towards 1 of the trivial Floquet multiplier and the quick convergence of the other quantities. In Table 3.5 we show the number of time integrations for the computation of 12 points on the interval $[0.820, 0.967]$. A matrix–vector product is again counted as one time integration. For $N = 41$ the construction of the full matrix M in (3.48) would require $8 \times 41 = 328$ time integrations. If two or three Jacobian matrices need to be constructed in the full Newton approach before the Jacobian matrix can be kept fixed and if the Floquet multipliers are computed from the monodromy matrix using the QR algorithm, the computation of one point with full Newton is more expensive than the computation of the whole segment with our Newton–Picard method.

Point	1	2	3	4	5	6	7	8	9	10	11	12	total
$N = 11$	88	88	88	88	88	88	88	88	88	88	88	124	1138
$N = 21$	75	85	85	85	85	85	85	85	75	85	85	85	1000
$N = 41$	81	81	81	81	81	81	81	89	89	95	95	100	1035

Table 3.5: Number of time integration for the segment $[0.820, 0.967]$ of the Olmstead DDE test problem.

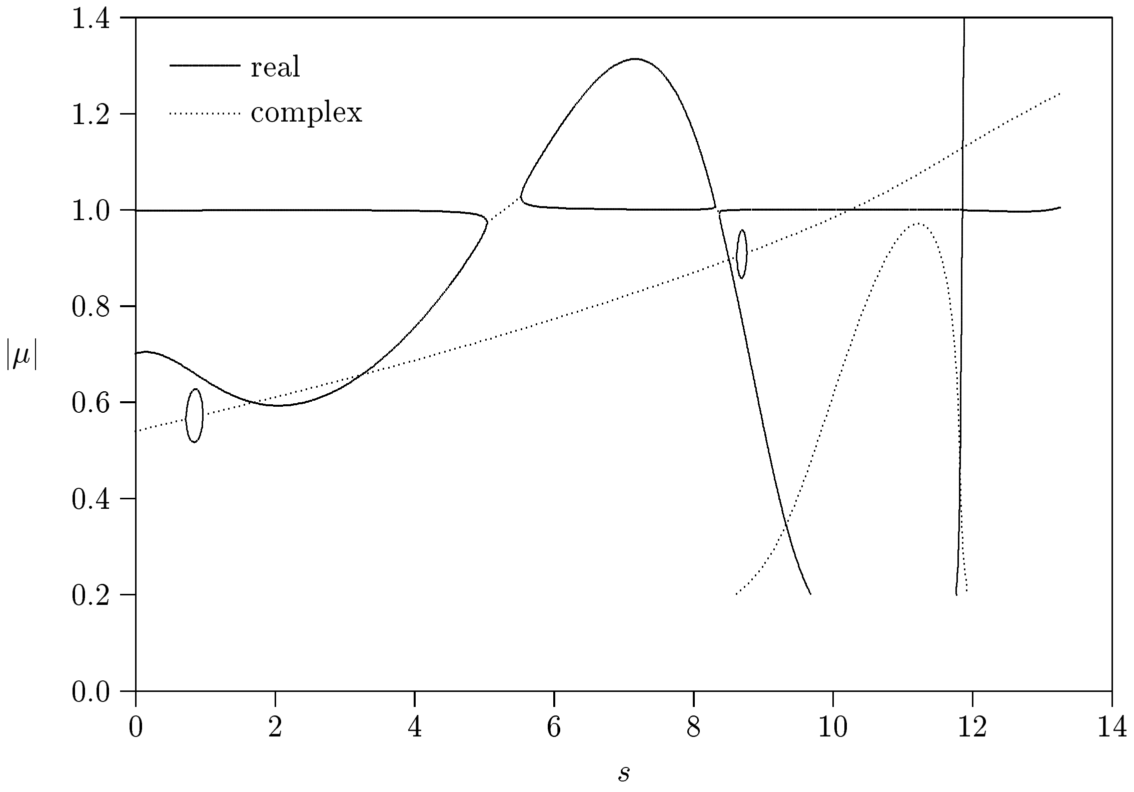


Figure 3.11: Floquet multipliers along part of the branch of periodic solutions of the Olmstead DDE model originating at the Hopf bifurcation for $R \approx 0.6544$. The figure starts at $R = 1.25$. s denotes an approximation to the arclength. (8 discretization points in the space direction, 2 delays, $N = 41$.)

3.10 Conclusions

In this chapter, we investigated strategies to increase the robustness of the single shooting Newton–Picard method presented in the previous chapter.

First we slightly extended the Moore–and–Spence-like approach to take into account a small term that was neglected in chapter 2. In this method, two high-dimensional subsystems have to be solved if an orbit at a predetermined value of the parameter is computed and three subsystems if pseudo-arclength continuation is used. This method leads to an algebraic framework in which we can fit the methods studied in chapter 2 including the methods that neglect certain coupling terms. In this framework, we developed a relationship between the final residual of a Newton–Picard step, the accuracy of the basis and the results of the solvers of the various subsystems and the higher-order terms that are neglected in the Newton linearization. This relationship is a very powerful tool and can be used at the beginning of a step to set convergence thresholds for the basis computation process and the iterative solvers for the high-dimensional subsystems, and at the end of a step to analyze why a step did not give the desired result. This allows us to develop much more robust methods. The relationship also clearly shows the influence of neglecting the coupling between the high- and low-dimensional subsystems or other terms in the projected system. The convergence behaviour of the Shroff–and–Keller-like $NPJ(l)$ and $CRP(l)$ methods is often very irregular. The latter methods usually need a better basis than the $NPGS$ and CNP methods to get a result in the same number of Newton–Picard steps.

In the new framework we do not have to use the same number of Picard steps anymore for each of the high-dimensional subsystems. Instead, based on the initial residual and the goal for the step, we determine a convergence threshold and iterate until that goal is met. We also showed in this chapter that the Picard iteration can be replaced by a GMRES solver. This solver can often produce a similar result as the Picard scheme with fewer matrix–vector products. In fact, the pure GMRES method (without restart) will never perform worse than the Picard iteration method. We also studied the use of a least-squares based solver to solve the low-dimensional subsystem. In this case, the pseudo-arclength equation can be omitted. The phase condition can also be omitted, but our tests have shown that this reduces the robustness of the method. We have also shown that it is possible to use an integral phase condition in single shooting methods, although it is not easy to make an efficient implementation.

At the end of the chapter, we presented some test results that illustrate the increased robustness. We have also shown that the method can be extended to the computation and path-following of periodic solutions of systems of ordinary differential equations with one or more finite discrete delays (DDEs).

In [56], Jarausch and Mackens also developed a strategy to control the convergence behaviour and to determine whether they should solve a high-dimensional subsystem (= 1 Picard step), a low-dimensional subsystem or improve the basis. However, their method is for a symmetric problem. They obtain fully decoupled subsystems if the parameter is kept fixed and have one coupling term if pseudo-arclength is used. Our problem is nonsymmetric (even nonnormal) and this leads to a totally different strategy to control the iterations. In [58], Jarausch and Mackens further extend their strategy to use a trust region approach. We have not yet made such an extension. However, the convergence analysis returns enough information to develop such strategies and we believe this is an interesting option for further research. The algebraic framework, the practical convergence analysis, the use of this analysis to control the iterations and the use of the GMRES method to solve the high-dimensional subsystems are original contributions of this thesis. In [55], Jarausch also uses a least-squares based method. However, in his method, the phase condition is omitted. Our tests have shown that this may lead to failure of the continuation method.

The work on delay equations was done together with T. Luzyanina (Institute of Mathematical Problems in Biology, Pushchino, Russia) and K. Engelborghs (K.U.Leuven) and published in [76]. Numerical techniques for DDE problems is still a rather unexplored area and much remains to be done in that field.

Chapter 4

Steady-state solutions

Our method borrows from the work of Jarausch and Mackens [56, 57, 58] and Shroff and Keller [107]. Jarausch and Mackens developed a method for the computation of steady-state solutions of the system

$$Ax = f(x, \gamma)$$

where A is a symmetric positive definite matrix and $\partial f / \partial x$ is symmetric. They named their method the "*condensed Newton-supported Picard method*". Shroff and Keller developed the "*recursive projection method*" which generalizes the technique of Jarausch and Mackens to problems

$$x = F(x, \gamma),$$

where $\partial F / \partial x$ can be a nonsymmetric or even nonnormal matrix. Some other authors have also used the Newton–Picard idea in their work. However, all of the work done up to now except some work reported in [55] concentrates on the computation of steady-state solutions.

In this short chapter, we wish to make a comparison with other work done on Newton–Picard methods. To make it easier to compare the different methods with each other, we will first derive steady-state variants of our methods. In section 4.2 we will discuss an important problem in the application of the Newton–Picard idea to the computation of steady-state solutions of partial differential equations. We make a detailed comparison with the work by Shroff and Keller in section 4.3, and discuss some work done by Burrage, Erhel, Pohl and Williams in section 4.4. They also built on the work by Shroff and Keller and discovered the Gauss-Seidel variant at about the same time as we did but in a different manner. An interesting related idea is the use of a deflation procedure to construct a preconditioner for the GMRES method [36]. In section 4.5 we will briefly discuss the work by Jarausch and Mackens and later work by Jarausch.

4.1 The Newton–Picard approach for steady-state solutions

We consider the computation of steady-state solutions of (1.4), i.e., solutions of the nonlinear algebraic system

$$f(x, \gamma) = 0. \tag{4.1}$$

We will also compute branches of solutions of (4.1) by allowing γ to vary. This requires adding a parametrizing equation leading to the system

$$\begin{cases} f(x,\gamma) = 0, \\ n(x,\gamma;\eta) = 0. \end{cases} \tag{4.2}$$

Many methods have been developed to solve the nonlinear problem (4.1) or (4.2), or the linearized one obtained after a Newton-linearization. A discussion of these methods is outside the scope of this work.

Two techniques deserve our special attention. One can use direct methods to solve the Newton linearization of (4.1) or (4.2). If f is the result of a finite difference or finite element discretization of a PDE, the linearized systems are sparse and techniques have been developed that exploit the structure of the system. Note that most of these techniques are developed for (4.1). The parameterizing equation in (4.2) adds a bordering row and column to the system and often breaks the structure. Special solution procedures may be required to reduce this problem to the solution of two or more unbordered systems, e.g., the deflated block-elimination algorithm [17, 16].

A second option to solve (4.1) is the use of a fixed point iteration scheme (or Picard iteration scheme). Most of these methods act directly on the nonlinear equations and do not require a linearization. A Picard iteration scheme for (4.1) is a method of the type

$$x^{(\nu+1)} = F(x^{(\nu)},\gamma) \tag{4.3}$$

where $F(x,\gamma)$ is usually a simple expression that does not require the solution of linear systems, e.g., the forward Euler timestepping method

$$F(x,\gamma) = x + \Delta t\, f(x,\gamma) \tag{4.4}$$

or the time integration

$$F(x,\gamma) = \varphi(x,T,\gamma) \tag{4.5}$$

over a fixed time interval T with a time integration scheme. In the latter case one usually uses an explicit or semi-implicit scheme. If an implicit method were used, the nonlinear problem that has to be solved at each time step is almost as hard as the original problem. Consider for instance the implicit Euler timestepping scheme

$$\frac{x^{(\nu+1)} - x^{(\nu)}}{\Delta t} = f(x^{(\nu+1)},\gamma). \tag{4.6}$$

If Newton's method is used to solve (4.6), we need to solve linear systems with the matrix

$$I - \Delta t\,\frac{\partial f}{\partial x} \tag{4.7}$$

which is almost as hard as solving the original problem. Note however that $I - \Delta t\,\partial f/\partial x$ can be positive definite even if $\partial f/\partial x$ is indefinite if Δt is small enough. Hence other methods can be used to solve systems with the matrix (4.7).

Picard schemes are easy to implement; many schemes do not even require the computation of the Jacobian of f. However, they sometimes converge very slowly and usually fail to converge to unstable steady-state solutions of (1.4). Shroff and Keller [107] realized that often such codes are available and one wishes to extend the code to compute unstable solutions. They developed a method to stabilize and improve the convergence of the Picard method. Let us first discuss how we can change the methods which we proposed in chapter 2 to compute solutions of (4.1) or (4.2). We will do this for the continuation problem (4.2).

Suppose we have a Picard iteration scheme (4.3) available to compute a solution of (4.1) at a predefined parameter value. Instead of solving (4.2), we will solve the equivalent nonlinear system

$$\left\{ \begin{array}{l} r(x,\gamma) = F(x,\gamma) - x = 0 \\ n(x,\gamma;\eta) = 0 \end{array} \right. \tag{4.8}$$

for x and γ. After the Newton-Raphson linearization, we obtain the system

$$\begin{bmatrix} M - I & b_\gamma \\ c_n^T & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r \\ n \end{bmatrix} \tag{4.9}$$

with

$$\begin{bmatrix} M & b_\gamma \\ c_n^T & d_{n,\gamma} \end{bmatrix} = \frac{\partial(F,n)}{\partial(x,\gamma)}.$$

(4.9) has exactly the same structure as (2.19) except for the row and column corresponding to the derivative with respect to the period and the phase condition. We assume

Assumption 4.1 *Let $y^* = (x^*,\gamma^*)$ denote an isolated solution to (4.2) and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M(y) = \frac{\partial F}{\partial x}(y)$ for $y \in \mathcal{B}$ and denote its eigenvalues by μ_i, $i = 1, \ldots, N$. Assume that for all $y \in \mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{4.10}$$

and that no eigenvalue has modulus ρ; i.e., for all $y \in \mathcal{B}$

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

The basis V_p and V_q are defined as before (2.5-2.6). It is important to note that the bases are based on the eigenvalues of M, i.e., the linearization $\partial F/\partial x$ of the Picard scheme and not on the eigenvalues of $\partial f/\partial x$. Therefore, we do not necessarily obtain information on bifurcation points. If the method is to be used for bifurcation analysis, the Picard iteration scheme should map eigenvalues of $\partial f/\partial x$ with positive real part to eigenvalues of $\partial F/\partial x$ outside the unit circle and it should be easy to compute the eigenvalues of $\partial f/\partial x$ from the eigenvalues and corresponding eigenvectors of $\partial F/\partial x$. This condition is satisfied by both (4.4) and (4.5). The ideal Picard scheme maps the imaginary axis to the unit circle, eigenvalues of $\partial f/\partial x$ with positive real part outside the unit circle and eigenvalues of $\partial f/\partial x$ with negative real part inside the unit circle. In this case, stability changes of steady-state solutions of (1.4) correspond to stability changes of the map (4.3) and we only need to study the map (4.3). This condition is satisfied by (4.5) but not by (4.4).

Let

$$\Delta x = V_p \Delta\bar{p} + V_q \Delta\bar{q}, \quad \Delta p = V_p \Delta\bar{p} = P\Delta x, \quad \Delta q = V_q \Delta\bar{q} = Q\Delta x. \tag{4.11}$$

Inserting (4.11) into (4.9) and multiplying the first N rows of (4.9) with $\begin{bmatrix} V_q & V_p \end{bmatrix}^T$ leads to the system

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_\gamma \\ c_n^T V_q & c_n^T V_p & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ n \end{bmatrix}. \qquad (4.12)$$

This system has essentially the same structure as (2.21) and can be solved using either the Sherman–Morrison formula or the Moore–and–Spence-like approach. Suppose we use the latter option. We first have to solve the system

$$\left[V_q^T M V_q - I_q\right] \begin{bmatrix} \Delta\bar{q}_r & \Delta\bar{q}_\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r & V_q^T b_\gamma \end{bmatrix} \qquad (4.13)$$

for $\Delta\bar{q}_r$ and $\Delta\bar{q}_\gamma$. This can be done using the Picard iteration

$$\begin{cases} \Delta\bar{q}_* = 0, \\ \Delta\bar{q}_*^{[i]} = V_q^T M V_q \Delta\bar{q}_*^{[i-1]} + V_q^T r_*, \end{cases} \qquad (4.14)$$

with $r_* = r$ or $r_* = b_\gamma$. Instead of computing the $(N-p)$-dimensional vectors from (4.14), we can better compute the N-dimensional vectors $\Delta q_* = V_q \Delta\bar{q}_*$ from

$$\begin{cases} \Delta q_* = 0, \\ \Delta q_*^{[i]} = Q\left(M\, \Delta q_*^{[i-1]} + r_*\right), \quad i = 1, \ldots, l, \\ \Delta q_* = \Delta q_*^{[l]}. \end{cases} \qquad (4.15)$$

To evaluate this scheme, we do not need to compute V_q. The iteration scheme (4.15) is a linearized version of the Picard iteration scheme (4.3) projected onto the space $\mathcal{U}^\perp$. In fact, the first step of (4.15) for the right hand side $r_* = r$ is

$$\Delta q^{[1]} = Q\left(F(p^{[0]} + q^{[0]}, \gamma) - (p^{[0]} + q^{[0]})\right)$$

or

$$q^{[1]} = q^{[0]} + \Delta q^{[1]} = QF(p^{[0]} + q^{[0]}, \gamma),$$

i.e., the Q-projection of (4.3). After solving (4.13), $\Delta\bar{p}$ and $\Delta\gamma$ are computed from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T\left(b_\gamma + M\,\Delta q_\gamma\right) \\ c_n^T V_p & d_{n,\gamma} + c_n^T \Delta q_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T\left(r + M\,\Delta q_r\right) \\ n + c_n^T \Delta q_r \end{bmatrix}. \qquad (4.16)$$

The new values of x and γ are obtained from

$$\begin{aligned} x &\leftarrow x + \Delta q_r + \Delta\gamma\, \Delta q_\gamma + V_p \Delta\bar{p}, \\ \gamma &\leftarrow \gamma + \Delta\gamma. \end{aligned}$$

This method is fully equivalent to the $CNP(l)$ method for periodic solutions and will be denoted by the same name.

The $CRP(l)$ method is generalized by omitting the terms $V_p^T M V_q$ and $c_n^T V_q$ in (4.12). As before, we can reorder the operations and first compute $\Delta\bar{p}$ and $\Delta\gamma$ from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_\gamma \\ c_n^T & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ n \end{bmatrix} \qquad (4.17)$$

followed by computing Δq by the Picard scheme

$$\left\{\begin{array}{l} \Delta q = 0 \\ \Delta q^{[i]} = Q\left(M\,\Delta q^{[i-1]} + r + \Delta\gamma\, b_\gamma\right), \quad i = 1,\dots,l \\ \Delta q = \Delta q^{[l]}. \end{array}\right. \tag{4.18}$$

The $NPGS(l)$ and $NPJ(l)$ methods are also easily adapted.

- The $NPGS(l)$ scheme computes Δq from

$$\left\{\begin{array}{l} \Delta q = 0, \\ \Delta q^{[i+1]} = Q\left(M\,\Delta q^{[i]} + r\right), \quad i = 1,\dots,l, \\ \Delta q = \Delta q^{[l]}, \end{array}\right. \tag{4.19}$$

  and $\Delta\bar{p}$ from

$$\left[V_p^T M V_p - I_p\right]\Delta\bar{p} = -V_p^T\left(r + M\,\Delta q\right). \tag{4.20}$$

- The $NPJ(l)$ scheme computes Δq from (4.19) and $\Delta\bar{p}$ from

$$\left[V_p^T M V_p - I_p\right]\Delta\bar{p} = -V_p^T r. \tag{4.21}$$

Lemma 2.2 and corollary 2.3 are easily rewritten for the $NPGS(l)$ and $NPJ(l)$ schemes. Both schemes have the asymptotic convergence rate $\left|\mu_{p+1}\right|^l$.

It is clear that the algebraic framework of chapter 3 including the convergence analysis can also be adapted.

4.2 The Picard scheme for PDEs

If (1.4) is the result of a spatial discretization of a PDE system, most eigenvalues of $\partial f/\partial x$ have negative real part and few or no eigenvalues have a positive real part. Furthermore, the rightmost eigenvalues are almost independent of the discretization. If the discretization is refined, additional eigenvalues are added to the left hand side of the spectrum. This property causes problems for the selection of the Picard scheme. The convergence of simple Picard iteration schemes such as the forward Euler timestepping scheme (4.4) is often governed by both the rightmost and the leftmost eigenvalues of the spectrum. If ρ in assumption 4.1 is kept fixed, p will increase for these schemes if the discretization is refined. If we would fix p, the convergence rate of the Newton–Picard procedure would go down.

We illustrate this problem for the forward Euler scheme (4.4). This scheme converges if the condition

$$\sigma_r\left(I + \Delta t\,\frac{\partial f}{\partial x}\right) < 1$$

is satisfied. This condition can be satisfied if all eigenvalues of $\partial f/\partial x$ have a strictly negative real part by taking a small enough value for Δt. However, if some eigenvalues lie close to the imaginary axis or are large, Δt will have to be very small and the convergence speed will be low. Figure 4.1 shows the spectrum of $\partial f/\partial x$ at the trivial steady state

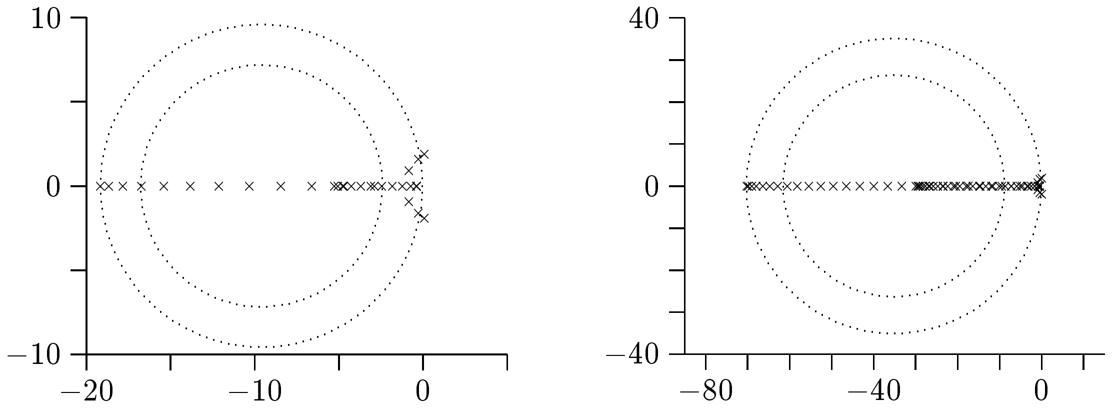


Figure 4.1: Spectrum of $\partial f/\partial x$ at the trivial equilibrium point of the Brusselator model, 15 (left) and 31 (right) discretization points, $L = 0.7$. The time step Δt is chosen such that the leftmost eigenvalue is mapped onto the unit circle by the foward Euler timestepping scheme. The circles correspond to eigenvalues of $\partial f/\partial x$ that are mapped to 1 and 0.75.

of the Brusselator model (2.55) studied in section 2.5.1 at $L = 0.7$. The value of Δt is determined such that the leftmost eigenvalue of $\partial f/\partial x$ is mapped onto the unit circle. Note that the value of Δt depends on N. The outer circle in the figure corresponds to eigenvalues of $\partial f/\partial x$ that are mapped onto the unit circle, the inner circle to eigenvalues that are mapped onto the circle in the complex plane centered at the origin with radius 0.75. The spectrum is not really clustered around 0 and the spectral picture is very different from Figure 1.2. Suppose we set $\rho = 0.75$ in assumption 4.1. All eigenvalues outside the inner circle correspond to vectors in the basis V_p and the basis size increases from 14 to 26 if the number of discretization points is increased from 15 to 31. We are not interested in the leftmost eigenvalues of $\partial f/\partial x$ since they provide no information on bifurcation points. We can make another choice for Δt so that the leftmost eigenvalue of $\partial f/\partial x$ maps onto an eigenvalue with modulus 0.75. This situation is shown in Figure 4.2. Now the dimension of $\mathcal{U}$ is 12 for $N = 15$ discretization points and 20 for $N = 31$. Although the dimension of the subspace V_p is slightly less for the latter choice of Δt, the overall computation cost may be larger since the eigenvalues are more clustered, decreasing the convergence speed of the subspace iterations. The problem of the growth of the subspace V_p is studied analytically in [20].

The more complex Picard scheme (4.5) does not have this problem if the length of the time interval is kept fixed: If $\lambda_1, \ldots, \lambda_N$ are the eigenvalues of $\partial f/\partial x$ at the equilibrium point, then

$$\mu_i = \exp(\lambda_i T), \ i = 1, \ldots, N$$

are the eigenvalues of $\partial\varphi(x, T, \gamma)/\partial x$. These eigenvalues do not depend on the time step used in the time integrator, but on the length T of the time integration interval. The Newton–Picard approach is often used to avoid the computation of $\partial f/\partial x$ and solutions of linear systems with $\partial f/\partial x$ or a matrix $\partial f/\partial x \pm \omega I$. It does not make much sense to use an implicit time integration method in this case. However, if an explicit time integration scheme is used for the time-accurate integration of (1.4), the time step has to be decreased if N is increased and the cost of the Picard scheme will grow fast.

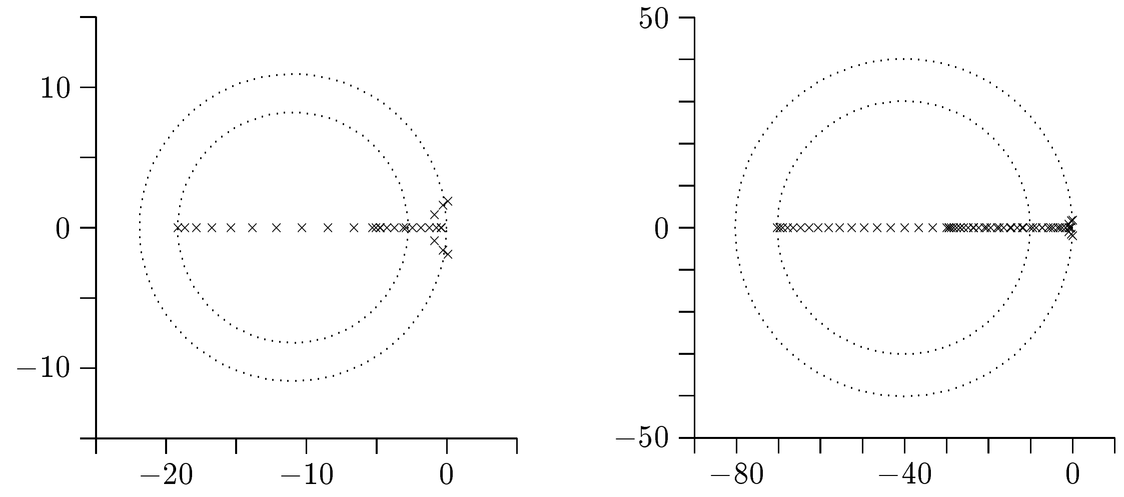


Figure 4.2: Figure similar to Figure 4.1. Now the timestep Δt is chosen such that the leftmost eigenvalue is mapped onto the circle with radius 0.75 by the foward Euler timestepping scheme. The circles correspond to eigenvalues of $\partial f \partial x$ that are mapped to 1 and 0.75.

It is clear that the Newton–Picard approach for the computation of steady-state solutions of PDEs will only be efficient with a good choice for the Picard iteration scheme. The ideal Picard iteration should compute an approximation for $\exp(T\, \partial f/\partial x)$ for a large enough value of T. The leftmost eigenvalues of $\partial f/\partial x$ should be mapped on eigenvalues of M close to 0, and if the method is to be used for bifurcation analysis also, all eigenvalues with negative real part should preferably be mapped inside the unit circle and all eigenvalues with positive real part should certainly be mapped outside the unit circle. Semi-implicit schemes where the implicit part is much easier to solve than linear systems of the form $\partial f/\partial x + \omega I$ might result in good Picard schemes. Some of the preconditioners used to compute the rightmost eigenvalues of a matrix (see, e.g., [81]) are also related to time integration techniques and might lead to interesting new Picard iteration schemes. For instance, integrating

$$\frac{dx}{dt} = Ax$$

with the θ-rule

$$\frac{x^{(\nu+1)} - x^{(\nu)}}{\Delta t} = \theta A x^{(\nu+1)} + (1-\theta) A x^{(\nu)}$$

(this is the trapezoidal rule if $\theta = 0.5$) leads to

$$x^{(\nu+1)} = \frac{\theta - 1}{\theta} \left(A - \frac{1}{\theta\, \Delta t} I \right)^{-1} \left(A - \frac{1}{(\theta-1)\, \Delta t} I \right) x^{(\nu)} \tag{4.22}$$

which resembles the Cayley transform

$$T_C(\sigma, \tau) = (A - \sigma I)^{-1} (A - \tau I).$$

In fact, we have that

$$x^{(\nu+1)} = \frac{\theta - 1}{\theta} T_C \left(\frac{1}{\theta\, \Delta t}, \frac{1}{(\theta - 1)\, \Delta t} \right) x^{(\nu)}$$

in (4.22). Of course, solving systems with the matrix $A - \sigma I$ is almost as hard as solving systems with the matrix A, although $A - \sigma I$ can be definite even if A is indefinite. It is certainly worth to investigate whether inexact rational transformations could lead to efficient Picard iteration schemes.

Another interesting approach is described in [20]. The author studies the computation of steady-state solutions of the semilinear evolution equation

$$\frac{\partial x}{\partial t} = Lx + g(x, \gamma)$$

where L is a linear differential operator including boundary conditions and g is a nonlinear mapping. He suggests to use the resolvent operator $(I - \omega L)^{-1}$ of L to "precondition" the forward Euler timestepping scheme and uses the Picard scheme

$$x_N^{[i+1]} = x_N^{[i]} + \omega \left(I_N - \omega L_N\right)^{-1} \left(L_N x_N^{[i]} + g_N(x_N^{[i]}, \gamma)\right) \tag{4.23}$$

where x_N is the discrete representation of x on a grid with N discretization points. The dominant eigenvalues of the linearization of this scheme are almost independent of N for larger values of N. The basis size needed in the Newton–Picard method depends on the value of ω but is almost independent of N if N is sufficiently large. Let G_N denote the linearization of g_N at a fixed point of (4.23). There exists a relation between the eigenvalues of $L_N + G_N$ and $(I_N - \omega L_N)^{-1}(I_N + \omega G_N)$ and for that reason with the eigenvalues of $I + \omega(I_N - \omega L_N)^{-1}(L_N + G_N)$. However, this relation is complex and has little practical use. It is also shown in [20] that, under certain conditions, eigenvalues of $L_N + G_N$ on the imaginary axis map to eigenvalues of $(I_N - \omega L_N)^{-1}(L_N + G_N)$ very close to the imaginary axis. This relation can be used to detect bifurcation points along the computed branch.

4.3 The recursive projection method

The recursive projection method (RPM) was proposed by Shroff and Keller in [107]. Their starting point is different from ours. Shroff and Keller assume that a Picard iteration scheme is available to compute steady-state solutions (4.1) and want to use the scheme to compute a bifurcation diagram. However, in certain parameter regions, the scheme does not converge and they want to stabilize the iteration by using the Picard scheme as a "black box" with as little additional work as possible. Their main purpose is the stabilization of the Picard procedure and the acceleration of the convergence speed. Our starting point is different. We start from a Newton-based shooting technique which converges well but is too expensive, and our goal is to derive a cheaper alternative by trying to combine the Newton iterations with a Picard iteration. In some of our variants for periodic solutions, we use the time integrator as a "black box", but we also developed variants that use a modified time integrator to integrate the variational equations. The different starting point and different application implies different trade-offs in our methods. Furthermore, starting from a Newton method instead of a Picard scheme made us discover several extension to RPM.

Let us first briefly describe the basic ideas behind RPM. Shroff and Keller start from an assumption that is very similar to assumption 4.1 and define bases V_p and V_q and corresponding projectors in the same way as in our work. (4.3) is projected on both spaces, leading to

$$\begin{cases} \text{(a) } q^{(\nu+1)} \leftarrow QF(p^{(\nu)} + q^{(\nu)}, \gamma), \\ \text{(b) } p^{(\nu+1)} \leftarrow PF(p^{(\nu)} + q^{(\nu)}, \gamma). \end{cases} \tag{4.24}$$

(4.24a) is a convergent Picard scheme. (4.24b) is replaced by the Newton update

$$\left[V_p^T M V_p - I_p\right] \Delta\bar{p}^{(\nu)} = -V_p^T \left(F(p^{(\nu)} + q^{(\nu)}, \gamma) - p^{(\nu)}\right),$$

$$p^{(\nu+1)} = p^{(\nu)} + V_p \Delta\bar{p}^{(\nu)},$$

for the nonlinear system $PF(p+q,\gamma) - p = 0$. This method is identical to our $NPJ(1)$ scheme. To derive a pseudo-arclength variant, Shroff and Keller use a parametrizing equation in the subspace $\mathcal{U}$, i.e., $c_s^T V_q = 0$. They first compute $\Delta\bar{p}^{(\nu)}$ and $\Delta\gamma^{(\nu)}$ from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_\gamma \\ c_s^T & d_{s,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{p}^{(\nu)} \\ \Delta\gamma^{(\nu)} \end{bmatrix} = - \begin{bmatrix} V_p^T (F(p^{(\nu)} + q^{(\nu)}, \gamma^{(\nu)}) - p^{(\nu)}) \\ n(p^{(\nu)} + q^{(\nu)}, \gamma^{(\nu)}) \end{bmatrix} \tag{4.25}$$

and then update q using one step of the Picard iteration

$$q^{(\nu+1)} = QF(p^{(\nu)} + q^{(\nu)}, \gamma^{(\nu)}) + Qb_\gamma \,\Delta\gamma^{(\nu)}. \tag{4.26}$$

Shroff and Keller present an extensive convergence proof of the fixed parameter variant. For the variant with varying parameter γ, they state that it is possible to give conditions for contractivity based on the Bauer–Fike theorem (see, e.g., [44]), but do not give those conditions in their paper since the conditions have no practical value.

We started from the Newton–Raphson scheme (4.9) instead of the Picard scheme. This lead to several extensions to RPM:

- We have found that the term $V_p^T M V_q$ plays an important role in the convergence behaviour. Although the term has no influence on the asymptotic convergence speed of our $NPGS$ scheme versus the NPJ scheme, there is a serious influence on the observed convergence behaviour away from the solution and with an imperfect basis as we have seen in section 3.3. We developed two new variants, the $NPGS$ and the CNP schemes, that take into account the term $V_p^T M V_q$ and are more robust.

- In [107], one step of a nonlinear Picard scheme is used at each RPM step. We used a linearization of the nonlinear Picard iteration scheme (although we have done some tests with a nonlinear scheme) and used for multiple Picard steps at each Newton–Picard step. For the particular starting value $\Delta q = 0$ in (4.19), the first step corresponds to a step with the nonlinear scheme.

- Shroff and Keller showed that $V_p^T b_\gamma^* \notin \text{Range}(V_p^T M^* V_p - I_p)$ if (x^*, γ^*) is a simple fold of $x = F(x,\gamma)$ and thus of (4.1) if $V_p^T M^* V_p$ is normal. Hence it is possible to choose the parametrizing equation to assure the non-singularity of (4.25) in this case. Lemma 2.5 and 2.6 provide us with a stronger expression that also holds for nonnormal matrices. The equivalent expression of lemma 2.6 in terms of steady-state solutions is

$$b^*_\gamma \in \operatorname{Range}(M^* - I)$$
$$\underline{\text{iff}}$$
$$V_p^T \left(b^*_\gamma - M V_q \left(V_q^T M^* V_q - I_q \right)^{-1} V_q^T b^*_\gamma \right) \in \operatorname{Range}(V_p^T M^* V_p - I_p).$$

Hence we can show that the P-system (4.16) is always nonsingular at a simple fold point provided Δq_γ is computed with enough accuracy and the parametrizing equation is well chosen. In fact, the P-system (4.16) will be nonsingular if (4.9) is nonsingular (again provided Δq_γ is computed with enough accuracy).

There are also differences in the basis computation process. In [107], the authors do one subspace iteration step after each continuation step to build a basis for the next continuation step. They suggest that the term QMV_p—this term should vanish—can be monitored to assure the accuracy, but did not implement this in their code. They claim that the strategy with one subspace iteration step was robust enough in their tests. They do not use extra vectors to detect if basis vectors should be added, but monitor the convergence speed of the Picard procedure and add one or two vectors if the convergence slows down too much. Note that to use this strategy, one must trust the accuracy of the basis V_p, since not only keeping necessary basis vectors out of the basis, but also inaccurate basis vectors can cause bad convergence of the Picard iterations. Shroff and Keller show that successive Picard updates are similar to a power iteration (up to higher order terms) and use this fact to detect the number of vectors to add and to derive starting values for the vectors. To detect a decrease of the basis size, they monitor the eigenvalues of $V_p^T M V_p$, just as we do.

The strategy used in [107] is very efficient, but is not completely robust if the method is used to detect bifurcation points. Their strategy is only able to detect a change in the basis size if the residual r in (4.8) has a component in the corresponding direction, otherwise the eigendirection does not influence the convergence of the Picard iteration. In their second testcase, they detect a Hopf bifurcation too late because of that reason. In the neighbourhood of the bifurcation, the solution is orthogonal to the eigenvectors for the complex eigenvalue pair crossing the imaginary axis and this orthogonality is preserved by the secant predictor. Note that it only makes sense to derive starting values from the Picard updates at successive RPM steps if the Jacobian matrix $F_x(x, \gamma)$ does not change too much. This is the case in the applications described in [107]. In fact, they could fix the projected Jacobian matrix $V_p^T M V_p$ during the iterations. In our tests the corrector iterations failed if the projected monodromy matrix was fixed early during the iterations.

Our strategy for basis updates is designed to sacrifice some efficiency in favor of robustness. At the beginning of the Newton–Picard iterations for a periodic orbit, we add some random perturbations to the basis V_p to assure that new vectors can easily enter the basis. Furthermore we do one or more subspace iteration steps until QMV_p is sufficiently small. To detect a basis size increase, we compute some additional eigenvalues that are smaller than ρ in modulus. The additional vectors are also used to accelerate the convergence of the dominant vectors in the subspace iterations by using subspace iteration with projection. Locking of converged eigenvectors is used to further increase the efficiency. In more recent versions of the code based on the convergence analysis of chapter 3, we always perform a subspace iteration step at the beginning of each Newton–Picard iteration step to check the accuracy of the basis. This strategy makes sense if a high convergence speed is desired since the monodromy matrix and the basis change considerably at each step, but it is too expensive at low convergence speeds. As an alternative, we could fix the basis and the projected monodromy matrix towards the end of the iterations and only check the accuracy of the basis by computing $QM\Delta p$ after the Newton–Picard step if the results of the step are not satisfactory. If we find out that new vectors should be added to the basis, random vectors are added to the basis. It may make sense to derive starting values for those vectors from the sequence of Picard iterations done during one Newton–Picard step. Note that it is not trivial to use this idea, since the most dominant vectors that are not in V_p are already included in the extended basis V_e and we need to derive starting values for the next most dominant vectors. Further fine-tuning of the basis computation process is certainly possible and worth to investigate, since this is an expensive step in the Newton–Picard method.

We have also derived a powerful framework for monitoring the convergence behaviour. This framework gives us a complete understanding of the observed convergence behaviour and also allows us to substitute the Picard iteration with a different iterative solver and to develop methods that are very robust and still remain efficient.

The work of [107] was extended to the computation of steady-state solutions of differential-algebraic equations in [115, 64]. Compared to [107] some subtle fine-tunings in the strategy to increase the basis size are done, but the main ideas have not changed. Further research on the method is going on at Caltech.

4.4 The work of Burrage, Erhel, Pohl and Williams

In [13], the method of [107] is applied to linear systems using Jacobi or Gauss–Seidel iterations as the underlying Picard scheme. The authors also present some extensions to the method of [107], one of which corresponds to our $NPGS(1)$ scheme.

Suppose we wish to solve the linear system

$$Ax = b. \tag{4.27}$$

Let

$$A = M_L - M_R = L + R + D$$

with L the strictly lower triangular part of A (i.e., with zeros on the diagonal), R the strictly upper triangular part of A and D the diagonal of A. The Jacobi and Gauss–Seidel schemes can both be written as

$$M_L x^{(\nu+1)} = M_R x^{(\nu)} + b. \tag{4.28}$$

For the Jacobi scheme

$$M_{L,J} = D \text{ and } M_{R,J} = -L - R$$

and for Gauss–Seidel

$$M_{L,GS} = L + D \text{ and } M_{R,GS} = -R.$$

Iteration (4.28) can be rewritten as

$$x^{(\nu+1)} = Mx^{(\nu)} + M_L^{-1}b = F(x^{(\nu)}) \tag{4.29}$$

with

$$M = M_L^{-1}M_R.$$

Hence the Jacobi- and Gauss–Seidel schemes are equivalent to a fixed point iteration of type (4.3). Let V_p denote an approximation to the basis for the maximal invariant subspace of M for the dominant eigenvalues of M. The numerical RPM scheme computes

$$\left\{ \begin{array}{l} \text{(a) } p^{(\nu+1)} = h(p^{(\nu)} + q^{(\nu)}), \\ \text{(b) } q^{(\nu+1)} = g(p^{(\nu)} + q^{(\nu)}), \end{array} \right. \tag{4.30}$$

with

$$\left\{ \begin{array}{l} g(p^{(\nu)} + q^{(\nu)}) = \left(I - V_pV_p^T\right)\left(Mq^{(\nu)} + Mp^{(\nu)} + M_L^{-1}b\right), \\ h(p^{(\nu)} + q^{(\nu)}) = V_p\left(I_p - V_p^TMV_p\right)^{-1}V_p^T\left(Mq^{(\nu)} + M_L^{-1}b\right). \end{array} \right.$$

The Q-step (4.30b) corresponds to an iteration step with (4.29) projected on $(\text{Span}\{V_p\})^\perp$. In (4.30a), $p^{(\nu+1)} = V_p\bar{p}^{(\nu+1)}$, with $\bar{p}^{(\nu+1)}$ solved from

$$\bar{p}^{(\nu+1)} = V_p^TMV_p\bar{p}^{(\nu+1)} + V_p^T\left(Mq^{(\nu)} + M_L^{-1}b\right),$$

i.e., (4.29) projected onto $\text{Span}\{V_p\}$. One step of (4.30) is equivalent to our $NPJ(1)$ method. The authors suggest two possible Gauss–Seidel type variants:

$$\left\{ \begin{array}{l} p^{(\nu+1)} = h(p^{(\nu)} + q^{(\nu)}), \\ q^{(\nu+1)} = g(p^{(\nu+1)} + q^{(\nu)}), \end{array} \right. \tag{4.31}$$

called the *Gauss–Seidel* scheme, and

$$\left\{ \begin{array}{l} q^{(\nu+1)} = g(p^{(\nu)} + q^{(\nu)}), \\ p^{(\nu+1)} = h(p^{(\nu)} + q^{(\nu+1)}), \end{array} \right. \tag{4.32}$$

called the *reverse Gauss–Seidel* scheme. One step of the reverse Gauss–Seidel scheme is equivalent to our $NPGS(1)$ scheme applied to the linear system (4.27) and the Picard scheme (4.29):

$$\begin{array}{rcl} q^{(\nu+1)} = q^{(\nu)} + \Delta q^{(\nu)} & = & q^{(\nu)} + \left(I - V_pV_p^T\right)\left(M\left(p^{(\nu)} + q^{(\nu)}\right) + M_L^{-1}b - \left(p^{(\nu)} + q^{(\nu)}\right)\right) \\ & = & \left(I - V_pV_p^T\right)\left(M\left(p^{(\nu)} + q^{(\nu)}\right) + M_L^{-1}b\right) \end{array}$$

and

$$\begin{array}{rcl} p^{(\nu+1)} = p^{(\nu)} + \Delta p^{(\nu)} & = & p^{(\nu)} + V_p\left(V_p^TMV_p - I_p\right)^{-1}V_p^T\left(-M\left(p^{(\nu)} + q^{(\nu)}\right)\right. \\ & & \left. -M_L^{-1}b + p^{(\nu)} - M\Delta q^{(\nu)}\right) \\ & = & p^{(\nu)} + V_p\left(I_p - V_p^TMV_p\right)^{-1}V_p^T\left(Mq^{(\nu+1)} - M_L^{-1}b\right) \\ & & +V_p\left(V_p^TMV_p - I_p\right)^{-1}\left(I_p - V_p^TMV_p\right)V_p^Tp^{(\nu)} \\ & = & V_p\left(I_p - V_p^TMV_p\right)^{-1}V_p^T\left(Mq^{(\nu+1)} - M_L^{-1}b\right). \end{array}$$

The Gauss–Seidel scheme corresponds to neglecting the term $V_p^T M V_q$ in (4.12) but taking into account the term $V_q^T M V_p$ which vanishes with a perfect basis and is usually small. The authors derive formulas for the spectrum of the iteration matrix

$$\frac{\partial(h,g)}{\partial(p,q)}$$

and show that for a perfect basis all three schemes have the same convergence rate. With an imperfect basis, the iteration matrices of the two Gauss–Seidel variants have the same spectral properties. These two methods also compute the same sequence of updates but with different starting and finishing values. The Gauss–Seidel scheme started from $x + g(x)$ produces the same sequence as the reverse Gauss–Seidel scheme started from x. $q^{(\nu)}$ computed by the Gauss–Seidel method corresponds to $q^{(\nu+1)}$ produced by the reversed Gauss–Seidel scheme and the value of $p^{(\nu)}$ is the same for both schemes.

If we would apply one step of the schemes of [13] to solve the linearized systems resulting from the Newton–Raphson method, we would notice an important difference between the two Gauss–Seidel variants if the matrix M were nonnormal. The Gauss–Seidel scheme (4.31) neglects the term $V_p^T M V_q$ in (4.12) which can be large if M is nonnormal but takes into account the term $V_q^T M V_p$ which is very small if the basis is accurate enough. We expect that the results produced by this scheme will not differ much from the results produced by (4.30) and for that matter our $NPJ(1)$ scheme. The reverse Gauss–Seidel scheme (4.32) (equivalent to our $NPGS(1)$ scheme) neglects the usually small term $V_q^T M V_p$ but takes into account the often large term $V_p^T M V_q$. As we have seen in section 3.3, this has considerable consequences for the convergence of Newton's method. Note that this does not contradict the results of [13] since that paper considers the asymptotic convergence behaviour for linear systems, while we are only interested in the first step for a linear system and the long-term behaviour of Newton's method for a nonlinear problem.

Because of the different problem, there is also a large difference in the way the basis V_p is computed. If a linear system is solved, the matrix M in (4.27) does not change and there is no need for updates of the basis vectors. In [13] the authors start with an empty basis V_p and isolate one or two vectors from the Q-updates every n_{iter} iterations until a predetermined number of vectors is isolated. In their testcases, they get bad results if n_{iter} is too small. This is easily explained as follows. The successive Δq's converge to the most dominant eigendirections of $\left(I - V_p V_p^T\right) M$. However, if n_{iter} is small, there are still large contributions of the other eigenvectors in the last Q-updates and the vectors that are isolated do not span an invariant subspace of that matrix. The basis is inaccurate and the terms $V_q^T M V_p$ and $V_p^T M V_q$ are rather large, even if M is a symmetric or normal matrix. Also, $r_\sigma\left(V_q^T M V_q\right)$ can be larger than expected.

To compare the three variants, they consider the two-dimensional Poisson equation on the unit square with Dirichlet boundary conditions discretized with the usual second order finite difference scheme and use Jacobi relaxation as the underlying Picard scheme. This example is not very well chosen in order to show the difference between the three schemes. As the iteration matrix M for this problem is symmetric, all three methods are fully equivalent when a perfect basis is used, and we can only observe differences in the convergence behaviour caused by the basis imperfection. Further testing on harder testcases is only done with the reverse Gauss–Seidel method.

The reverse Gauss–Seidel method is also used in [14, 118, 119] to solve a generalized cross validation problem that requires the repetitive solution of related linear problems.

A different but interesting idea is studied in [36]. In this paper, the authors consider a preconditioner for the flexible GMRES method (FGMRES) of [102]. The FGMRES method is a restarted GMRES method that uses a different preconditioner after every restart. In [36], a preconditioner is constructed by extracting eigenvalue and eigenvector information on the smallest eigenvalues (in modulus) from the GMRES method at every restart. Suppose we wish to solve (4.27). Let V_p be an invariant subspace of dimension p of A corresponding to the *smallest* eigenvalues and let V_q be a basis for the orthogonal complement of Span(V_p). Let $S = V_p^T A V_p$. A is preconditioned using the matrix

$$\begin{aligned} P_C &= \left(\begin{bmatrix} V_p & V_q \end{bmatrix} \begin{bmatrix} S/|\lambda_N| & 0 \\ 0 & I_{N-p} \end{bmatrix} \begin{bmatrix} V_p & V_q \end{bmatrix}^T \right)^{-1} \\ &= \begin{bmatrix} V_p & V_q \end{bmatrix} \begin{bmatrix} |\lambda_N|\, S^{-1} & 0 \\ 0 & I_{N-p} \end{bmatrix} \begin{bmatrix} V_p & V_q \end{bmatrix}^T \end{aligned}$$

where λ_N is the estimate for the *largest* eigenvalue of A. The GMRES method is applied to the system

$$P_C\, A\, x = P_C\, b.$$

Note that a matrix–vector product $P_C\, v$ is easily computed since

$$\begin{aligned} P_C\, v &= |\lambda_N|\, V_p S^{-1} V_p^T v + \left(I - V_p V_p^T\right) v \\ &= v + V_p \left(|\lambda_N|\, S^{-1} - I_p\right) V_p^T v. \end{aligned}$$

We do not need to compute V_q. At the end of the GMRES step, the smallest eigenvalues and corresponding eigenvectors are isolated from the projection H of $P_C\, A$ on the Krylov space and added to the basis V_p and $S = V_p^T A V_p$ is recomputed. The method shows excellent behaviour compared to the unpreconditioned restarted GMRES method. In terms of iterations, the method performs almost as well as the unpreconditioned full GMRES method, but a large reduction in memory requirements and CPU time is observed.

4.5 The work of Jarausch and Mackens

Jarausch and Mackens developed the "*condensed Newton - supported Picard*" method in the early 80's [56, 57, 58]. Their intention is to compute solutions to

$$Ax = f(x, \gamma) \tag{4.33}$$

where A is a symmetric positive definite matrix and $\partial f / \partial x$ is a symmetric matrix. They suppose that an efficient solver for A is available. In many applications, the Picard iteration scheme

$$x^{(\nu+1)} = A^{-1} f(x^{(\nu)}, \gamma) \tag{4.34}$$

has no or only a few divergent directions. Jarausch and Mackens wish to combine this scheme with a Newton method in the divergent or slowly convergent directions to obtain a convergent iteration scheme. Note that although A and $\partial f / \partial x$ are both symmetric matrices, $A^{-1} \partial f / \partial x$ is usually a nonsymmetric matrix (and even nonnormal). The methods proposed by Jarausch and Mackens are similar to our $NPGS$ and CNP schemes if $A = I$ (note that in this case, $NPGS = NPJ$ and $CNP = CRP$), but for other matrices A, they still obtain a full decoupling of (4.33) in two subsystems by using A-orthonormal bases. Our approach applied to (4.34) does not result in a full decoupling if $A^{-1}\partial f/\partial x$ is a nonnormal matrix. Let us briefly discuss the main ideas behind their method.

Suppose

$$|\mu_1| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|$$

are the eigenvalues of $A^{-1}\partial f/\partial x$ Let $v_1, \ldots, v_p$ be a set of A-orthonormal eigenvectors of $A^{-1}\partial f/\partial x$ corresponding to the p dominant eigenvalues and let $V_p = \left[\begin{array}{ccc} v_1 & \cdots & v_p \end{array}\right]$. Note that $V_p^T A V_p = I_p$. Projectors are constructed according to

$$\begin{aligned} P &= V_p V_p^T A, \\ Q &= I - P. \end{aligned}$$

Let

$$x = p + q, \ p = Px = V_p\bar{p}, \ q = Qx.$$

(4.33) is equivalent to

$$p = PA^{-1}f(p+q,\gamma), \tag{4.35a}$$
$$q = QA^{-1}f(p+q,\gamma). \tag{4.35b}$$

(4.35a) can be rewritten is terms of $\bar{p}$:

$$\bar{p} = V_p^T f(q + V_p\bar{p}, \gamma). \tag{4.36}$$

This equation is solved for $\bar{p}$ using Newton's method [56] or a minimization method for

$$\phi(\Delta\bar{p}) = \left\| \bar{h}(\bar{p}, q) + \partial h/\partial\bar{p}\, \Delta\bar{p} \right\|_2^2$$

with

$$\bar{h}(\bar{p}, q) = \bar{p} - V_p^T f(q + V_p\bar{p}, \gamma).$$

In [58] Powell's dogleg step (see [93]) is used to solve the minimization problem. This method selects the update $\Delta\bar{p}$ in the plane spanned by the Newton update for (4.36) and the gradient direction of $\phi(\Delta\bar{p})$.

In [56], Jarausch and Mackens show that a small change of q has little influence on the residual of (4.35a) and similarly, a small change of p has little influence on the residual of (4.35b) if the basis V_p is accurate enough. This decoupling is exploited in the algorithm presented in [56]. If the residual of (4.35a) is larger than that of (4.35b), a P-step (update of p, e.g., by doing a Newton-Raphson step on (4.36)) is performed, otherwise a Q-step (a Picard step $q \leftarrow QA^{-1}f(p+q,\gamma)$) is taken. If the corresponding residual does not decrease enough or if the other residual increases too much, the basis is updated by a subspace iteration step based on A-orthonormal bases. They also propose a scheme for pseudo-arclength continuation. They do not use an explicit parametrizing equation but solve the underdetermined linear system resulting from a Newton-Raphson linearization of (4.36) using the least-squares approach with the generalized inverse. In the P-step, they also update q according to

$$q \leftarrow q + \Delta\gamma \left(I - QA^{-1}\,\partial f/\partial x\right) QA^{-1}\,\partial f/\partial\gamma. \tag{4.37}$$

This update corresponds to our component Δq_γ in the update Δq in the algebraic framework of chapter 3. Jarausch and Mackens suggest that using

$$q \leftarrow q + \Delta\gamma\, QA^{-1}\partial f/\partial\gamma \tag{4.38}$$

is a sufficiently good approximation. (4.38) corresponds to doing one step of the Picard scheme

$$\Delta q_\gamma \leftarrow QA^{-1}\frac{\partial f}{\partial x}\,\Delta q_\gamma + QA^{-1}\,\frac{\partial f}{\partial\gamma}$$

using the starting value $\Delta q_\gamma^{[0]} = 0$. By applying the update (4.37) during the P-step, they avoid a significant increase of the residual of (4.35b).

The method is extended to compute bifurcation points in [57] by using an extended system on the projected Jacobian matrix. Their algorithm is an iterative algorithm. At every step, a singular point of the desired type of (4.36) is computed, than the residual of (4.35b) is reduced using Picard iterations and finally the basis V_p and $q_\gamma = (I - QA^{-1}\,\partial f/\partial x)\,QA^{-1}\,\partial f/\partial\gamma$ are updated.

In [58], the method is fine-tuned to solve (4.33) at a given parameter value γ without a good starting value. The iteration scheme presented in [56] is combined with damping of the Q- and P-steps based on trust regions (see [84] for an exposition of such techniques). They show that the resulting algorithm is globally convergent under certain assumptions on the nonlinearity.

In [54] Jarausch studies schemes to split ODE systems resulting from the discretization of parabolic differential equations in two subsystems. Different numerical techniques can be used to integrate the two subsystems. The approach can also cope with problems with a nonsymmetric Jacobian. Two splittings are proposed. One uses orthogonal projectors but does not lead to a full decoupling and the other uses oblique projectors and leads to a full decoupling. The basis is time dependent. The idea behind the splitting with oblique projectors is also usable for solving linear systems and therefore we shall discuss it here.

The splitting is done according to the real part of the eigenvalues of $\partial f/\partial x$: the (generalized) eigenvectors for the rightmost eigenvalues are used to construct the basis V_p and a basis W_p for the corresponding left (generalized) eigenvectors is also constructed. Jarausch suggests two possible splittings:

1. A splitting using the orthogonal projectors $P = V_pV_p^T$ and $Q = I - V_pV_p^T$. This splitting is equivalent to our splitting except for the choice of the basis V_p and leads to a block triangular structure. The P-subsystem depends on the Q-subsystem.

2. A splitting based on the oblique projectors

$$\begin{aligned} P &= V_pV_p^T, \\ Q &= I - V_p\left(W_p^TV_p\right)^{-1}W_p^T. \end{aligned}$$

Here $x = p + q$ with $p = Px \in \mathrm{Span}(V_p)$ and $q = Qx \in \mathrm{Span}(W_p)^\perp$. This splitting leads to a complete decoupling of the ODE system in two subsystems.

However, it requires the computation of left eigenvectors of the matrix $\partial f / \partial x$. This is not always an easy task since it is often easy to compute matrix–vector products $\partial f / \partial x \, v$ but much harder to compute those products with the transpose of the matrix. Oblique projectors also lead to less stable numerical methods.

In both cases, the splitting is applied to the nonlinear system $f(x)$ directly as in [107].

Note that the idea of strictly splitting according to the real part of the eigenvalues of $\partial f / \partial x$ is very interesting for bifurcation analysis purposes, but as we have discussed before is not very appealing if the only purpose is to solve a sequence of related linear systems to solve the nonlinear equation $f(x) = 0$. In the latter case, we believe that it is a better approach to first select a Picard scheme that has not too many divergent or slowly converging directions and then build the basis based on the Picard iterations scheme instead of first selecting the basis and then looking for a Q-system solver that works. However, the main interest of [54] is in splitting the differential equation and developing time integration techniques. Then the splitting isolates the growing and slowly decaying solution components in the P-system and the rapidly decaying modes in the large Q-system. The P-system has a slowly evolving solution but may contain growing components, while the Q-system has a rapidly decaying solution but is stiff since the eigenvalue λ_N goes off to $-\infty$ as N is increased. It requires implicit or semi-implicit time integration techniques.

A different approach is taken in [55]. In this paper, Jarausch studies the computation of both steady-state solutions and periodic solutions of parabolic PDEs. The periodic solutions are computed using single shooting. As in our approach, the PDE system is first space-discretized and all computations start from (1.4). To compute steady-state solutions, he considers the Picard iteration

$$\begin{bmatrix} \gamma \\ x \end{bmatrix} \leftarrow \begin{bmatrix} \gamma \\ L^{-1}\left(L\,x - f(x,\gamma)\right) \end{bmatrix} \tag{4.39}$$

where L is some easy invertible approximation to $\partial f / \partial x$. Periodic solutions are computed using the Picard iteration

$$\begin{bmatrix} \gamma \\ T \\ x(0) \end{bmatrix} \leftarrow \begin{bmatrix} \gamma \\ T \\ \varphi(x(0), T, \gamma) \end{bmatrix}. \tag{4.40}$$

Let u denote the set of variables ($y = \begin{bmatrix} \gamma & x^T \end{bmatrix}^T$ or $y = \begin{bmatrix} \gamma & T & x(0)^T \end{bmatrix}^T$). We have to compute solutions to

$$y = F(y)$$

or

$$r(y) = F(y) - y = 0. \tag{4.41}$$

Jarausch wants to obtain a full local decoupling of the P- and Q-system while using only orthogonal projectors and other orthogonal transformations for the sake of numerical stability.

Let N denote the dimension of the vector y. Let

$$A = \frac{\partial r}{\partial y}$$

evaluated at the starting point of the iteration step. Let $V_p \in \mathbb{R}^p$ and $V_q \in \mathbb{R}^{N-p}$ be orthogonal matrices corresponding to orthogonal right singular subspaces of the matrix A and let U_p and U_q denote bases for the corresponding left singular subspaces. In the final scheme, the bases V_q and U_q will not be needed. We will discuss later which singular vectors should go in the basis V_p. Let $P = V_p V_p^T$ and $Q = V_q V_q^T = I - P$ be orthogonal projectors on $\operatorname{Span}(V_p)$ and $\operatorname{Span}(V_p)^\perp$ respectively. The matrix A acting on a vector in $\operatorname{Span}(V_p)$ results in a vector of $\operatorname{Span}(U_p)$ and similarly, $\operatorname{Span}(V_q)$ is mapped onto $\operatorname{Span}(U_q)$. Jarausch also defines an orthogonal mapping J ("rotator") that maps vectors from $\operatorname{Span}(V_p)$ in $\operatorname{Span}(U_p)$ and vectors from $\operatorname{Span}(V_p)^\perp$ in $\operatorname{Span}(U_p)^\perp$ and describes a method to compute such a transformation using only V_p and U_p. It is essential that the rotator J satisfies

$$J_{|V_p} = U_p V_p^T$$

where $J_{|V_p}$ denotes the mapping J restricted to $\operatorname{Span}(V_p)$.

Let

$$y = Py + Qy = V_p \bar{p} + q \text{ with } q = Qy. \tag{4.42}$$

(4.41) is equivalent to $J^T r(y) = 0$ since J is nonsingular. After projection with P and Q and substitution of y with (4.42) we obtain

$$\begin{aligned} 0 &= U_p^T r(V_p \bar{p} + q), && \text{(4.43a)} \\ q &= q + QJ^T r(V_p \bar{p} + q). && \text{(4.43b)} \end{aligned}$$

(4.43a) and (4.43b) are locally decoupled since the partial derivatives of (4.43a) with respect to q and the partial derivatives of (4.43b) with respect to $\bar{p}$ are all zero. The basis V_p is constructed such that (4.43a) is a small system and (4.43b) gives lead to a contractive Picard iteration. Jarausch proves that T and γ in (4.39) and (4.40) give both lead to a vector in the basis V_p. Note that the Jacobian of (4.43a) has one zero row for (4.39) and two zero rows for (4.40). The corresponding singular vectors can be chosen so that γ and T are components of $\bar{p}$. The P-system (4.43a) is solved using the Gauss-Newton method (Newton combined with the SVD-based least squares approach). To assure that (4.43b) leads to a contractive Picard iteration scheme, the basis V_p has to be computed such that

$$\sigma_{\max}\left(\left(I + J^T A\right)_{|V_q}\right) = \sigma_{\max}\left((J + A)_{|V_q}\right) < \rho$$

where $\rho < 1$ is a predetermined threshold. Note that V_p and V_q are invariant subspaces of $J^T A$ and right singular subspaces of $J + A$. New vectors are added to the basis by performing some subspace iteration steps on $Q(J + A)^T (J + A) Q$ and adding the most dominant vectors to V_p. Jarausch suggests two methods to refine the basis V_p and the corresponding basis U_p. He also gives a strategy to decrease the dimension of the basis. It should be noted that matrix–vector products with A^T are needed to compute the singular subspaces. In our method, we do not need such products. Jarausch shows how to compute these products if A comes from a time discretization of (1.4), but the procedure is difficult and requires complete knowledge of the internals of the time integration code.

The efficiency of the method is demonstrated by computing a stable periodic solution of the 1-dimensional Brusselator model (2.55) using the same parameter values as we did at $L = 0.8$. In this case, V_p contained four vectors. This basis lead to

$$\sigma_{\max}\left(\left(I + J^T A\right)_{|V_q}\right) = 0.2390.$$

The actual contraction rates were even better.

I believe that our approach for the computation of periodic solutions does offer several advantages over the method described in [55] despite the fact that it does not offer a full decoupling.

- The Floquet multipliers are readily available and it is easy to monitor the stability. Recovering that information from the technique described by Jarausch in [55] requires a separate postprocessing step. The SVD used in his method is based on the bordered matrix and preserves singular values but not eigenvalues in the P-system. Hence it is not possible to compute the dominant Floquet multipliers from the P-system. Information on transcritical and pitchfork bifurcations is contained in the P-system. These types of bifurcation points are characterized by rank conditions for the bordered matrix $[\varphi_x - I \quad \varphi_T \quad \varphi_\gamma]$. Furthermore, since γ and T are contained in the vector $\bar{p}$, we expect that information on fold bifurcation points can also be recovered from the P-system. It is not clear whether information on period doubling points and torus bifurcation points is also available.

- We have more freedom in the choice of the phase condition and parametrizing equation. The Gauss–Newton method used in [55] to solve the P-system will automatically select a phase condition and a parametrizing equation just as our approach based on least squares does if no phase- or pseudo-arclength condition is specified. However, we are free to specify our own conditions. For instance, we can use an integral phase condition that helps to avoid too frequent remeshing if a time integration code with adaptive stepsize is used. Furthermore, as we already discussed in section 3.8, we sometimes noted difficulties with the automatic phase condition chosen by the SVD-based least-squares approach if branches of periodic solutions are computed with a secant predictor.

- The construction of the bases is considerably more difficult than in our case. We do not need matrix–vector products with the transpose of a Jacobian matrix.

- We have also shown that our approach can be generalized to multiple shooting without a large increase in computation time (see chapter 6) and have also shown that our technique can be easily extended to compute various types of bifurcation points (See chapter 7).

The main advantages of the approach of [55] over our approach are

- Because the splitting is based on singular subspaces and because of the way the basis is determined, Jarausch can assure that the Q-system will show monotone convergence without initial blow-up.

- The complete decoupling makes it also somewhat easier to develop strategies to control the residuals of the P- and Q-equations.

4.6 Conclusions

The purpose of this chapter was to compare our methods with work done by other authors. Since most related methods have been developed for the computation of steady-state solutions, we started the chapter with the construction of some variant of our method for steady-state problems. Here the splitting is based on the spectrum of the Jacobian of a Picard iteration scheme and not on the spectrum of the Jacobian of $f(x, \gamma)$ itself. This has consequences if the method is to be used for bifurcation analysis.

There can be efficiency problems if the Newton–Picard method is applied to a steady-state problem. The convergence of the Newton–Picard method and the dimension of the low-dimensional subspace is determined by the convergence behaviour of the Picard iteration scheme. Simple methods such as forward Euler timestepping have a lot of eigenvalues close to the unit circle. Moreover, that number and hence the basis size for a given convergence rate increases if the discretization is refined. An ideal Picard scheme would approximate $\exp(T\partial f/\partial x)$ at the equilibrium point for some positive value of T. Such a scheme is also interesting for bifurcation analysis since steady-state bifurcation points corresponds to eigenvalues of the linearized Picard iteration scheme that cross the unit circle. However, such methods are hard to construct since the scheme should be much easier to evaluate than it is to solve a system $\partial f/\partial x\, \Delta x = -r$. Time-accurate time integration methods do lead to an interesting spectrum. However, explicit time integration methods (used for instance in [107]) are not good candidates since ODE systems resulting from the space discretization of a PDE are stiff. Furthermore, if the discretization is refined, smaller time steps must be used and the number of time steps increases for a fixed value of T. Fully implicit time integration formulas are often almost as hard to evaluate as it is to solve a system $\partial f/\partial x\, \Delta x = -r$. An interesting approach to obtain an efficient Picard iteration is studied in [20].

We described the basic ideas behind the Recursive Projection method of Shroff and Keller [107]. Their starting point is very different from ours and this leads to some differences with our techniques. They wish to stabilize a Picard iteration scheme that ceases to converge for certain parameter ranges, while we start from a Newton-based method and wish to reduce the cost of that method without loosing too much of its favorable convergence properties. Shroff and Keller project the nonlinear Picard iteration scheme and replace the low-dimensional projection with a Newton step. The high-dimensional projection of the Picard iteration scheme is contractive in its subspace. Shroff and Keller did not notice the coupling term between both subsystems. This term can have a big influence on the convergence. In this thesis, we develop new methods that take this term into account. Our methods also allow for multiple Picard iteration steps before every Newton step. To compute the basis, Shroff and Keller use orthogonal subspace iterations. The basis size is increased if the convergence of the Picard scheme slows down. Their strategy is designed for maximum efficiency. Our strategy is very different and designed to be as robust as possible at a reasonable cost. We have found that in our applications, a good basis accuracy is important. Shroff and Keller also present a condition to distinguish between a fold point and a pitchfork- or transcritical bifurcation point. However, that theorem is limited to the case of a normal Jacobian. In this case, there is no coupling term if the parameter is kept fixed. Our projected system in the Moore–and–Spence-like variants contains all information to make this distinction also in the nonnormal case.

We compared with work done by Burrage, Erhel, Pohl and Williams. They used the method to solve linear systems and stabilize and accelerate the Jacobi or Gauss–Seidel iteration schemes. They developed two Newton–Picard Gauss–Seidel variants. One first solves the low-dimensional system and uses that result in the high-dimensional system and the other one first solves the high-dimensional system. The latter variant corresponds to one of our methods. The asymptotic convergence speed of both methods is the same. However, the convergence behaviour is very different in the first step and precisely this behaviour is very important in our method since we apply only one Newton–Picard step to solve the Newton linearization and then proceed with a slightly different system. They also developed an interesting variant of the flexible GMRES method that uses spectral information gathered from the GMRES iterations to construct a new preconditioner at each restart.

We made an extensive comparison with the work of Jarausch and Mackens. They solve a very different (symmetric) problem and use subspaces that are A-orthogonal (with A a symmetric positive definite matrix). Fully decoupled projected systems are obtained in the fixed parameter case. To compute a basis for their low-dimensional subspace, they use a variant of the orthogonal subspace iteration method. They developed very nice strategies to control the convergence of the method. However, the strategies rely on a decoupling property that does not hold in our case. In [57] their method is extended to compute some types of bifurcation points.

In later work, Jarausch studied several splittings for nonsymmetric problems. In [54] he studies two splittings to split a system of differential equations. One leads to a complete decoupling but is based on oblique projectors, while the other uses numerically more stable orthogonal projectors but does not decouple the systems. In [55] Jarausch studies a splitting based on singular subspaces that leads to a complete decoupling. However, these subspaces are harder to compute than eigenspaces. He also uses a unifying approach that make no distinction between the system variables and the parameters. The method is applied to both steady-state problems and single shooting. Both the pseudo-arclength equation and phase condition are defined implicitly. We have shown in the previous chapter that such an approach may lead to a failure of the continuation method. Also, because of the unifying approach, less information is preserved on bifurcations. In contrast to our method, it is impossible to cheaply recover the Floquet multipliers from the computations.

Chapter 5

Accurate computation of Floquet multipliers

This chapter deals with the accurate computation of Floquet multipliers in multiple shooting codes and in the Gauss–Legendre collocation based package AUTO. In the traditional approaches, the monodromy matrix is constructed explicitly and the eigenvalues are computed using the QR-algorithm. As pointed out by Fairgrieve and Jepson in [37, 38], there are problems with the accuracy of the smaller eigenvalues if the ratio of the smallest Floquet multiplier to the largest $|\mu_N| / |\mu_1|$ is of the order of the machine precision or smaller. It is particularly important to be able to compute the Floquet multipliers in the neighbourhood of the unit circle with high accuracy in order to be able to detect bifurcation points also in the presence of very large Floquet multipliers. Fairgrieve and Jepson propose a solution to this problem which extends the range of the computable Floquet multipliers. As we shall see, this solution also has its limitations and only postpones the problem if some of the Floquet multipliers continue to increase along a branch of periodic solutions. We will show that the Floquet multipliers can be computed with very high accuracy by using the periodic Schur decomposition proposed in [11]. We will briefly discuss the QR algorithm for the reader who is not familiar with this algorithm. Next we will discuss the particular case of the periodic QR algorithm for multiple shooting codes. This algorithm is an extension of the QR algorithm and computes the periodic Schur decomposition in a numerically stable way. We also show a result for an artificial set of matrices to demonstrate the potential of the method.

5.1 The periodic Schur decomposition

The Floquet multipliers are the solution to the eigenvalue problem

$$Mv = \mu v \tag{5.1}$$

where M is the monodromy matrix. Since the product Mv is the solution at time T of the initial value problem

$$\frac{dv(t)}{dt} = \left.\frac{\partial f(x,\gamma)}{\partial x}\right|_{(\varphi(x(0),t,\gamma),\gamma)} v(t), \quad v(0) = v,$$

(5.1) can be rewritten as the boundary value problem

$$\begin{cases} \dfrac{dv(t)}{dt} = \left.\dfrac{\partial f(x,\gamma)}{\partial x}\right|_{(\varphi(x(0),t,\gamma),\gamma)} v(t), \\ v(T) = \mu v(0). \end{cases} \tag{5.2}$$

This boundary value problem can be solved using the multiple shooting method on the time mesh (1.27). This leads to the system

$$\begin{bmatrix} G_1 & -I & & & \\ & G_2 & -I & & \\ & & \ddots & \ddots & \\ & & & G_{m-1} & -I \\ -\mu I & & & & G_m \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{m-2} \\ v_{m-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{5.3}$$

where G_i is given by (1.30) and we have to determine the finite values of μ at which this system is singular. Now let

$$A = \begin{bmatrix} G_1 & -I & & \\ & \ddots & \ddots & \\ & & G_{m-1} & -I \\ & & & G_m \end{bmatrix}, \ B = \begin{bmatrix} 0_{N\times N} & \cdots & \cdots & 0_{N\times N} \\ \vdots & & & \vdots \\ 0_{N\times N} & 0_{N\times N} & & \vdots \\ I & 0_{N\times N} & \cdots & 0_{N\times N} \end{bmatrix}, \ v = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{m-1} \end{bmatrix}, \tag{5.4}$$

then (5.3) can be rewritten as the generalized eigenvalue problem

$$Av = \mu Bv. \tag{5.5}$$

This problem has an infinite eigenvalue repeated $(m-1)N$ times. The other N eigenvalues are precisely the Floquet multipliers. Suppose μ_j is an eigenvalue of (5.5) and

$$v^{(j)} = \begin{bmatrix} v_0^{(j)} \\ \vdots \\ v_{m-1}^{(j)} \end{bmatrix}$$

the corresponding eigenvector. From (5.4) and (5.5) we get

$$G_m \cdots G_1 v_0^{(j)} = \mu_j v_0^{(j)}, \tag{5.6}$$

i.e., $v_0^{(j)}$ is the eigenvector of $M(x(0),T,\gamma) = G_m \cdots G_1$ for the Floquet multiplier μ_j. Note that all matrices G_i are nonsingular. This follows from lemma 1.2. Hence (5.6) is equivalent to

$$G_k \cdots G_1 G_m \cdots G_1 v_0^{(j)} = \mu_j G_k \cdots G_1 v_0^{(j)}$$

and since $v_k^{(j)} = G_k \cdots G_1$ we have

$$G_k \cdots G_1 G_m \cdots G_{k+1} v_k^{(j)} = \mu_j v_k^{(j)}$$

or $v_k^{(j)}$ is the eigenvalue of $M(x(s_kT),T,\gamma)$ for the eigenvalue μ_j. Solving the generalized eigenvalue problem (5.5) thus returns the Floquet multipliers and the corresponding eigenvectors of the monodromy matrices at the startpoints $x(s_kT)$ of all multiple shooting intervals.

A similar observation can be made if (5.2) is discretized with the Gauss–Legendre collocation method used in AUTO (see section 1.4.3). After the first step of the linear system solver in AUTO—the condensation of the parameters—we obtain the system

$$\begin{bmatrix} G_1 & -H_1 & & \\ & \ddots & \ddots & \\ & & G_m & -H_m \\ -\mu I & & & I \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{m-1} \\ v_m \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}. \tag{5.7}$$

This corresponds to the generalized eigenvalue problem

$$\begin{bmatrix} G_1 & -H_1 & & \\ & \ddots & \ddots & \\ & & G_m & -H_m \\ & & & I \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{m-1} \\ v_m \end{bmatrix} = \mu \begin{bmatrix} 0_{N\times N} & \cdots & \cdots & 0_{N\times N} \\ \vdots & & & \vdots \\ 0_{N\times N} & 0_{N\times N} & & \vdots \\ I & 0_{N\times N} & & 0_{N\times N} \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{m-1} \\ v_m \end{bmatrix}. \tag{5.8}$$

This problem has Nm eigenvalues at infinity and the other ones are again the Floquet multipliers. Now we have

$$H_m^{-1}G_m \cdots H_1^{-1}G_1 v_0 = \mu v_0,$$

i.e., the Floquet multipliers are the eigenvalues of the matrix product

$$H_m^{-1}G_m \cdots H_1^{-1}G_1. \tag{5.9}$$

Note that in analogy to the multiple shooting case, v_k in (5.8) is an eigenvector of the monodromy matrix $M(x(s_kT),T,\gamma)$ and the generalized eigenvalue problem returns the eigenvectors at each of the collocation mesh boundaries simultaneously.

In most codes the product $G_m \cdots G_1$ or (5.9) is implicitly or explicitly constructed and the eigenvalues of the matrix product are computed using the QR algorithm. In [38] Fairgrieve and Jepson demonstrate that this procedure can return inaccurate results for the smaller Floquet multipliers. As an eigenvalue gets smaller and smaller compared to the largest one, more and more accuracy is lost. Information on the smaller Floquet multipliers gets lost during the computation of the matrix products and the subsequent run of the QR algorithm to compute the Schur factorization. This is easy to understand since the small eigenvalues are computed from large matrix entries and subtraction is used in the process. We shall demonstrate later in this section that it is impossible to compute eigenvalues smaller than $\varepsilon_{mach}\,|\mu_1|$ where μ_1 is the largest eigenvalue and ε_{mach} is the machine precision. Fairgrieve and Jepson propose an other reduction of a generalized eigenvalue problem equivalent to (5.8). They propose to compute the eigenvalues and eigenvectors from the generalized eigenvalue problem

$$G_0 v = \mu H_0 v \tag{5.10}$$

	$G_0x = \mu H_0 x$		$Mx = \mu x$	
exact	computed	rel. error	computed	rel. err.
10e+10	1.000002020560038e+10	2.02e-06	9.999999999999969e+09	3.05e-15
1e+00	9.999999999203655e-01	7.96e-11	1.000000282853425e+00	2.82e-07
10e-10	9.999983579230538e-11	1.64e-06	1.147964523340317e-05	1.15e+05

Table 5.1: Eigenvalues computed by theMatlab version 5 `qr()` function. The construction of the matrices A, B and C is described in the text.

where G_0 and H_0 are given by (1.45). G_0 and H_0 are computed by the AUTO linear system solver. (5.10) is solved using the QZ algorithm [82]. The QZ algorithm will compute the generalized real Schur decomposition, i.e., orthogonal matrices Q and Z are constructed such that $A = Q^T G_0 Z$ is quasi-upper triangular (i.e., has 2×2-blocks on the diagonal corresponding to complex eigenvalue pairs of (5.10)) and $B = Q^T H_0 Z$ is upper triangular. A. Real eigenvalues are computed as the ratio a_{ii}/b_{ii} of corresponding diagonal elements of A and B. Complex eigenvalue pairs are also easily computed from corresponding blocks. The ratio of the largest to the smallest values on the diagonals of A and B can be much smaller than for the diagonal elements of the Schur decomposition of $H_0^{-1}G_0$. However, if the largest eigenvalue becomes too large or the smallest too small, problems will appear again. This is easily demonstrated with the following experiment in Matlab version 5 on a machine with standard IEEE double precision arithmetic (i.e., ≈ 16 digits).

1. Let
$$D_1 = \begin{bmatrix} 10^5 & & \\ & 1 & \\ & & 10^{-5} \end{bmatrix}, \; D_2 = \begin{bmatrix} 10^{-5} & & \\ & 1 & \\ & & 10^5 \end{bmatrix}.$$

2. Y_1 and Y_2 are random nonsingular 3×3-matrices.

3. $G_0 = Y_1 D_1 Y_2$, $H_0 = Y_1 D_2 Y_2$ and $M = H_0^{-1} G_0 = Y_2^{-1} D_2^{-1} D_1 Y_2 = Y_2^{-1} D_1^2 Y_2$.

The eigenvalues of the generalized eigenvalue problem $G_0 x = \mu H_0 x$ and the eigenvalue problem $Mx = \mu x$ are 10^{10}, 1 and 10^{-10}. The eigenvectors depend on Y_2. The nondiagonal entries of $Y_2 Y_2^{-1}$ were all around $2\varepsilon_{mach}$. The eigenvalues returned by the Matlab version 5 `qr()` function are shown in table 5.1. Note that for the QR algorithm, the largest eigenvalue is correct up to machine precision, the eigenvalue 1 is found with approximately 6 correct digits and the eigenvalue 10^{-10} is lost. Instead, a value around $10^{10}\varepsilon_{mach}$ is returned. Using the QZ algorithm, the eigenvalue 1 is found with twelve accurate digits and both the eigenvalues 10^{10} and 10^{-10} have six to seven accurate digits. If the QR algorithm is used, the smallest eigenvalues have the largest errors. If the generalized eigenvalue problem is solved with the QZ algorithm, the extreme eigenvalues have the largest error. These are computed as the ratio of a large and a small number and the smallest diagonal entries of $A = QH_0Z$ and $B = QH_0Z$ have the largest relative error of all diagonal entries.

In [11] Bojanczyk, Golub and Van Dooren show the following lemma.

Lemma 5.1 *Let G_i, $i = 1, \ldots, m$ and H_i, $i = 1, \ldots, m$ be real $N \times N$-matrices. Then there exist orthogonal $N \times N$-matrices Q_i, $i = 0, \ldots, m-1$ and Z_i, $i = 1, \ldots, m$ such that*

$$\begin{aligned} \hat{G}_i &= Z_i^T G_i Q_{i-1}, \\ \hat{H}_i &= Z_i^T H_i Q_i, \quad Q_m := Q_0, \end{aligned}$$

where all matrices $\hat{G}_i$ and $\hat{H}_i$ are upper triangular except the matrix $\hat{G}_m$. $\hat{G}_m$ is quasi-upper triangular with 1×1 blocks corresponding to real eigenvalues of (5.9) and 2×2-blocks corresponding to complex eigenvalues.

(The indices starting at 0 for the matrices Q_i may seem awkward at this moment. The numbering corresponds to the indices used for the unknowns in the multiple shooting and collocation systems and will be useful in the next chapter.)

Proof. See [11]. □

Since

$$\begin{aligned} & Q_0^T H_m^{-1} G_m \cdots H_1^{-1} G_1 Q_0 \\ & \quad = Q_0^T H_m^{-1} Z_m Z_m^T G_m Q_{m-1} Q_{m-1}^T H_{m-1}^T Z_{m-1} \cdots Q_1^T H_1^{-1} Z_1 Z_1^T G_1 Q_0 \\ & \quad = \hat{H}_m^{-1} \hat{G}_m \cdots \hat{H}_1^{-1} \hat{G}_1 \end{aligned}$$

is a quasi-upper triangular matrix,

$$\left(H_m^{-1} G_m \cdots H_1^{-1} G_1\right) Q_0 = Q_0 \left(\hat{H}_m^{-1} \hat{G}_m \cdots \hat{H}_1^{-1} \hat{G}_1\right)$$

is the Schur decomposition of

$$H_m^{-1} G_m \cdots H_1^{-1} G_1.$$

Note that

$$\begin{aligned} & Q_{k-1}^T H_{k-1}^{-1} G_{k-1} \cdots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_k^{-1} G_k Q_{k-1} \\ & \quad = \hat{H}_{k-1}^{-1} \hat{G}_{k-1} \cdots \hat{H}_1^{-1} \hat{G}_1 \hat{H}_m^{-1} \hat{G}_m \cdots \hat{H}_k^{-1} \hat{G}_k \end{aligned}$$

is also quasi-upper triangular. Hence

$$\begin{aligned} & \left(H_{k-1}^{-1} G_{k-1} \cdots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_k^{-1} G_k\right) Q_{k-1} \\ & \quad = Q_{k-1} \left(\hat{H}_{k-1}^{-1} \hat{G}_{k-1} \cdots \hat{H}_1^{-1} \hat{G}_1 \hat{H}_m^{-1} \hat{G}_m \cdots \hat{H}_k^{-1} \hat{G}_k\right) \end{aligned}$$

is the Schur decomposition of

$$H_{k-1}^{-1} G_{k-1} \cdots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_k^{-1} G_k.$$

Hence the decomposition of lemma 5.1 is called the periodic Schur decomposition. If $G_m[i,i]$ is not part of a 2×2-block,

$$\hat{H}_m[i,i]^{-1} \hat{G}_m[i,i] \cdots \hat{H}_1[i,i]^{-1} \hat{G}_1[i,i]$$

is a real eigenvalues of (5.9). If $\hat{G}_m[i{:}i+1, i{:}i+1]$ is a 2×2-block we have to construct the 2×2-block

$$\hat{H}_m[i{:}i+1, i{:}i+1]^{-1} \hat{G}_m[i{:}i+1, i{:}i+1] \cdots \hat{H}_1[i{:}i+1, i{:}i+1]^{-1} \hat{G}_1[i{:}i+1, i{:}i+1]. \quad (5.11)$$

From this block we can compute a pair of complex conjugate eigenvalues. Note that the matrices $\hat{H}_i[i : i+1, i : i+1]$ are upper triangular matrices, hence the product (5.11) is easy to compute (at least if none of the factors $\hat{H}_i$ is singular, which is the case in our applications).

In this thesis, we are particularly interested in the special case $H_i = I$, $i = 1, \ldots, m$, i.e., in the eigenvalue problem (5.6). In this case it is possible to show

Corollary 5.2 *Let the matrices G_i, $i = 1, \ldots, m \in \mathbb{R}^{N \times N}$ all be real. Then there exist orthogonal $N \times N$-matrices Q_i, $i = 0, \ldots, m-1$, such that the matrices*

$$\hat{G}_i = Q_i^T G_i Q_{i-1}, \; Q_m := Q_0 \tag{5.12}$$

are all upper triangular except $\hat{G}_m$. $\hat{G}_m$ is a quasi-upper triangular matrix with 1×1 and 2×2-blocks on the diagonal corresponding to real and complex conjugate eigenvalues respectively of $G_m \cdots G_1$.

Proof. Follows immediately from lemma 5.1 with $H_i = I$ and $Z_i = Q_i$. □

Note that (5.12) corresponds to the transformation

$$Q^T A Z y = \mu Q^T B Z y$$

of (5.5) where

$$Q = \begin{bmatrix} Q_1 & & & \\ & \ddots & & \\ & & Q_{m-1} & \\ & & & Q_0 \end{bmatrix}, \; Z = \begin{bmatrix} Q_0 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & Q_{m-1} \end{bmatrix}$$

are orthogonal matrices. $Q^T B Z = B$ and

$$Q^T A Z = \begin{bmatrix} \hat{G}_1 & -I & & \\ & \ddots & \ddots & \\ & & \hat{G}_{m-1} & -I \\ & & & \hat{G}_m \end{bmatrix}.$$

Bojanczyk, Golub and Van Dooren also present an algorithm to compute the periodic Schur decomposition without explicitly computing the matrix $G_m \cdots G_1$ or (5.9). The algorithm is a generalization of the QR algorithm and is called the *periodic QR algorithm.* We have made a FORTRAN77 implementation of problem (5.6) where we took into account that all matrices G_i are nonsingular.

5.2 The QR algorithm

In this section, we will briefly introduce the basics of the QR iteration for the computation of the real Schur decomposition of a real matrix. This section can be skipped by readers familiar with the QR algorithm.

Consider the subspace iteration for the eigenvalue problem $GX = X\Lambda$ where G is a real $N \times N$-matrix:

$$\begin{aligned} V^{[0]} &= I_N, \\ V^{[i]}R^{[i]} &= GV^{[i-1]}, \end{aligned} \tag{5.13}$$

with $V^{[i]}$ an orthogonal matrix and $R^{[i]}$ an upper triangular matrix. The matrices $V^{[i]}$ will essentially converge to an orthogonal matrix V and the products ${V^{[i]}}^T GV^{[i]}$ to a quasi-upper triangular matrix H and $GV = VH$ is the real Schur decomposition of G. Note that there is no strict convergence in the mathematical sense, e.g., column vectors of $V^{[i]}$ may change sign at every step. The QR algorithm is a variant of this method and computes the matrix ${V^{[i]}}^T GV^{[i]}$ directly from ${V^{[i-1]}}^T GV^{[i-1]}$. The basic QR algorithm is given by the iteration

$$\begin{aligned} &\text{(a) } H^{[0]} = G \\ &\text{(b) } Q^{[i]}R^{[i]} = H^{[i-1]} \text{ with } Q^{[i]} \text{ an orthogonal matrix and } R^{[i]} \text{ upper triangular} \\ &\text{(c) } H^{[i]} = R^{[i]}Q^{[i]}. \end{aligned} \tag{5.14}$$

Note that $H^{[i]} = {Q^{[i]}}^T H^{[i-1]} Q^{[i]}$. The relation between the QR iteration (5.14) and the subspace iteration (5.13) is given by the following property.

Property 5.3 *Let $V^{[0]} = I$, $V^{[i]}R^{[i]} = GV^{[i-1]}$ with $V^{[i]}$ an orthogonal matrix and $R^{[i]}$ an upper triangular matrix. Let*

$$\begin{aligned} H^{[i]} &:= {V^{[i]}}^T GV^{[i]}, \quad i = 0, \dots, \\ Q^{[i]} &:= {V^{[i-1]}}^T V^{[i]}, \quad i = 1, \dots, \end{aligned}$$

then

$$\begin{aligned} &\textit{(a) } H^{[i-1]} = Q^{[i]}R^{[i]}, \\ &\textit{(b) } H^{[i]} = R^{[i]}Q^{[i]}. \end{aligned}$$

Proof. First, note that the matrix $Q^{[i]}$ is orthogonal, as required for the QR algorithm. Point (a) is proven by

$$\begin{aligned} V^{[i]}R^{[i]} = GV^{[i-1]} \quad &\Leftrightarrow \quad \left({V^{[i-1]}}^T V^{[i]}\right) R^{[i]} = {V^{[i-1]}}^T GV^{[i-1]} \\ &\Leftrightarrow \quad Q^{[i]}R^{[i]} = H^{[i-1]} \end{aligned}$$

and point (b) by

$$\begin{aligned} V^{[i]}R^{[i]} = GV^{[i-1]} \quad &\Leftrightarrow \quad R^{[i]} = {V^{[i]}}^T GV^{[i-1]} \\ &\Leftrightarrow \quad {V^{[i]}}^T GV^{[i-1]} \left({V^{[i-1]}}^T V^{[i]}\right) = R^{[i]} \left({V^{[i-1]}}^T V^{[i]}\right) \\ &\Leftrightarrow \quad H^{[i]} = R^{[i]}Q^{[i]}. \end{aligned}$$

□

One iteration (5.14) requires $O(N^3)$ operations if H is a general square matrix. However, if $H^{[i-1]}$ is an upper Hessenberg matrix, $Q^{[i]}$ is also an upper Hessenberg matrix and so is the product $R^{[i]}Q^{[i]}$. Algorithm 7.4-1 in [44] shows that in this case, the QR-factorization (5.14b) can be computed implicitly in $O(N^2)$ operations. $Q^{[i]}$ can be computed as the product of $N - 1$ Givens rotations and requires little storage. Hence in a practical implementation the matrix G is first reduced to an upper Hessenberg matrix using an orthogonal transformation matrix, i.e., $H^{[0]} = Q^{[0]^T} G Q^{[0]}$ in (5.14) and the QR algorithm is applied to the resulting matrix. Since the QR iteration is basically a version of subspace iteration, the upper left entries of the matrix $H^{[i]}$ will converge to the largest eigenvalues and the lower right entries to the smallest eigenvalues. The convergence speed of a simple eigenvalue μ_k, $k < N$, is determined by its ratio to the next largest eigenvalue in modulus, i.e., the ratio $|\mu_{k+1}| / |\mu_k|$.

The QR iteration converges very slowly if eigenvalues are close to each other. To accelerate the convergence, we can incorporate "shifts", i.e., we replace the iteration (5.14) with

$$\begin{array}{l}\text{(a) } H^{[0]} = Q^{[0]^T} G Q^{[0]} \\ \text{(b) } Q^{[i]} R^{[i]} = H^{[i-1]} - \mu^{[i]} I \text{ with } Q^{[i]} \text{ an orthogonal matrix and } R^{[i]} \text{ upper triangular} \\ \text{(c) } H^{[i]} = R^{[i]} Q^{[i]} + \mu^{[i]} I.\end{array} \tag{5.15}$$

Note that again $H^{[i]} = Q^{[i]} H^{[i-1]} Q^{[i]}$, hence the matrix $H^{[i]}$ is similar to all other matrices $H^{[j]}$ and to G. The shifts $\mu^{[i]}$ can be real or complex numbers and can vary from iteration step to iteration step. The single-shift QR iteration (5.14) is related to shifted inverse iterations of G^T, i.e., the subspace iteration method applied to $\left(G^T - \mu I\right)^{-1}$. This relationship was observed in [44]. We have worked out this relationship in detail in the following property.

Property 5.4 *Suppose all shifts $\mu^{[i]}$ are real. Let E be the $N \times N$-matrix with ones on the antidiagonal and zeros on the other positions. Suppose all matrices $G^T - \mu^{[i]} I$ are nonsingular. Let $V^{[0]} = Q^{[0]} E$,*

$$V^{[i]} S^{[i]} = \left(G^T - \mu^{[i]} I\right)^{-1} V^{[i-1]},$$

with $V^{[i]}$ an orthogonal matrix and $S^{[i]}$ an upper triangular matrix. Let

$$\begin{aligned} H^{[i]} &:= \left(V^{[i]} E\right)^T G \left(V^{[i]} E\right), \quad i = 0, \ldots, \\ Q^{[i]} &:= \left(V^{[i-1]} E\right)^T \left(V^{[i]} E\right), \quad i = 1, \ldots, \\ R^{[i]} &:= E S^{[i]^{-T}} E, \quad i = 1, \ldots, \end{aligned}$$

then

$$\begin{array}{l}\textit{(a) } H^{[i-1]} - \mu^{[i]} = Q^{[i]} R^{[i]} \\ \textit{(b) } H^{[i]} = R^{[i]} Q^{[i]} + \mu^{[i]} I.\end{array}$$

Proof. Note $E^T = E$ and $EE = I$. Furthermore,

$$Q^{[i]} = \left(V^{[i-1]} E\right)^T \left(V^{[i]} E\right) = E \left(V^{[i-1]^T} V^{[i]}\right) E$$

is an orthogonal matrix as required by the QR algorithm. $S^{[i]}$ is an upper triangular matrix. Hence, $S^{[i]^{-T}}$ is a lower triangular matrix and $R^{[i]} = E S^{[i]^{-T}} E$ is again upper triangular. Note also that by construction, all matrices $S^{[i]}$ are nonsingular.

Point (a) is proven by

$$\begin{aligned}
&V^{[i]}S^{[i]} = \left(G^T - \mu^{[i]}I\right)^{-1} V^{[i-1]} \Leftrightarrow \\
&G^T V^{[i]}S^{[i]} - \mu^{[i]}V^{[i]}S^{[i]} = V^{[i-1]} \Leftrightarrow \\
&S^{[i]^T} V^{[i]^T} G - \mu^{[i]} S^{[i]^T} V^{[i]^T} = V^{[i-1]^T} \Leftrightarrow \\
&V^{[i]^T} G V^{[i-1]} - \mu^{[i]} V^{[i]^T} V^{[i-1]} = S^{[i]^{-T}} \Leftrightarrow \\
&V^{[i-1]^T} V^{[i]} V^{[i]^T} G V^{[i-1]} - \mu^{[i]} I = V^{[i-1]^T} V^{[i]} S^{[i]^{-T}} \Leftrightarrow \\
&\left(V^{[i-1]}E\right)^T G \left(V^{[i-1]}E\right) - \mu^{[i]}I = \left(V^{[i-1]}E\right)^T \left(V^{[i]}E\right) \left(ES^{[i]^{-T}}E\right) \Leftrightarrow \\
&H^{[i-1]} - \mu^{[i]}I = Q^{[i]}R^{[i]}
\end{aligned}$$

and point (b) by

$$\begin{aligned}
&V^{[i]}S^{[i]} = \left(G^T - \mu^{[i]}I\right)^{-1} V^{[i-1]} \Leftrightarrow \\
&S^{[i]^T} V^{[i]^T} G - \mu^{[i]} S^{[i]^T} V^{[i]^T} = V^{[i-1]^T} \Leftrightarrow \\
&S^{[i]^T} V^{[i]^T} G V^{[i]} - \mu^{[i]} S^{[i]^T} = V^{[i-1]^T} V^{[i]} \Leftrightarrow \\
&V^{[i]^T} G V^{[i]} = S^{[i]^{-T}} V^{[i-1]^T} V^{[i]} + \mu^{[i]} I \Leftrightarrow \\
&\left(V^{[i]}E\right)^T G \left(V^{[i]}E\right) = \left(ES^{[i]^{-T}}E\right) \left(V^{[i-1]}E\right)^T \left(V^{[i]}E\right) + \mu^{[i]}I \Leftrightarrow \\
&H^{[i]} = R^{[i]}Q^{[i]} + \mu^{[i]}I.
\end{aligned}$$

□

For a fixed shift $\mu^{[i]} = \mu$, the matrices $V^{[i]}$ will essentially converge to an orthogonal matrix V and the matrices $V^{[i]^T} \left(G^T - \mu I\right)^{-1} V^{[i]}$ to a matrix $\hat{H}$ with the property that $\left(G^T - \mu I\right)^{-1} V = V\hat{H}$ is the real Schur decomposition of $\left(G^T - \mu I\right)^{-1}$. It is easy to show that $G^T V = V\tilde{H}$, with $\tilde{H} = \hat{H}^{-1} + \mu I = V^T G^T V$ a quasi-upper triangular matrix is the Schur decomposition of G^T and $G\,(VE) = (VE)\left(E\tilde{H}^T E\right)$ is the Schur decomposition of G. Hence the matrices $H^{[i]}$ in the iteration (5.15) will converge to a quasi-upper triangular matrix H which has blocks corresponding to the eigenvalues of G on its main diagonal. The upper left diagonal entries (1×1 and 2×2-blocks) of $V^{[i]^T} G^T V^{[i]}$ will converge to the eigenvalues of G closest to the shift μ. Since $H^{[i]^T} = E\left(V^{[i]^T} G^T V^{[i]}\right)E$, the lower right diagonal entries of $H^{[i]}$ correspond to the upper left entries of $V^{[i]^T} G^T V^{[i]}$ and converge to the eigenvalues closest to the shift. In an actual implementation, the shifts $\mu^{[i]}$ are not fixed but adapted as better estimates for the eigenvalues of G become available. Once an eigenvalue is isolated at the bottom right of the matrix $H^{[i]}$, the eigenvalue is deflated (or "locked") and the computations proceed on the upper left part of $H^{[i]}$.

To accelerate the convergence to a pair of complex conjugate eigenvalues, complex shifts have to be used. However, we want to avoid complex arithmetic. This can be done by combining two single-shift QR iteration steps with complex conjugate shifts $\mu^{[i]}$ and $\mu^{[i+1]} = \bar{\mu}^{[i]}$ in a double-shift QR step. This procedure is outlined in section 7.5 of [44]. It can be shown that the procedure corresponds to the iteration

$$\begin{aligned}
&QR = H^2 - sH + tI, \\
&H \leftarrow Q^T H Q,
\end{aligned} \tag{5.16}$$

with $s = \mu^{[i]} + \mu^{[i+1]}$ and $t = \mu^{[i]}\mu^{[i+1]}$. This procedure can also be used with two real shifts. If the initial matrix H is an upper Hessenberg matrix, the final matrix H is also upper Hessenberg.

Both the single- and the double-shift QR steps can be written as

$$\begin{array}{l}\text{(a) } QR = M \\ \text{(b) } H \leftarrow Q^T H Q\end{array} \tag{5.17}$$

with $M = H - \mu I$ for the single-shift QR step and $M = H^2 - sH + hI$ with $s = \mu_1 + \mu_2$ and $h = \mu_1\mu_2$ for the double-shift QR step with two real or complex conjugate shifts μ_1 and μ_2. Of course, one wishes to avoid the explicit computation of M in a double-shift QR step and the QR factorization of M. It is possible to make a more efficient implementation of the iteration (5.17) based on the following theorem.

Theorem 5.5 Implicit Q theorem *([44], p. 223) Suppose* $Q = \left[\begin{array}{ccc} q_1 & \cdots & q_N \end{array}\right]$ *and* $V = \left[\begin{array}{ccc} v_1 & \cdots & v_N \end{array}\right]$ *are orthogonal matrices with the property that both* $Q^T A Q = H$ *and* $V^T A V = G$ *are upper Hessenberg. Let* k *denote the smallest positive integer for which* $H[k+1,k] = 0$ *with the convention that* $k = N$ *if* H *is unreduced. If* $v_1 = q_1$ *then* $v_i = \pm q_i$ *and* $|H[i,i-1]| = |G[i,i-1]|$ *for* $i = 2, \ldots, k$. *Moreover, if* $k < N$ *then* $G[k+1,k] = 0$.

Proof. See [44], pp. 223–224. □

Note that $v_1 = q_1$ can be replaced with $v_1 = \pm q_1$ without any problem.

In the iteration (5.17), the first column of Q is equal to the first column of M up to a scalar factor. Hence $Q^T M[1]$ (where $M[1]$ denotes the first column of M) is a multiple of the first unit vector e_1.

Lemma 5.6 *Suppose* $QR = M$ *is the QR-decomposition of* M *and* Q_0 *is an orthogonal* $N \times N$*-matrix satisfying* $Q_0^T M[1] = k e_1$, *then* $Q_0[1] = \pm Q[1]$.

Proof. $Q_0^T M[1] = k e_1 \Leftrightarrow M[1] = k Q_0[1]$. The first columns of Q and Q_0 are both a multiple of the first column of M. Hence $Q[1] = l Q_0[1]$. Since $\|Q_0[1]\|_2 = \|Q[1]\|_2 = 1$, $l = \pm 1$ and $Q[1] = \pm Q_0[1]$. □

Theorem 5.5 and lemma 5.6 are used to implement QR step (5.17) efficiently as follows.

1. Compute Me_1, the first column of M, where $M = H - \mu I$ for the single-shift QR iteration and $M = H^2 - sH + tI$ for the double-shift QR iteration with two real or two complex conjugate shifts μ_1 and μ_2 and $s = \mu_1 + \mu_2$ and $t = \mu_1\mu_2$.

2. Compute an orthogonal transformation Q_0 such that $Q_0^T(Me_1)$ is a multiple of e_1. For the single-shift QR iteration, a Givens rotation can be used and for the double-shift step where two elements must be zeroed, a Householder reflection is usually used. Note that the first column of Q_0 is also the first column of Q in the QR-decomposition $QR = M$ of M up to a factor ± 1.

3. Compute the orthogonal transformation Q_1 such that $Q_1^T Q_0^T H Q_0 Q_1$ is upper Hessenberg and the first column of Q, Q_0 and $Q_0 Q_1$ are the same up to a factor ± 1 for Q. According to the implicit Q theorem (theorem 5.5) $Q_1^T Q_0^T H Q_0 Q_1$ and the matrix H obtained by the QR step (5.17) are essentially the same.

Of course this procedure assumes that H is an unreduced Hessenberg matrix. Otherwise we have to apply the procedure on an unreduced subblock. At each step the shifts are determined from the lower right entry (single-shift) or lower right 2×2-block (double-shift) of the upper Hessenberg matrix H. It is also possible to compute the Schur vectors by accumulating all transformations applied to the original matrix G to reduce the matrix to upper Hessenberg form and then to quasi-upper triangular form.

5.3 The periodic QR algorithm

We will only consider the case (5.6) with all matrices G_i real and of full rank.

The periodic Schur decomposition is a generalization of the QR algorithm. It will compute the orthogonal matrices Q_i, $i = 0, \ldots, m-1$, upper triangular matrices $\hat{G}_i$, $i = 1, \ldots, m-1$ and the quasi-upper triangular matrix $\hat{G}_m$ from corollary 5.2. Note that the cyclic transformation

$$G_i \leftarrow Q_i^T G_i Q_{i-1}, \quad Q_m := Q_0, \quad i = 1, \ldots, m$$

with the set of orthogonal matrices Q_i preserves the eigenvalues and Jordan structure of the product $G_m \ldots G_1$. The periodic QR algorithm will construct a series of simple cyclic transformations of the matrices G_i that reduce the matrices G_i to the periodic Schur form. In the first phase of the algorithm, the matrices G_i, $i = 1, \ldots, m-1$, are reduced to the upper triangular matrices H_i, and G_m is reduced to the upper Hessenberg matrix H_m. The product $H = H_m \ldots H_1$ is then also an upper Hessenberg matrix. Note that in our implementation, H_m is the upper Hessenberg matrix. In [11], G_1 is reduced to upper Hessenberg form instead. Our code was developed before we knew of the work of Bojanczyk, Golub and Van Dooren and this explains the differences. In the second phase, double shift QR steps are used to reduce H_m to a quasi-upper triangular matrix. The double-shift QR steps are applied to the product of the matrices without explicitly constructing the product. After every step, the structure of the matrices $H_1, \ldots, H_m$ is restored. H_m will essentially converge to a quasi-upper triangular matrix. Note that the QR steps must be applied to an unreduced part of the Hessenberg matrix, i.e., a block $H[k{:}l, k{:}l]$ such that the subdiagonal contains no zeros. In [11] two possible scenarios are given that lead to a zero on the subdiagonal: a zero in the subdiagonal of H_m or a zero on the diagonal of one of the upper triangular matrices H_i, $i = 1, \ldots, m-1$. We can exclude the latter case since we assume that all matrices G_i and hence H_i have full rank. The algorithm starts with $k = 1$ and $l = N$. If one eigenvalue at position (l, l) or a 2×2-block at position $(l-1{:}l, l-1{:}l)$ with two real eigenvalues or a pair of complex conjugate eigenvalues is isolated, l is decreased with the number of eigenvalues that have been isolated. Furthermore, at each QR step the algorithm determines the smallest possible k that leads to an unreduced block $H[k{:}l, k{:}l]$. We will extend the Newton–Picard method to multiple shooting in the next chapter, and we need to have the eigenvalues ordered according to decreasing modulus. As soon as one or two eigenvalues have been isolated, we move them to their correct place in the periodic Schur decomposition using the "bubble sort" algorithm. This requires four different exchange operations: the exchange of two real eigenvalues, the exchange of two pairs of complex conjugate eigenvalues, the exchange of a real eigenvalue and a pair of complex conjugate eigenvalues where the real eigenvalue has to move down in the Schur decomposition and the exchange of a pair of complex conjugate eigenvalues and a real eigenvalue where the real one has to move up.

All the above operations rely on two elementary orthogonal transformations, the Householder reflection and the Givens rotation. Both transformations were already mentioned in the previous section. We now briefly recall these transformations for reader's convenience. A detailed discussion of the properties and the stable implementation of these transformation can be found in [44].

- The Householder reflection is an orthogonal transformation that can be used to put several elements of a vector at once to zero. Suppose we have the vector
$$x = \begin{bmatrix} 0 & \cdots & 0 & x_k & x_{k+1} & \cdots & x_l & 0 & \cdots & 0 \end{bmatrix}^T$$
and we wish to construct an orthogonal transformation P such that $Px = \|x\|_2 \, e_k$. This goal can be accomplished by the transformation
$$P = I - 2\frac{vv^T}{v^Tv}$$
with
$$v = \begin{bmatrix} 0 & \cdots & 0 & x_k \pm \|x\|_2 & x_{k+1} & \cdots & x_l & 0 & \cdots & 0 \end{bmatrix}^T.$$
The approach can be extended to introduce zeros at prescribed positions of x, changing only one other entry of x.

- The Givens rotation introduces only one zero at a time in a vector. Suppose we have a vector $x = \begin{bmatrix} x_1 & \cdots & x_k \end{bmatrix}^T$ and we wish to construct a transformation $J(i,j)$ such that $(J(i,j)x)[j] = 0$ and $(J(i,j)x)[k] = x[k] \; \forall k \neq i,j$. This can be done by using the matrix
$$J(i,j) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & \ddots & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix},$$
i.e., a c on positions (i,i) and (j,j), s on position (i,j), $-s$ on position (j,i), ones on the other diagonal entries and zeros elsewhere, with
$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad s = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}.$$
The Householder reflection can also be used to introduce one zero and is computationally less expensive. However, results obtained with a Givens rotation are usually slightly better. In practice, Householder reflections are most often used when many zeros can be introduced at once, and Givens rotations if a more selective introduction of zeros is desired.

We will now explain each step of the periodic QR algorithm in more detail.

5.3.1 The reduction to periodic upper Hessenberg form

The reduction to periodic Hessenberg form is done using Householder reflections. Suppose we have three 5×5-matrices and suppose we have brought the first two columns in the desired form. We have

$$\begin{array}{c} H_3 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \end{array} \end{array} \quad \begin{array}{c} H_2 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \end{array} \end{array} \quad \begin{array}{c} H_1 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \end{array} \end{array} .$$

Let $P^{[i]}_{j,k:l}$ denote the Householder reflection that, if applied to the rows of H_i, changes row j and introduces zeros on rows k to l in column j. We first apply the Householder reflection $P^{[1]}_{3,4:5}$ to the rows of H_1 and the columns of H_2. This does not alter the structure of H_2. We get

$$\begin{array}{c} H_3 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \end{array} \end{array} \quad \begin{array}{c} H_2 \\ \begin{array}{ccccc} \bullet & \bullet & * & * & * \\ & \bullet & * & * & * \\ & & * & * & * \\ & & * & * & * \\ & & * & * & * \end{array} \end{array} \quad \begin{array}{c} H_1 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & * & * & * \\ & & 0 & * & * \\ & & 0 & * & * \end{array} \end{array}$$

where the stars denote values that have changed. Then a second reflection $P^{[2]}_{3,4:5}$ is constructed that if applied to the rows of H_2 zeros the $(4,3)$ and $(5,3)$ elements and leaves the first two rows unchanged. This transformation is applied to the rows of H_2 and the columns of H_3 and reduces the matrices to

$$\begin{array}{c} H_3 \\ \begin{array}{ccccc} \bullet & \bullet & * & * & * \\ \bullet & \bullet & * & * & * \\ & \bullet & * & * & * \\ & & * & * & * \\ & & * & * & * \end{array} \end{array} \quad \begin{array}{c} H_2 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & * & * & * \\ & & 0 & * & * \\ & & 0 & * & * \end{array} \end{array} \quad \begin{array}{c} H_1 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \\ & & & \bullet & \bullet \end{array} \end{array} .$$

Next, we construct a Householder transformation $P^{[3]}_{4,5}$ that puts a zero at the $(5,3)$ position of H_3 and leaves its first three rows unchanged. This transformation is applied to the rows of H_3 and the columns of H_1. Note only the last two columns of H_1 change and the structure is left intact. After this transformation we get

$$\begin{array}{c} H_3 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & * & * & * \\ & & 0 & * & * \end{array} \end{array} \quad \begin{array}{c} H_2 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \\ & & & \bullet & \bullet \end{array} \end{array} \quad \begin{array}{c} H_1 \\ \begin{array}{ccccc} \bullet & \bullet & \bullet & * & * \\ & \bullet & \bullet & * & * \\ & & \bullet & * & * \\ & & & * & * \\ & & & * & * \end{array} \end{array} .$$

and we see that now the first three columns are in their final shape. Hence we obtain the following algorithm

Algorithm 5.1 *Reduction to periodic upper Hessenberg form.*

```
Input:
    $G_1, \dots, G_m$
Output:
    $H_1, \dots, H_m$ and orthogonal matrices $Q_0, \dots, Q_{m-1}$ such that $Q_i^T G_i Q_{i-1} = H_i$
    ($Q_m := Q_0$).
begin
    for $i = 1$ to $m$ do
        $H_i \leftarrow G_i$
        $Q_{i-1} \leftarrow I_N$
    endfor
    for $i = 1$ to $N - 1$ do
        for $j = 1$ to $m - 1$ do
            Construct the Householder reflection $P$ such that $PH_j$ has zeros at
            position $(i + 1, i)$ to $(N, i)$ and the rows $1, \dots, i - 1$ are unchanged.
            $H_j \leftarrow PH_j$
            $H_{j+1} \leftarrow H_{j+1}P^T$
            $Q_j \leftarrow Q_jP^T$
        endfor
        if $i < N - 1$ then
            Construct the Householder reflection $P$ such that $PH_m$ has zeros at
            position $(i + 2, i)$ to $(N, i)$ and the rows $1, \dots, i$ are unchanged.
            $H_m \leftarrow PH_m$
            $H_1 \leftarrow H_1P^T$
            $Q_0 \leftarrow Q_0P^T$
        endif
    endfor
end
```

This algorithm requires $O(mN^3)$ operations.

5.3.2 The single-shift QR step

The single-shift QR step will be used in the exchange operations that move down a real eigenvalue along the diagonal of the product Hessenberg matrix $H_m \cdots H_1$. The shift is determined from the known eigenvalue (or the approximation to it) or from the lower right element of the product Hessenberg matrix.

To apply a single-shift QR step to the block $[k{:}l, k{:}l]$, we first need to compute the elements $H[k, k]$ and $H[k + 1, k]$ of $H = H_m \cdots H_1$. This is easy:

$$\begin{aligned} r &= H_{m-1}[k, k] \dots H_1[k, k] \\ H[k, k] &= H_m[k, k]\, r \\ H[k + 1, k] &= H_m[k + 1, k]\, r. \end{aligned}$$

These elements can be computed without any problem as long as no overflow or underflow occurs. Then we construct a Givens rotation J such that

$$J \begin{bmatrix} H[k,k]-\mu \\ H[k+1,k] \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

where μ is the shift. This transformation is applied to the rows k and $k+1$ of H_m and the columns k and $k+1$ of H_1 and introduces a nonzero element at position $(k+1,k)$ of H_1. We can shift this nonzero element to H_m by a sequence of Givens rotations. First we construct a Givens rotation $J^{[1]}(k,k+1)$ that introduces a zero at the $(k+1,k)$ position of H_1 and apply that transformation to the rows of H_1 and the columns of H_2 (i.e., $H_1 \leftarrow J^{[1]}(k+1,k)H_1$ and $H_2 \leftarrow H_2 J^{[1]}(k+1,k)^T$). This transformation zeroes out the nonzero subdiagonal element in H_1 and introduces a nonzero element at the $(k+1,k)$ position of H_2. This element is shifted to H_m using another $m-2$ Givens rotations. In total, m Givens rotations are required: the first one performs the single-shift QR step and the other $m-1$ transformations are needed to restore the structure.

5.3.3 The double-shift QR step

The double-shift QR step is used as the basic tool in order to compute the eigenvalues and it is also used during the reordering of eigenvalues when a pair of complex conjugate eigenvalues is moved down along the diagonal of the product Hessenberg matrix. The shifts are determined from the eigenvalues or from the lower right part of the product Hessenberg matrix. In the latter case, we need to construct the $[l-1{:}l, l-1{:}l]$ block of H. To determine the first column of the transformation Q in (5.16) we need to compute the $[k{:}k+2, k{:}k+1]$ block. Both blocks can be constructed by first calculating the $[k{:}k+1, k{:}k+1]$ and $[l-2{:}l, l-2{:}l]$ blocks of $H_{m-1}\cdots H_1$ and then computing

$$\begin{array}{l} \text{(a) } H[k{:}k+2, k{:}k+1] = H_m[k{:}k+2, k{:}k+1]\,(H_{m-1}\cdots H_1)\,[k{:}k+1, k{:}k+1] \\ \text{(b) } H[l-1{:}l, l-1{:}l] = H_m[l-1{:}l, l-2{:}l]\,(H_{m-1}\cdots H_1)\,[l-2{:}l, l-1{:}l]. \end{array} \tag{5.18}$$

(5.18b) can be computed without too much loss of accuracy if the moduli of its eigenvalues do not differ by too many orders of magnitude. Since these eigenvalues are estimates for eigenvalues of G, a large difference in the moduli will usually indicate a large difference in the moduli of the corresponding eigenvalues of G and QR steps without shift can be used to quickly separate the matrix H into smaller blocks. We have not yet implemented such a feature since we noticed no problems in our tests, even with eigenvalues with extreme differences in magnitude.

The $[k{:}k+2, k]$-block of $M = H^2 - sH + tI$ and the values $s = \lambda_1 + \lambda_2$ and $t = \lambda_1\lambda_2$ are not computed in the way presented in algorithm 7.5-1 in [44]. In some cases, these formulas can lead to a non-convergent iteration because of catastrophic cancelation, as is shown in [92]. Instead we used the more stable approach suggested in [92] and implemented in the EISPACK routine HQR2 and compute

$$\begin{aligned} r_1 &= (H[l,l]-H[k,k])\,(H[l-1,l-1]-H[k,k]) \\ &\quad -H[l,l-1]\,H[l-1,l]/H[k+1,k] + H[k,k+1] \\ r_2 &= H[k+1,k+1]-H[k,k]-(H[l,l]-H[k,k])-(H[l-1,l-1]-H[k,k]) \\ r_3 &= H[k+2,k+1]. \end{aligned}$$

Note that

$$M[k{:}k+2,k] = H[k+1,k]\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}.$$

However, the factor $H[k+1,k]$ has no influence on the transformation that is used to zero out the $(k+1,k)$ and $(k+2,k)$ elements of M. Next, a Householder reflection is constructed such that

$$P\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} * \\ 0 \\ 0 \end{bmatrix} \tag{5.19}$$

and this transformation is applied to rows k to $k+2$ of H_m and columns k to $k+2$ of H_1. This introduces three nonzero elements in H_1 at the positions $(k+1,k)$, $(k+2,k)$ and $(k+2,k+1)$. Afterwards, we have to restore the structure of the matrices H_i. This is considerably more complex than for a single-shift QR step. Several sequences of Householder reflections are needed. The procedure is very similar to the procedure used to construct the periodic Hessenberg form in section 5.3.1. We will demonstrate the procedure for three 4×4-matrices. After the application of the transformation (5.19) we have

$$\begin{array}{cccccccccccccc}
\multicolumn{4}{c}{H_3} & & \multicolumn{4}{c}{H_2} & & \multicolumn{4}{c}{H_1} \\
\ast & \ast & \ast & \ast & & \bullet & \bullet & \bullet & \bullet & & \ast & \ast & \ast & \bullet \\
\ast & \ast & \ast & \ast & & & \bullet & \bullet & \bullet & & \ast & \ast & \ast & \bullet \\
\ast & \ast & \ast & \ast & & & & \bullet & \bullet & & \ast & \ast & \ast & \bullet \\
 & & \bullet & \bullet & & & & & \bullet & & & & & \bullet
\end{array}.$$

First, a Householder reflection $P^{[1]}_{1,2:3}$ is constructed to zero the $(2,1)$ and $(3,1)$ elements of H_1 and the transformation is applied to the rows of H_1 and the columns of H_2 resulting in

$$\begin{array}{cccccccccccccc}
\multicolumn{4}{c}{H_3} & & \multicolumn{4}{c}{H_2} & & \multicolumn{4}{c}{H_1} \\
\bullet & \bullet & \bullet & \bullet & & \ast & \ast & \ast & \bullet & & \ast & \ast & \ast & \ast \\
\bullet & \bullet & \bullet & \bullet & & \ast & \ast & \ast & \bullet & & 0 & \ast & \ast & \ast \\
\bullet & \bullet & \bullet & \bullet & & \ast & \ast & \ast & \bullet & & 0 & \ast & \ast & \ast \\
 & & \bullet & \bullet & & & & & \bullet & & & & & \bullet
\end{array}.$$

Then the $(2,1)$ and $(3,1)$ elements of H_2 are zeroed using the Householder reflection $P^{[2]}_{1,2:3}$ applied to the rows of H_2 and the columns of H_3. This results in

$$\begin{array}{cccccccccccccc}
\multicolumn{4}{c}{H_3} & & \multicolumn{4}{c}{H_2} & & \multicolumn{4}{c}{H_1} \\
\ast & \ast & \ast & \bullet & & \ast & \ast & \ast & \ast & & \bullet & \bullet & \bullet & \bullet \\
\ast & \ast & \ast & \bullet & & 0 & \ast & \ast & \ast & & & \bullet & \bullet & \bullet \\
\ast & \ast & \ast & \bullet & & 0 & \ast & \ast & \ast & & & \bullet & \bullet & \bullet \\
\ast & \ast & \ast & \bullet & & & & & \bullet & & & & & \bullet
\end{array}.$$

Then the $(3,1)$ and $(4,1)$ elements of H_3 are zeroed using a Householder transformation $P^{[3]}_{2,3:4}$. This transformation is applied to the rows of H_3 and the columns of H_1 and does not affect the first column of H_1. We end up with the matrix structure

$$\begin{array}{cccc} & H_3 & & \\ \bullet & \bullet & \bullet & \bullet \\ \ast & \ast & \ast & \ast \\ 0 & \ast & \ast & \ast \\ 0 & \ast & \ast & \ast \end{array} \quad \begin{array}{cccc} & H_2 & & \\ \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & & \bullet \end{array} \quad \begin{array}{cccc} & H_1 & & \\ \bullet & \ast & \ast & \ast \\ & \ast & \ast & \ast \\ & \ast & \ast & \ast \\ & \ast & \ast & \ast \end{array} .$$

In the next phase, we first zero the $(3, 2)$ and $(4, 2)$ elements of H_1 introducing 2 new nonzero elements in H_2, then zeros the corresponding elements in H_2 and finally zero the $(4, 2)$-element in H_3. This results in the structure

$$\begin{array}{cccc} & H_3 & & \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \end{array} \quad \begin{array}{cccc} & H_2 & & \\ \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & & \bullet & \bullet \end{array} \quad \begin{array}{cccc} & H_1 & & \\ \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & & \bullet & \bullet \end{array} .$$

In the last phase, we zero the $(4, 3)$ elements of H_1 and H_2. This can be done without destroying the structure of H_3 by using transformations that only involve the last two rows/columns of the matrices.

The double-shift QR step sometimes isolates a 2×2-block that corresponds to two real eigenvalues. In this case, we need to construct a transformation that reduces the 2×2-block $H_m[l-1{:}l, l-1{:}l]$ to an upper triangular block. The product Hessenberg matrix has a 2×2-block

$$\tilde{H} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix}. \tag{5.20}$$

The eigenvalues are

$$\mu_{1,2} = \frac{h_{1,1} + h_{2,2}}{2} \pm \frac{1}{2}\sqrt{(h_{1,1} - h_{2,2})^2 + 4h_{1,2}h_{2,1}} \tag{5.21}$$

and the corresponding eigenvectors

$$v_{1,2} = \begin{bmatrix} (h_{1,1} - h_{2,2}) \pm \sqrt{(h_{1,1} - h_{2,2})^2 + 4h_{1,2}h_{2,1}} \\ 2h_{2,1} \end{bmatrix}.$$

We use $(h_{1,1} - h_{2,2})^2 + 4h_{1,2}h_{2,1}$ as in the EISPACK routine HQR2 and not the equivalent formula $(h_{1,1} + h_{2,2})^2 - 4(h_{1,1}h_{2,2} - h_{1,2}h_{2,1})$. We do not derive the transformation to zero the element $h_{2,1}$ in (5.20) from the eigenvectors. Instead we use a Givens rotation

$$J = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

derived from the rows of $\tilde{H} - \mu I$ where μ is the largest of the two eigenvalues μ_1 and μ_2 (5.21). c and s are determined such that

$$\left[\tilde{H} - \mu I\right] \begin{bmatrix} c \\ s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.22}$$

and

$$c^2 + s^2 = 1. \tag{5.23}$$

Since μ is an eigenvalue of $\tilde{H}$, a nonzero vector $[c \quad s]^T$ that satisfies (5.22) is an eigenvector of $\tilde{H}$, and the additional normalization condition (5.23) can also be satisfied. The transformation is determined from the row of $\tilde{H} - \mu I$ with the largest diagonal element. Applying the transformation J to rows $l-1$ and l of H_m and columns $l-1$ and l of H_1 introduces a nonzero element at position $(l, l-1)$ in H_1. This nonzero element can be removed easily using a sequence of $m-1$ Givens rotations. After that sequence, the $(l, l-1)$ element of H_m will be (approximately) zero.

5.3.4 Exchange of Schur vectors

Let us first consider the exchange of two real eigenvalues next to each other in the periodic Schur decomposition at position j and $j+1$. We first compute the 2×2 upper triangular block

$$H = \begin{bmatrix} \mu_1 & x \\ 0 & \mu_2 \end{bmatrix}$$

of the product Hessenberg matrix corresponding to the two eigenvalues. Next we compute a Givens rotation J such that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \mu_1 & x \\ 0 & \mu_2 \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} \mu_2 & y \\ 0 & \mu_1 \end{bmatrix}.$$

This coefficients c and s are computed from

$$\begin{aligned} &s_1 = \frac{\mu_2 - \mu_1}{x}, \ \beta = \sqrt{1 + s_1^2}, \\ &c = 1/\beta, \ s = s_1/\beta. \end{aligned}$$

This transformation is applied to the rows j and $j+1$ of H_m and the columns j and $j+1$ of H_1. This introduces nonzero elements at position $(j+1, j)$ in H_1 and H_m. The nonzero subdiagonal element introduced in H_1 is shifted to H_2 using a Givens rotation and further down to H_m. After $m-1$ Givens rotations, the $(j+1, j)$-element of H_m should also be zero because of the particular choice of the first transformation. For sake of robustness, we check the size of this element in our code and further refine this element using single-shift QR steps. μ_1 is used as the shift.

In the other three cases we proceed in a slightly different way. We first permute the columns of H_1 and the rows of H_m corresponding to the eigenvalues that need to be exchanged, then restore the periodic Hessenberg structure and finally use single- or double-shift QR iterations with the eigenvalue(s) as the shift(s).

To exchange a real eigenvalue at position j with a complex eigenvalue pair at position $j+1$ and $j+2$, we apply the cyclic permutation

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

to rows j to $j+2$ of H_m and columns j to $j+2$ of H_1(i.e., postmultiplying with P^T). After restoring the Hessenberg structure, we use single-shift QR iterations using the real eigenvalue as the shift to zero the $(j+2, j+1)$-element of H_m.

To exchange a pair of complex conjugate eigenvalues at positions j and $j+1$ with a real eigenvalue at position $j+2$, we use the permutation matrix

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

and apply double-shift QR iterations using the complex eigenvalue pair for the shifts. Since we wish to use the complex eigenvalues themselves as the shifts and not the values obtained from the lower right entries in the Hessenberg matrix, we have to use the formulas from algorithm 7.5-1 in [44] to compute the $[j{:}j+2, j]$-block of $H^2 - sH + tI$. The $(j+1, j)$ element of H_m decreases a lot in the first few iterations, even in the event of catastrophic cancellation in algorithm 7.5-1 in [44]. To make sure that the algorithm converges, our code switches to the strategy explained in section 5.3.3 after five iterations.

To exchange a complex eigenvalue pair at position j and $j+1$ with one at position $j+2$ and $j+3$, we apply the permutation

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

to rows j to $j+3$ of H_m and columns j to $j+3$ of H_1 and use double-shift QR steps using the eigenvalues from the $[j{:}j+1, j{:}j+1]$-block of the original product Hessenberg matrix for the shifts in the same way as in the previous case.

Note that if we are able to compute an eigenvector for the real eigenvalue or an eigenvector for one of the complex eigenvalues of the 3×3 or 4×4 subblock of $H = H_m \cdots H_1$ corresponding to the Schur vector(s) that need to be moved up in the decomposition, we can use this vector to construct a transformation matrix that reorders the eigenvalues in one step. If the eigenvector that is used is not accurate enough, one should apply one or more single- or double-shift QR steps to refine the result. Consider for instance the case of a real eigenvalue at position j and a complex eigenvalue pair at position $j+1$ and $j+2$. First we construct the $[j{:}j+2, j{:}j+2]$ block of H. Let us denote this block with $\tilde{H}$. Next we compute an eigenvector $v + iw$ for one of the complex eigenvalues of $\tilde{H}$ and we build an orthogonal matrix P such that the first two columns of P lie in $\mathrm{Span}\{v, w\}$. Then the product $P^T \tilde{H} P$ will be a quasi-upper triangular matrix with a 2×2-block in the upper left corresponding to the pair of complex conjugate eigenvalues and a 1×1-block in the lower right. Next we compute

$$\begin{aligned} H_m[j{:}j+2, :] &\leftarrow P^T H_m[j{:}j+2, :], \\ H_1[:, j{:}j+2] &\leftarrow H_1[:, j{:}j+2]\, P, \\ Q_1[:, j{:}j+2] &\leftarrow Q_1[:, j{:}j+2]\, P. \end{aligned} \tag{5.24}$$

Now the $[j{:}j+2, j{:}j+2]$-block of the product $H_m \cdots H_1$ is the matrix $P^T \tilde{H} P$ and the product $H_m \cdots H_1$ is still a quasi-upper triangular matrix and hence a Hessenberg matrix. However, $H_m \cdots H_1$ is not in periodic Hessenberg form. We have to restore the periodic Hessenberg structure. If the transformation matrix would be exact, zeros will appear at position $(j+2, j)$ and $(j+2, j+1)$ of H_m. This can be seen as follows. Consider the case of three blocks. We will only monitor the $[j{:}j+2, j{:}j+2]$-blocks. After applying the transformations (5.24), the structure of the blocks and the product $H_3H_2H_1$ is as follows:

$$\begin{array}{ccc} \multicolumn{3}{c}{H_3} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_2} \\ \bullet & \bullet & \bullet \\ & \bullet & \bullet \\ & & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_3H_2H_1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ & & \bullet \end{array} .$$

We first zero out the $(2,1)$ and $(3,1)$ elements of H_1 by applying the Householder transformation $P^{[1]}_{1,2:3}$ to the rows of H_1 and the columns of H_2. This does not change the product $H_3H_2H_1$. We obtain

$$\begin{array}{ccc} \multicolumn{3}{c}{H_3} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_2} \\ * & * & * \\ * & * & * \\ * & * & * \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_1} \\ * & * & * \\ 0 & * & * \\ 0 & * & * \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_3H_2H_1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ & & \bullet \end{array} .$$

Next we zero out the $(2,1)$ and $(3,1)$ elements of H_2 by applying the Householder transformation $P^{[2]}_{1,2:3}$ to the rows of H_2 and the columns of H_3. Again, these transformations leave the product $H_3H_2H_1$ unchanged. After this step, we get

$$\begin{array}{ccc} \multicolumn{3}{c}{H_3} \\ * & * & * \\ * & * & * \\ 0 & * & * \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_2} \\ * & * & * \\ 0 & * & * \\ 0 & * & * \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_1} \\ \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_3H_2H_1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ & & \bullet \end{array} .$$

The $(3,1)$ block of the product $H_3H_2H_1$ must be zero. Since the $(1,1)$-elements of H_1 and H_2 are nonzero, the $(3,1)$ element of H_3 must be zero after this step. Next we proceed by applying the transformation $P^{[1]}_{2,3}$ to the rows of H_1 and the columns of H_2, and the transformation $P^{[2]}_{2,3}$ to the rows of H_2 and the columns of H_1 and get

$$\begin{array}{ccc} \multicolumn{3}{c}{H_3} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ 0 & & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_2} \\ \bullet & \bullet & \bullet \\ & \bullet & \bullet \\ & & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_1} \\ \bullet & \bullet & \bullet \\ & \bullet & \bullet \\ & & \bullet \end{array} \quad \begin{array}{ccc} \multicolumn{3}{c}{H_3H_2H_1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ & & \bullet \end{array} .$$

These transformations again do not change the product $H_3H_2H_1$. Since the $(2,2)$-elements of H_1 and H_2 are nonzero, the $(3,2)$-element of H_3 must be zero and we have obtained the desired result.

If the transformation P is not exact, we will obtain a small nonzero element at the $(3,1)$-position of H_3 after the first sequence of Householder transformations and we will need to zero out that element by applying a Householder transformation $P^{[3]}_{2,3}$ to the rows of H_3 and the columns of H_1. At the end, the $(3,2)$-element of H_3 is also a small nonzero element and we have to further refine the eigenvalues using one or more single shift QR steps as before. Note that we can get into trouble if it is hard or impossible to compute the eigenvector that is needed in the procedure accurately. This occurs if the real eigenvalue and complex eigenvalue pair are very close to each other. In this case, the eigenvalue and eigenvector can be very ill-conditioned. However, in our application, we are not very interested in reordering such clustered eigenvalues. Note that whatever strategy is used, correct reordering of ill-conditioned eigenvalues is impossible since these eigenvalues may change considerably as they change place in the Schur decomposition. Since we are interested in moving the largest eigenvalues up, we have no problems if the eigenvalues of the 3×3 or 4×4-blocks differ extremely in magnitude, since the largest eigenvalues and corresponding eigenvectors are always quite well preserved when constructing the product of the corresponding subblocks.

5.4 Testcase

We implemented the procedure described in the previous section in a FORTRAN77 routine and interfaced the routine to Matlab version 4. To demonstrate the accuracy of the procedure, we constructed a set of matrices as follows:

- We constructed a block diagonal matrix D of size 10 with 1×1 and 2×2-blocks such that the eigenvalues of D are 100, $(100/\sqrt{2})(1 \pm i)$, 10, 1 ($2\times$), $(1/\sqrt{2})(1 \pm i)$, 0.1 and 0.01 (in a different order on the block diagonal).
- We constructed five random upper block-triangular matrices $R_1, \ldots, R_5$ such that the block diagonal of each of the matrices was D.
- We constructed five random orthogonal matrices $Q_0, \ldots, Q_4$.
- We set
$$\begin{aligned} G_i &= Q_i R_i Q_{i-1}^T,\ Q_1 := Q_0 \\ G &= Q_0 R_5 \cdots R_1 Q_0^T. \end{aligned}$$
Hence, if exact arithmetic were used, $G = G_5 \cdots G_1$ and the eigenvalues of G are the fifth powers of the eigenvalues of D. Note that 1 is a double eigenvalue and is ill-conditioned.

We computed the eigenvalues of G using the Matlab 4 `qr()` routine and computed the periodic Schur decomposition of $G_5 \cdots G_1$ using our FORTRAN77 routine. All tests were done on a IBM RS/6000 workstation with double precision arithmetic. $\varepsilon_{mach} \approx 2.2 \cdot 10^{-16}$. The Matlab version 4 `qr()` routine does not order the eigenvalues. We took the eigenvalues and sorted them by hand such that each eigenvalue corresponds as much as possible to the corresponding exact eigenvalue at the same position. Since 1 is an ill-conditioned eigenvalue and since there are two other eigenvalues with modulus 1, we also had to change the order of the eigenvalues returned by the periodic QR algorithm. The relative errors of the results are given in table 5.2. The results for the three eigenvalues of modulus 10^{10} are comparable for both approaches as we would expect. For the eigenvalue 10^5, the periodic QR algorithm obtains almost machine precision while 4 to 5 digits are lost with the Matlab version 4 `qr()` routine. The double eigenvalue 1 is less accurate because of its ill-conditioning. Still the relative error of the result returned by the periodic QR algorithm is around $5\sqrt{\varepsilon_{mach}}$ while the result returned by Matlab has a relative error around 10^{-3}. Using the QR decomposition of the product, we are unable to compute the

Exact eigenvalue	rel. error per. QR	rel. error Matlab 4 `qr()`
1e+10	5.72e-16	5.72e-16
$(1\text{e}+10/\sqrt{2})(-1 \pm i)$	2.26e-15	2.51e-15
1.e+05	2.91e-16	1.45e-12
1.e+00	6.89e-08	9.51e-04
1.e+00	6.89e-08	9.51e-04
$(1\text{e}+00/\sqrt{2})(-1 \pm i)$	1.06e-14	6.23e-07
1.e-05	3.21e-13	7.75e-01
1.e-10	2.22e-12	7.40e+04

Table 5.2: Results obtained with the periodic QR algorithm and the Matlab version 4 `qr()` routine.

smallest eigenvalue 10^{-10}. Instead, a value around $10^{10}\varepsilon_{mach}$ is returned. The periodic Schur decomposition computes this eigenvalue with a relative error $\approx 10^4\varepsilon_{mach}$. This corresponds to the relative difference of the values on the diagonal of the (quasi) upper-triangular factors of the periodic Schur decomposition. We also used our periodic QR routine to compute the eigenvalues of G directly (i.e., as a product and not as five separate factors) and obtained results comparable to those of the Matlab version 4 `qr()` routine. Of course much more matrices were used to test the routine, but we always observed the same tendency in the results.

5.5 Conclusions

In this section, we showed that traditional approaches for the computation of the Floquet multipliers based on the monodromy matrix can return inaccurate results for the smaller Floquet multipliers if the ratio of the largest to the smallest Floquet multiplier is very large. The solution proposed by Fairgrieve and Jepson in [37, 38] where the monodromy matrix is not constructed explicitly but two matrices that together constitute the monodromy matrix are used, extends the range of computable Floquet multipliers, but can still return inaccurate results. We argued that the periodic Schur decomposition proposed by Bojanczyk, Golub and Van Dooren in [11] is an ideal tool to compute the Floquet multipliers in multiple shooting codes or in AUTO. We also discussed the periodic QR algorithm to compute the variant of the periodic Schur decomposition that occurs in multiple shooting codes and also briefly discussed a strategy to reorder the eigenvalues in the periodic Schur decomposition. These routines will be used in the multiple shooting extensions for the Newton-Picard method proposed in the next chapter. All routines were implemented in FORTRAN77. We also demonstrated the accuracy of the algorithm and the potential of the method.

The algorithm discussed in this chapter is too expensive to be used for high-dimensionalproblems. However, in the next chapter we will show how the algorithm can be combined with subspace iteration and used to compute the dominant Floquet multipliers in a multiple shooting code for large-scale problems.

Our main contributions are the understanding that the periodic Schur decomposition is a very good tool to compute the Floquet multipliers in multiple shooting codes or in AUTO and the implementation in FORTRAN77 and testing of a particular variant suited for multiple shooting codes.

Chapter 6

Multiple shooting methods

In this chapter, we extend our single-shooting based Newton-Picard methods to multiple shooting, a method which we recalled in section 1.4.2. More specifically, we extend the $NPGS$ and CNP variants developed in the algebraic framework of chapter 3. It is hard to extend the view from chapter 2 and interpret the multiple shooting variants as methods that solve an approximation to the true multiple shooting Jacobian matrix exactly. The extension is based on the ideas of reorthogonalization and decoupling used in the multiple shooting method of [4] and on the periodic Schur decomposition presented in the previous chapter. Coupled with the ideas behind our Newton-Picard methods, this leads to a new and efficient multiple shooting algorithm for large-scale problems. Note that in this chapter, we concentrate on the Newton-Picard-specific aspects and we make complete abstraction of all time integration aspects, including the selection of the multiple shooting mesh (1.27).

The overview of this chapter is as follows. We first present a naive method based on forward recursion to solve the linearized multiple shooting system and we argue why this is not a good approach. We proceed with the adaptation of the multiple shooting method of [4] to periodic boundary conditions. Next we extend the splitting used in the single-shooting-based Newton-Picard methods to the multiple shooting case. This will require the use of several related bases, one for each multiple shooting interval. The splitting leads to a set of large but easy to solve Q-systems and a small but hard P-system. The Q-system can be solved in a numerically stable way using techniques based on forward recursion. For the P-system, backward or combined forward and backward recursion is needed. We extend the convergence analysis of section 3.2.2 to the multiple shooting case. To solve the Q-system we adapt the Picard iteration scheme and the GMRES method. We also briefly discuss techniques for solving the P-system. Next we derive a variant of the subspace iteration process with projection and locking based on the periodic Schur decomposition. This algorithm is stable in the presence of large Floquet multipliers and can compute all required bases simultaneously, at a cost that is comparable to the cost of the basis computation in our single-shooting based method. We extend the criterion for the accuracy check of the basis vectors. Finally, we present some test results, and we conclude the chapter with some remarks on avenues for further research.

6.1 Newton-based multiple shooting methods

6.1.1 A naive approach: forward recursion.

In a multiple shooting method for the autonomous system (1.4) we have to solve the nonlinear system (1.28) or, if pseudo-arclength continuation is used, the system

$$\left\{\begin{array}{rcl} \varphi(x_0, \nabla s_1 T, \gamma) - x_1 &=& 0, \\ \varphi(x_1, \nabla s_2 T, \gamma) - x_2 &=& 0, \\ &\vdots& \\ \varphi(x_{m-2}, \nabla s_{m-1} T, \gamma) - x_{m-1} &=& 0, \\ \varphi(x_{m-1}, \nabla s_m T, \gamma) - x_0 &=& 0, \\ s(x_0, x_1, \ldots, x_{m-1}, T, \gamma) &=& 0, \\ n(x_0, x_1, \ldots, x_{m-1}, T, \gamma; \eta) &=& 0. \end{array}\right. \tag{6.1}$$

These nonlinear systems can be solved using Newton's method. For (6.1), this method leads to the repeated solution of linear systems

$$\begin{bmatrix} G_1 & -I & & & b_{T,1} & b_{\gamma,1} \\ & G_2 & -I & & b_{T,2} & b_{\gamma,2} \\ & & \ddots & \ddots & \cdots & \cdots \\ -I & & & G_m & b_{T,m} & b_{\gamma,m} \\ c_{s,1}^T & c_{s,2}^T & \cdots & c_{s,m}^T & d_{s,T} & d_{s,\gamma} \\ c_{n,1}^T & c_{n,2}^T & \cdots & c_{n,m}^T & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_{m-1} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \\ s \\ n \end{bmatrix}, \tag{6.2}$$

where

$$r_i(x_{i-1}, x_i, T, \gamma) = \varphi(x_{i-1}, \nabla s_i T, \gamma) - x_i$$

and

$$\begin{array}{rclcl} G_i & & & = & \dfrac{\partial\varphi(x_{i-1}, \nabla s_i T, \gamma)}{\partial x_{i-1}} = \dfrac{\partial r_i(x_{i-1}, x_i, \nabla s_i T, \gamma)}{\partial x_{i-1}}, \\ B_i & = & \begin{bmatrix} b_{T,i} & b_{\gamma,i} \end{bmatrix} & = & \dfrac{\partial r_i(x_{i-1}, x_i, \nabla s_i T, \gamma)}{\partial (T,\gamma)}, \\ C_i & = & \begin{bmatrix} c_{s,i}^T \\ c_{n,i}^T \end{bmatrix} & = & \dfrac{\partial (s,n)(x_0, \cdots, x_{m-1}, T, \gamma)}{\partial x_{i-1}}, \\ D & = & \begin{bmatrix} d_{s,T} & d_{s,\gamma} \\ d_{s,\gamma} & d_{n,\gamma} \end{bmatrix} & = & \dfrac{\partial (s,n)(x_0, \cdots, x_{m-1}, T, \gamma)}{\partial (T,\gamma)}. \end{array}$$

Note

$$b_{T,i} = \nabla s_i f(\varphi(x_{i-1}, \nabla s_i T, \gamma), \gamma).$$

Remark that the matrices G_i are nonsingular (see, e.g., [4, 25]). Some of the quantities in (6.1) and (6.2) are visually represented in figure 6.1. Terms that are located at shooting mesh interval boundaries are indexed according to the number of the mesh point $s_0 < s_1 < \cdots < s_m = s_0$. Quantities that express a transport from one point to another (e.g., G_i) or a difference between states at two mesh points (e.g., $\nabla s_i T$) are indexed according to the number of the interval from 1 to m.

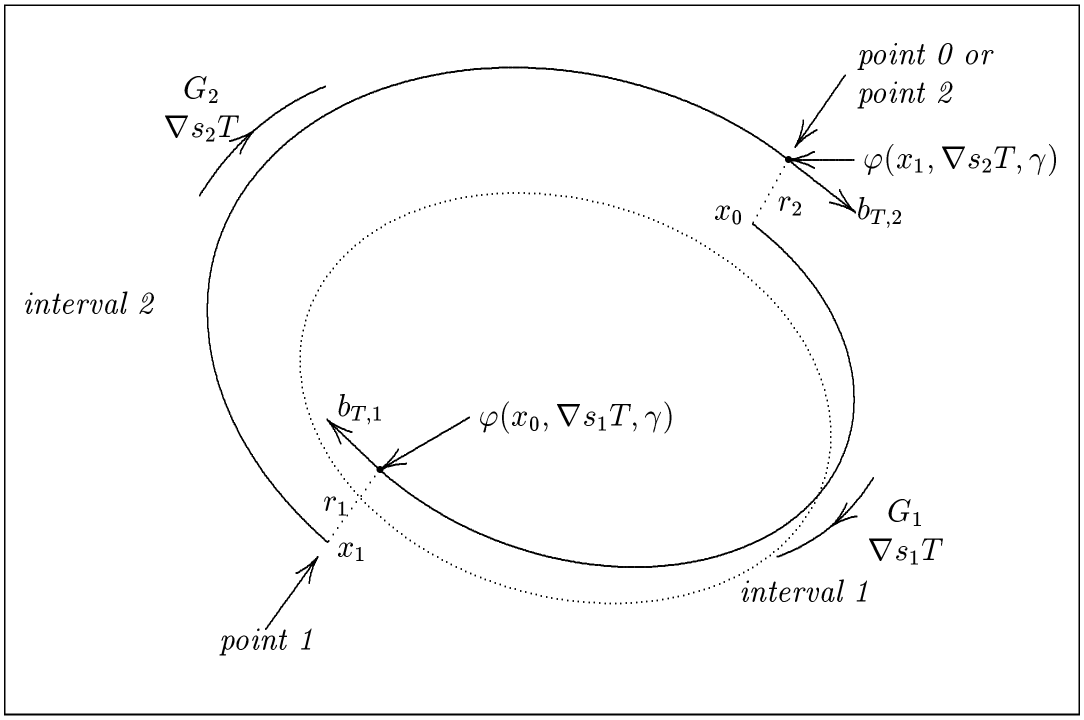


Figure 6.1: Quantities in the multiple shooting system.

A naive way of solving this system is based on using the forward recursion

$$\begin{aligned} \Delta x_i &= G_i \Delta x_{i-1} + b_{T,i}\Delta T + b_{\gamma,i}\Delta\gamma + r_i \\ &= G_i \Delta x_{i-1} + \begin{bmatrix} B_i & r_i \end{bmatrix} \begin{bmatrix} \Delta T \\ \Delta\gamma \\ 1 \end{bmatrix}. \end{aligned} \tag{6.3}$$

Using this recursion, (6.2) is reduced to the $(N+2)\times(N+2)$-dimensional condensed system

$$\begin{bmatrix} \tilde{M} - I & \tilde{B} \\ \tilde{C}^T & \tilde{D} \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \tilde{r} \\ \tilde{s} \\ \tilde{n} \end{bmatrix} \tag{6.4}$$

with

$$\begin{aligned} \tilde{M} &= G_m \cdots G_1, \\ \begin{bmatrix} \tilde{B} & \tilde{r} \end{bmatrix} &= \begin{bmatrix} B_m & r_m \end{bmatrix} + \cdots + G_m \cdots G_2 \begin{bmatrix} B_1 & r_1 \end{bmatrix}, \\ \tilde{C}^T &= C_1^T + C_2^T G_1 + \cdots + C_m^T G_{m-1} \cdots G_1, \\ \begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} & \tilde{s} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} & \tilde{n} \end{bmatrix} &= \begin{bmatrix} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{bmatrix} + C_2^T \begin{bmatrix} B_1 & r_1 \end{bmatrix} + \cdots + \\ & \quad C_m^T \left(\begin{bmatrix} B_{m-1} & r_{m-1} \end{bmatrix} + \cdots + G_{m-1} \cdots G_2 \begin{bmatrix} B_1 & r_1 \end{bmatrix} \right), \\ \tilde{D} &= \begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} \end{bmatrix}. \end{aligned}$$

The matrix elements are easily computed using the following algorithm.

Algorithm 6.1 *Condensation via forward recursion, coefficient computation*

```
    $\tilde{M} = I_N$, $\tilde{B} = 0_{N\times 2}$, $\tilde{C} = 0_{N\times 2}$, $\tilde{D} = 0_{2\times 2}$, $\tilde{r} = 0_{N\times 1}$, $\tilde{s} = 0$ and $\tilde{n} = 0$.
    for $i = 1$ to $m$ do
        $\tilde{C}^T \leftarrow \tilde{C}^T + C_i^T \tilde{M}$
        if $i = 1$ then
            $\begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} & \tilde{s} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} & \tilde{n} \end{bmatrix} \leftarrow \begin{bmatrix} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{bmatrix}$
        else
            $\begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} & \tilde{s} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} & \tilde{n} \end{bmatrix} \leftarrow \begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} & \tilde{s} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} & \tilde{n} \end{bmatrix} + C_i^T \begin{bmatrix} \tilde{B} & \tilde{r} \end{bmatrix}$
        end if
        $\tilde{M} \leftarrow G_i \tilde{M}$
        $\tilde{B} \leftarrow G_i \tilde{B} + B_i$
        $\tilde{r} \leftarrow G_i \tilde{r} + r_i$
    end for
```

After solving (6.4) for Δx_0, ΔT and $\Delta\gamma$, one can compute the other Δx_i's using the recursion (6.3). At the limit cycle, $\tilde{M}$ is the monodromy matrix and

$$\begin{aligned}
\frac{\partial\varphi(x_0,T,\gamma)}{\partial T} &= \frac{\partial\varphi\left(\varphi(x_0, s_{m-1}T,\gamma), \nabla s_m T,\gamma\right)}{\partial T} \\
&= \frac{\partial\varphi(x_{m-1},\nabla s_m T,\gamma)}{\partial T} + \frac{\partial\varphi(x_{m-1},\nabla s_m T,\gamma)}{\partial x_{m-1}}\frac{\partial\varphi(x_0,s_{m-1}T,\gamma)}{\partial T} \\
&= b_{T,m} + G_m \frac{\partial\varphi(x_0,s_{m-1}T,\gamma)}{\partial T} \\
&= \cdots \\
&= b_{T,m} + G_m b_{T,m-1} + \cdots + G_m \cdots G_2 b_{T,1} \\
&= \tilde{b}_T.
\end{aligned}$$

Similarly,

$$\tilde{b}_\gamma = \frac{\partial\varphi(x_0,T,\gamma)}{\partial\gamma}.$$

Hence the system (6.4) has exactly the same structure as the linearized single-shooting system (2.19) and can be solved using the Newton–Picard procedure. Remark also

Lemma 6.1 *The matrix in (6.2) is nonsingular* <u>*iff*</u> *the condensed matrix (6.4) is nonsingular.*

Proof. Let

$$G = \begin{bmatrix} G_1 & -I & & b_{T,1} & b_{\gamma,1} \\ & \ddots & \ddots & \vdots & \vdots \\ -I & & G_m & b_{T,m} & b_{\gamma,m} \\ c_{s,1}^T & \cdots & c_{s,m}^T & d_{s,T} & d_{s,\gamma} \\ c_{n,1}^T & \cdots & c_{n,m}^T & d_{n,T} & d_{n,\gamma} \end{bmatrix}. \tag{6.5}$$

One can apply $m-1$ steps of block Gauss elimination to G, i.e., the matrix G is premultiplied by the matrices E_1, ..., E_{m-1} with

$$E_i = \begin{array}{c c}
\begin{array}{cccccc} & & i & & & \\ & & \downarrow & & & \end{array} & \\
\begin{bmatrix} I & & & & & \\ & \ddots & & & & \\ & & I & & & \\ & & G_{i+1} & I & & \\ & & & & I & \\ & & & & \ddots & \\ & & c_{i+1}^T & & & 1 \end{bmatrix} & \begin{array}{c} \\ \\ \\ \leftarrow i+1, \\ \\ \\ \\ \end{array}
\end{array}$$

i.e., E_i has is an $(mN+1)\times(mN+1)$-matrix with ones on its diagonal, a block G_{i+1} on blockrow $i+1$ in blockcolumn i and a block c_{i+1}^T on the last row in blockcolumn i. One obtains

$$E_{m-1}\cdots E_1 G = \begin{bmatrix} G_1 & -I & & & B_1 \\ G_2G_1 & & -I & & B_2+G_2B_1 \\ \vdots & & & \ddots & \vdots \\ G_{m-1}\cdots G_1 & & & -I & \vdots \\ \tilde{M}-I & & & & \tilde{B} \\ \tilde{C}^T & 0_{N\times 1} & \cdots & \cdots\ 0_{N\times 1} & \tilde{D} \end{bmatrix}.$$

This matrix is easily reordered to

$$\begin{bmatrix} -I & & & & G_1 & B_1 \\ & -I & & & G_2G_1 & B_2+G_2B_1 \\ & & \ddots & & \vdots & \vdots \\ & & & -I & G_{m-1}\cdots G_1 & \vdots \\ & & & & \tilde{M}-I & \tilde{B} \\ 0_{N\times 1} & \cdots & \cdots & 0_{N\times 1} & \tilde{C}^T & \tilde{D} \end{bmatrix}. \tag{6.6}$$

It is easy to see that (6.6) and hence (6.5) are nonsingular <u>iff</u>

$$\begin{bmatrix} \tilde{M}-I & \tilde{B} \\ \tilde{C}^T & \tilde{D} \end{bmatrix}$$

is nonsingular. □

The approach of solving (6.2) by using (6.3) and (6.4) was first suggested in [111] and is known as the "condensation method" [25] or the "compactification algorithm" [4]. It is shown in [4] and [26] that this algorithm is as unstable as single shooting and not very interesting to compute highly unstable limit cycles. However, if the linear system solver functions well enough, we can expect a larger domain of attraction.

6.1.2 A stable algorithm based on reorthogonalization and decoupling

A more stable algorithm to solve (6.2) can be developed based on the ideas of reorthogonalization and decoupling of [4]. The method presented in this section is an adaptation for computing periodic solutions of autonomous systems of the methods in [4]. This algorithm will be the basis for our Newton–Picard procedure for multiple shooting which we will derive in section 6.2.

Suppose we computed the periodic Schur decomposition of $G_m \cdots G_1$ such that the p eigenvalues larger than or equal to one in modulus come first. Note the different meaning of p in this section compared to chapter 2! We then have orthonormal matrices $Q_0, \ldots, Q_{m-1}$, upper triangular matrices $R_1, \ldots, R_{m-1}$ and an upper block triangular matrix R_m with 1×1-blocks corresponding to the real eigenvalues and 2×2-blocks corresponding to pairs of complex conjugate eigenvalues such that

$$\begin{aligned} &Q_i^T G_i Q_{i-1} = R_i, \quad i = 1, \ldots, m-1, \\ &Q_0^T G_m Q_{m-1} = R_m. \end{aligned}$$

Let

$$\begin{bmatrix} \Delta\tilde{x}_{i,p} \\ \Delta\tilde{x}_{i,q} \end{bmatrix} = \Delta\tilde{x}_i = Q_i^T \Delta x_i$$

with $\Delta\tilde{x}_{i,p}$ the first p components of $\Delta\tilde{x}_i$, let

$$\begin{bmatrix} \tilde{r}_{i,p} \\ \tilde{r}_{i,q} \end{bmatrix} = \tilde{r}_i = Q_i^T r_i, \text{ etc.}$$

Furthermore, let

$$R_i = \begin{bmatrix} R_{i,p} & R_{i,pq} \\ 0 & R_{i,q} \end{bmatrix}.$$

Using these transformations, we can transform (6.2) to the system

$$\begin{bmatrix} R_{1,p} & R_{1,pq} & -I_p & & & & \tilde{b}_{T,1,p} & \tilde{b}_{\gamma,1,p} \\ & R_{1,q} & & -I_q & & & \tilde{b}_{T,1,q} & \tilde{b}_{\gamma,1,q} \\ & & \ddots & \ddots & \ddots & & \vdots & \vdots \\ & & & \ddots & & \ddots & \vdots & \vdots \\ -I_p & & & & R_{m,p} & R_{m,pq} & \tilde{b}_{T,m,p} & \tilde{b}_{\gamma,m,p} \\ & -I_q & & & & R_{m,q} & \tilde{b}_{T,m,q} & \tilde{b}_{\gamma,m,q} \\ \tilde{c}_{s,1,p}^T & \tilde{c}_{s,1,q}^T & \cdots & \cdots & \tilde{c}_{s,m,p}^T & \tilde{c}_{s,m,q}^T & d_{s,T} & d_{s,\gamma} \\ \tilde{c}_{n,1,p}^T & \tilde{c}_{n,1,q}^T & \cdots & \cdots & \tilde{c}_{n,m,p}^T & \tilde{c}_{n,m,q}^T & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}_{0,p} \\ \Delta\tilde{x}_{0,q} \\ \vdots \\ \vdots \\ \Delta\tilde{x}_{m-1,p} \\ \Delta\tilde{x}_{m-1,q} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{1,p} \\ \tilde{r}_{1,q} \\ \vdots \\ \vdots \\ \tilde{r}_{m,p} \\ \tilde{r}_{m,q} \\ s \\ n \end{bmatrix}. \tag{6.7}$$

The blocks $R_{i,q}$ correspond to modes that shrink when moving forward in time, the blocks $R_{i,p}$ to modes that shrink when moving backwards. We will now try to compact the system to a system of the form

$$\begin{bmatrix} \hat{R}_p - I_p & \hat{R}_{pq} & \hat{B}_p \\ 0 & \hat{R}_q - I_q & \hat{B}_q \\ \hat{C}_p^T & \hat{C}_q^T & \hat{D} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}_{0,p} \\ \Delta\tilde{x}_{0,q} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \hat{r}_p \\ \hat{r}_q \\ \hat{s} \\ \hat{n} \end{bmatrix} \tag{6.8}$$

and derive relations to compute the other unknowns once $\Delta\tilde{x}_{0,p}$, $\Delta\tilde{x}_{0,q}$, ΔT and $\Delta\gamma$ are known. Therefore, we first build a relationship between the different $\Delta\tilde{x}_{i,q}$'s and $\Delta\tilde{x}_{0,q}$, ΔT and $\Delta\gamma$ using a forward recursion, and then we construct a relationship that relates the different $\Delta\tilde{x}_{i,p}$'s to $\Delta\tilde{x}_{0,p}$, $\Delta\tilde{x}_{0,q}$, ΔT and $\Delta\gamma$ using a backward recursion.

From the even blockrows of (6.7),

$$\Delta\tilde{x}_{i,q} = R_{i,q}\Delta\tilde{x}_{i-1,q} + \tilde{b}_{T,i,q}\Delta T + \tilde{b}_{\gamma,i,q}\Delta\gamma + \tilde{r}_{i,q}, \quad i = 1,\ldots,m, \quad \Delta\tilde{x}_{m,q} := \Delta\tilde{x}_{0,q}, \tag{6.9}$$

we can construct the relation

$$\Delta\tilde{x}_{i,q} = \hat{R}_{q,i}\Delta\tilde{x}_{0,q} + \hat{B}_{q,i}\left[\begin{array}{cc} \Delta T & \Delta\gamma \end{array}\right]^T + \hat{r}_{q,i}, \quad i = 1,\ldots,m.$$

The terms $\hat{R}_{q,i}$, $\hat{B}_{q,i}$ and $\hat{r}_{q,i}$ can be computed using the following algorithm:

Algorithm 6.2 *Forward recursion for building the terms $\hat{R}_q$, $\hat{B}_q$ and $\hat{r}_q$ of the condensed system*

```
    $\hat{R}_{q,0} = I_q$, $\hat{B}_{q,0} = 0_{q\times 2}$, $\hat{r}_{q,0} = 0_{q\times 1}$
    for $i = 1$ to $m$
        $\hat{R}_{q,i} = R_{i,q}\hat{R}_{q,i-1}$
        $\hat{B}_{q,i} = R_{i,q}\hat{B}_{q,i-1} + \tilde{B}_{i,q}$
        $\hat{r}_{q,i} = R_{i,q}\hat{r}_{q,i-1} + \tilde{r}_{i,q}$
    end for
    $\hat{R}_q = \hat{R}_{q,m}$, $\hat{B}_q = \hat{B}_{q,m}$, $\hat{r}_q = \hat{r}_{q,m}$
```

The intermediate results $\hat{R}_{q,i}$, $\hat{B}_{q,i}$ and $\hat{r}_{q,i}$ must be stored because they will be needed to compute the other coefficients in the condensed system (6.8). During the recursion, we can also already compute a matrix $\breve{C}_q$ according to

$$\breve{C}_q^T = \sum_{i=1}^{m} \tilde{C}_{i,q}^T \hat{R}_{q,i-1}$$

and the coefficients

$$\left[\begin{array}{ccc} \breve{d}_{s,T} & \breve{d}_{s,\gamma} & \breve{s} \\ \breve{d}_{n,T} & \breve{d}_{n,\gamma} & \breve{n} \end{array}\right] = \left[\begin{array}{ccc} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{array}\right] + \sum_{i=1}^{m} C_{i,q}^T \left[\begin{array}{cc} \hat{B}_{q,i-1} & \hat{r}_{q,i-1} \end{array}\right].$$

These are not yet the coefficients of the condensed system (6.8). The complete terms $\hat{C}_q^T$ and $\hat{D}$ can only be constructed after the backward recursion step.

To compute the other matrices, we derive a relationship

$$\Delta\tilde{x}_{i-1,p} = \hat{R}_{p,i-1}\Delta\tilde{x}_{0,p} + \hat{R}_{pq,i-1}\Delta\tilde{x}_{0,q} + \hat{B}_{p,i-1}\left[\begin{array}{cc} \Delta T & \Delta\gamma \end{array}\right] + \hat{r}_{p,i-1}, \quad i = m,\ldots,1,$$

using the backward recursion

$$\begin{gathered} \Delta\tilde{x}_{i-1,p} = R_{i,p}^{-1}\left(\Delta\tilde{x}_{i,p} - R_{i,pq}\Delta\tilde{x}_{i-1,q} - \tilde{b}_{T,i,p}\Delta T - \tilde{b}_{\gamma,i,p}\Delta\gamma - \tilde{r}_{i,p}\right), \\ i = m,\ldots,1,\ \Delta\tilde{x}_{m,p} := \Delta\tilde{x}_{0,p}. \end{gathered} \tag{6.10}$$

The coefficients $\hat{R}_{p,}$, $\hat{R}_{pq,i}$, $\hat{B}_{p,i}$ and $\hat{r}_{p,i}$, $i = 0,\ldots,m-1$, can be computed using the following algorithm:

Algorithm 6.3 *Backward recursion for building the terms $\hat{R}_p$, $\hat{R}_{pq}$, $\hat{B}_p$ and $\hat{r}_p$ of the condensed system.*

```
    $\hat{R}_{p,m} = I_p$, $\hat{R}_{pq,m} = 0_{p\times q}$, $\hat{B}_{p,m} = 0_{p\times 2}$ and $\hat{r}_{p,m} = 0_{p\times 1}$.
    for $i = m$ downto 1
        $\hat{R}_{p,i-1} = R_{i,p}^{-1}\hat{R}_{p,i}$
        $\hat{R}_{pq,i-1} = R_{i,p}^{-1}\left(\hat{R}_{pq,i} - R_{i,pq}\hat{R}_{q,i-1}\right)$
        $\hat{B}_{p,i-1} = R_{i,p}^{-1}\left(\hat{B}_{p,i} - \tilde{B}_{i,p} - R_{i,pq}\hat{B}_{q,i-1}\right)$
        $\hat{r}_{p,i-1} = R_{i,p}^{-1}\left(\hat{r}_{p,i} - \tilde{r}_{i,p} - R_{i,pq}\hat{r}_{q,i-1}\right)$
    end for
    $\hat{R}_p = \hat{R}_{p,0}$, $\hat{R}_{pq} = \hat{R}_{pq,0}$, $\hat{B}_p = \hat{B}_{p,0}$, $\hat{r}_p = \hat{r}_{p,0}$
```

Finally, we can build the last two equations of (6.8) as

$$\hat{C}_q^T = \sum_{i=1}^{m} \tilde{C}_{i,q}^T \hat{R}_{q,i-1} + \sum_{i=1}^{m} \tilde{C}_{i,p}^T \hat{R}_{pq,i-1} = \breve{C}_q^T + \sum_{i=1}^{m} \tilde{C}_{i,p}^T \hat{R}_{pq,i-1},$$

$$\hat{C}_p^T = \sum_{i=1}^{m} \tilde{C}_{i,p}^T \hat{R}_{p,i-1}$$

$$\begin{aligned}\begin{bmatrix} \hat{d}_{s,T} & \hat{d}_{s,\gamma} & \hat{s} \\ \hat{d}_{n,T} & \hat{d}_{n,\gamma} & \hat{n} \end{bmatrix} &= \begin{bmatrix} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{bmatrix} + \textstyle\sum_{i=1}^m \tilde{C}_{i,q}^T \begin{bmatrix} \hat{B}_{q,i-1} & \hat{r}_{q,i-1} \end{bmatrix} \\ &\quad + \textstyle\sum_{i=1}^m \tilde{C}_{i,p}^T \begin{bmatrix} \hat{B}_{p,i-1} & \hat{r}_{p,i-1} \end{bmatrix} \\ &= \begin{bmatrix} \breve{d}_{s,T} & \breve{d}_{s,\gamma} & \breve{s} \\ \breve{d}_{n,T} & \breve{d}_{n,\gamma} & \breve{n} \end{bmatrix} + \textstyle\sum_{i=1}^m \tilde{C}_{i,p}^T \begin{bmatrix} \hat{B}_{p,i-1} & \hat{r}_{p,i-1} \end{bmatrix}.\end{aligned}$$

The computation of these coefficients can be done during the backward recursion algorithm 6.3.

Remark that (6.8) is upper block triangular except for the two rows corresponding to the phase- and pseudo-arclength conditions. It can be solved using some variant of Gauss elimination and after solving the system, we can compute the unknowns $\Delta\tilde{x}_{i,q}$ using the forward recursion (6.9) and then $\Delta\tilde{x}_{i,p}$ via the backward recursion (6.10). Finally, we reconstruct $\Delta x_i = Q_i \Delta\tilde{x}_i$. Similarly to the single shooting procedure, we can omit the pseudo-arclength condition and/or the phase condition and solve the underdetermined condensed system using the least squares procedure.

Instead of using the above procedure, (6.7) can also be solved using an algorithm based on the Moore–and–Spence-like approach which we used in our single shooting method with pseudo-arclength continuation. We can first compute $\Delta\tilde{x}_{i,q,r}$, $\Delta\tilde{x}_{i,q,T}$ and $\Delta\tilde{x}_{i,q,\gamma}$

from

$$\begin{bmatrix} R_{1,q} & -I_q & & \\ & \ddots & \ddots & \\ & & R_{m-1,q} & -I_q \\ -I_q & & & R_{m,q} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}_{0,q,r} & \Delta\tilde{x}_{0,q,T} & \Delta\tilde{x}_{0,q,\gamma} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \Delta\tilde{x}_{m-1,q,r} & \Delta\tilde{x}_{m-1,q,T} & \Delta\tilde{x}_{m-1,q,\gamma} \end{bmatrix} = \\ - \begin{bmatrix} \tilde{r}_{1,q} & \tilde{b}_{1,1,q} & \tilde{b}_{2,1,q} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \tilde{r}_{m,q} & \tilde{b}_{1,m,q} & \tilde{b}_{2,m,q} \end{bmatrix} \tag{6.11}$$

using compactification based on forward recursion. The matrix in system (6.11) is nonsingular since $R_{m,q}\cdots R_{1,q}$ has only eigenvalues smaller than one in modulus. Next, $\Delta\tilde{x}_{i,p}$, ΔT and $\Delta\gamma$ are solved from

$$\begin{bmatrix} R_{1,p} & -I_p & & & \check{b}_{T,1,p} & \check{b}_{\gamma,1,p} \\ & \ddots & \ddots & & \vdots & \vdots \\ & & R_{m-1,p} & -I_p & \vdots & \vdots \\ -I_p & & & R_{m,p} & \check{b}_{T,m,p} & \check{b}_{\gamma,m,p} \\ \tilde{c}^T_{s,1,p} & \cdots & \cdots & \tilde{c}^T_{s,m,p} & \check{d}_{s,T} & \check{d}_{s,\gamma} \\ \tilde{c}^T_{n,1,p} & \cdots & \cdots & \tilde{c}^T_{n,m,p} & \check{d}_{n,T} & \check{d}_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}_{0,p} \\ \vdots \\ \vdots \\ \Delta\tilde{x}_{m-1,p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \check{r}_{1,p} \\ \vdots \\ \vdots \\ \check{r}_{m,p} \\ \check{s} \\ \check{n} \end{bmatrix} \tag{6.12}$$

where

$$\begin{bmatrix} \check{b}_{T,i,p} & \check{b}_{\gamma,i,p} & \check{r}_{i,p} \end{bmatrix} = \begin{bmatrix} \tilde{b}_{T,i,p} & \tilde{b}_{\gamma,i,p} & \tilde{r}_{i,p} \end{bmatrix} + R_{i,pq} \begin{bmatrix} \Delta\tilde{x}_{i-1,q,T} & \Delta\tilde{x}_{i-1,q,\gamma} & \Delta\tilde{x}_{i-1,q,r} \end{bmatrix}$$

and

$$\begin{bmatrix} \check{d}_{s,T} & \check{d}_{s,\gamma} & \check{s} \\ \check{d}_{n,T} & \check{d}_{n,\gamma} & \check{n} \end{bmatrix} = \begin{bmatrix} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{bmatrix} + \sum_{i=1}^{m} \tilde{C}^T_{i,q} \begin{bmatrix} \Delta\tilde{x}_{i-1,q,T} & \Delta\tilde{x}_{i-1,q,\gamma} & \Delta\tilde{x}_{i-1,q,r} \end{bmatrix}.$$

This system can be solved using compactification based on backward recursion. Finally, we compute

$$\Delta\tilde{x}_{i,q} = \Delta\tilde{x}_{i,q,r} + \Delta T\,\Delta\tilde{x}_{i,q,T} + \Delta\gamma\,\Delta\tilde{x}_{i,q,\gamma}$$

and retransform the variables according to $\Delta x_i = Q_i \Delta\tilde{x}_i$. We can also choose to omit the pseudo-arclength and/or phase conditions and solve the underdetermined system that is the result of the compactification algorithm applied to (6.12) using a least-squares approach. If we use explicit pseudo-arclength and phase conditions, both approaches are in theory equivalent. This is not the case if we omit the pseudo-arclength or phase conditions. Then the results of both approaches will generally differ.

The Moore–and–Spence-like approach is cheaper than the other approach based on the condensed system (6.8). *The condensed version of (6.11) is already in upper block-triangular form (with* 1×1 *and* 2×2 *blocks on its diagonal) and is easily solved.* Of course we can also use the Moore–and–Spence-like approach to solve (6.8). However, it is much easier to first apply the Moore–and–Spence-like step and then condense the system instead of vice-versa. The recursions are much simpler in the former case. Note however that the computation of the periodic Schur decomposition is usually the most expensive step in both methods.

The above methods are very expensive for large-scale systems. First, we have to compute and store the m $N \times N$-matrices G_i, requiring N IVP solves per matrix and a lot of storage space. We also have to compute the periodic Schur decomposition and the condensed system, which are algorithms of complexity $\mathrm{O}(mN^3)$. In the next section, we will apply the Newton–Picard idea to the above methods to come to a method with reasonable cost for many large problems.

6.2 The Newton–Picard procedure

The splitting used in the algorithm presented in the previous section has many similarities to the splitting which we used in our single shooting based methods. In fact, in the multiple shooting algorithm presented above, we split the system in a part corresponding to unstable modes and the orthogonal complement. All modes corresponding to eigenvalues outside or on the unit circle go in the unstable subspace and are treated using condensation with a backward recursion. In the orthogonal complement, condensation based on a forward recursion is used. In our single shooting based method, we used a circle with radius $\rho < 1$, and also used different techniques for modes outside C_ρ and the orthogonal complement. We will now combine the ideas behind these two methods to derive an efficient multiple shooting technique for the computation of periodic solutions of autonomous large-scale systems.

6.2.1 Splitting the linearized systems

Remark that the monodromy matrix is not determined by the periodic solution alone, but also by the point of the limit cycle at which we evaluate the monodromy matrix. Its eigenvalues are independent of that point, but the eigenvectors vary. The following lemma gives us information about the relationship of the eigenvectors of the monodromy matrix evaluated at the different multiple shooting points.

Lemma 6.2 *Suppose $M_0 = G_m \cdots G_1$ is nonsingular. Let $M_0 X_0 = X_0 \Lambda$ be the Jordan decomposition of M_0. Let $M_j = G_j \cdots G_1 G_m \cdots G_{j+1}$ and $X_j = G_j \cdots G_1 X_0$. Then $M_j X_j = X_j \Lambda$ is the Jordan decomposition of M_j.*

Proof. Since $M_0 = G_m \cdots G_1$ is nonsingular, all factors G_i are nonsingular. Since $MX_0 = X_0\Lambda$, $G_j \cdots G_1 G_m \cdots G_{j+1} G_j \cdots G_1 X_0 = G_j \cdots G_1 X_0 \Lambda$ and $M_j X_j = X_j \Lambda$. X_j is nonsingular since $G_1, \cdots, G_j$ and X_0 are nonsingular, so $M_j X_j = X_j \Lambda$ is the Jordan decomposition of M_j. □

In our applications, all matrices G_i are nonsingular and this lemma applies. The splitting is done as in the single shooting method along a circle with radius $\rho < 1$. From lemma 6.2, we learn that the subspaces will differ for each of the multiple shooting points, but are related. We can make the following assumption in direct analogy to assumption 2.4.

Assumption 6.3 *Let $y^* = (x_0^*, \cdots, x_{m-1}^*, T^*, \gamma^*)$ denote an isolated solution of (6.1), and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M_0(y) = G_m(y) \cdots G_1(y)$ for $y \in \mathcal{B}$ and denote its eigenvalues by μ_i, $i = 1, \cdots, N$. Assume that for all $y \in \mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho = |z| < \rho, \quad 0 < \rho < 1, \tag{6.13}$$

and that no eigenvalue has modulus ρ; i.e., for all $y \in \mathcal{B}$,

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}| \geq \cdots \geq |\mu_N|.$$

Again, eigenvalues may move in or out of C_ρ in actual computations, and our code can deal with that, but as the iteration comes close to convergence, the assumption is observed in practice and so is not unreasonable. Because of lemma 6.2, the assumption will also hold for the matrices $M_i = G_i \cdots G_1 G_m \cdots G_{i+1}$, $i = 0, \cdots, m-1$. Let the columns of the matrix $V_{p,i}$ be an orthonormal basis for the subspace $\mathcal{U}_i$ of $\mathbb{R}^N$ spanned by the (generalized) eigenvectors of M_i, $i = 0, \cdots, m-1$, corresponding to the eigenvalues μ_i, $i = 1, \cdots, p$, and let the columns of $V_{q,i} \in \mathbb{R}^{N \times (N-p)} = \mathbb{R}^{N \times q}$ be an orthonormal basis for $\mathcal{U}_i^\perp$, the orthogonal complement of $\mathcal{U}_i$ in $\mathbb{R}^N$. Notice that the matrices $V_{q,i}$ define bases for high-dimensional spaces. We want to avoid building these bases, but need them for theoretical purposes. One can easily show there exist nonsingular matrices $L_i \in \mathbb{R}^{p \times p}$ such that

$$G_i V_{p,i-1} = V_{p,i} L_i. \tag{6.14}$$

We can construct projectors P_i and Q_i of $\mathbb{R}^N$ onto $\mathcal{U}_i$ and $\mathcal{U}_i^\perp$ respectively as

$$\begin{array}{rcl} P_i & := & V_{p,i} V_{p,i}^T, \\ Q_i & := & V_{q,i} V_{q,i}^T = I - V_{p,i} V_{p,i}^T. \end{array} \tag{6.15}$$

To ease some notations, we use in this section

$$V_{p,m} := V_{p,0}, \; V_{q,m} := V_{q,0}, \; P_m := P_0 \text{ and } Q_m := Q_0.$$

Every $x_i \in \mathbb{R}^N$ can be uniquely decomposed as

$$x_i = V_{p,i}\bar{p}_i + V_{q,i}\bar{q}_i, \quad p_i = V_{p,i}\bar{p}_i = P_i x_i, \quad q_i = V_{q,i}\bar{q}_i = Q_i x_i, \tag{6.16}$$

with $\bar{p}_i \in \mathbb{R}^p$ and $\bar{q}_i \in \mathbb{R}^q = \mathbb{R}^{N-p}$. Substituting (6.16) in (6.2), multiplying the i$^{\text{th}}$ set of N rows with $[V_{q,i} \;\; V_{p,i}]^T$ at the left hand side ($i = 1, \cdots, m-1$), the m^{th} set of N rows with $[V_{q,0} \;\; V_{p,0}]^T$ at the left hand side and the i^{th} set of N columns with $[V_{q,i-1} \;\; V_{p,i-1}]$,

$i = 1, \cdots, m$, one obtains

$$
\left[\begin{array}{ccccccccc}
F_{qq,1} & F_{qp,1} & -I_q & & & & & & V_{q,1}^T B_1 \\
F_{pq,1} & F_{pp,1} & & -I_p & & & & & V_{p,1}^T B_1 \\
 & & \ddots & \ddots & \ddots & & & & \vdots \\
 & & \ddots & \ddots & & \ddots & & & \vdots \\
 & & & & F_{qq,m-1} & F_{qp,m-1} & -I_q & & V_{q,m-1}^T B_{m-1} \\
 & & & & F_{pq,m-1} & F_{pp,m-1} & & -I_q & V_{p,m-1}^T B_{m-1} \\
-I_q & & & & & & F_{qq,m} & F_{qp,m} & V_{q,m}^T B_m \\
 & -I_p & & & & & F_{pq,m} & F_{pp,m} & V_{p,m}^T B_m \\
C_1^T V_{q,0} & C_1^T V_{p,0} & \cdots & \cdots & \cdots & \cdots & C_m^T V_{q,m-1} & C_m^T V_{p,m-1} & D
\end{array}\right]
\left[\begin{array}{c}
\Delta\bar{q}_0 \\ \Delta\bar{p}_0 \\ \vdots \\ \vdots \\ \Delta\bar{q}_{m-1} \\ \Delta\bar{p}_{m-1} \\ \Delta T \\ \Delta\gamma
\end{array}\right]
= -
\left[\begin{array}{c}
V_{q,1}^T r_1 \\ V_{p,1}^T r_1 \\ \vdots \\ \vdots \\ V_{q,m}^T r_m \\ V_{p,m}^T r_m \\ s \\ n
\end{array}\right],
\tag{6.17}
$$

where

$$
\left[\begin{array}{cc} F_{qq,i} & F_{qp,i} \\ F_{pq,i} & F_{pp,i} \end{array}\right] = \left[\begin{array}{cc} V_{q,i}^T G_i V_{q,i-1} & V_{q,i}^T G_i V_{p,i-1} \\ V_{p,i}^T G_i V_{q,i-1} & V_{p,i}^T G_i V_{p,i-1} \end{array}\right], \quad i = 1, \cdots, m, \tag{6.18}
$$

$$
B_i = \left[\begin{array}{cc} b_{T,i} & b_{\gamma,i} \end{array}\right], \; C_i = \left[\begin{array}{cc} c_{s,i} & c_{n,i} \end{array}\right] \text{ and } D = \left[\begin{array}{cc} d_{s,T} & d_{s,\gamma} \\ d_{n,T} & d_{n,\gamma} \end{array}\right].
$$

Let us introduce the following notations:

$$
F_{qq}^o = \left[\begin{array}{ccc} F_{qq,1} & -I_q & \\ & \ddots & \ddots \\ -I_q & & F_{qq,m} \end{array}\right], \; F_{qp}^o = \left[\begin{array}{ccc} F_{qp,1} & & \\ & \ddots & \\ & & F_{qp,m} \end{array}\right],
$$

$$
F_{pq}^o = \left[\begin{array}{ccc} F_{pq,1} & & \\ & \ddots & \\ & & F_{pq,m} \end{array}\right], \; F_{pp}^o = \left[\begin{array}{ccc} F_{pp,1} & -I_p & \\ & \ddots & \ddots \\ -I_p & & F_{pp,m} \end{array}\right],
$$

and

$$
\left\{V_{q,i}^T b_{*,i}\right\} = \left[\begin{array}{c} V_{q,1}^T b_{*,1} \\ \vdots \\ V_{q,m}^T B_{*,m} \end{array}\right],
$$

$$
\left\{c_{*,i}^T V_{q,i-1}\right\} = \left[\begin{array}{ccc} c_{*,1}^T V_{q,0} & \cdots & c_{*,m-1}^T V_{q,m-1} \end{array}\right], \text{ etc.}
$$

$\{v_i\}$ with v_i a column vector is the column vector obtained by putting the m column vectors $v_1, \ldots, v_m$ on top of each other, $\left\{v_i^T\right\}$ with v_i^T a row vector is the row vector obtained by putting $v_1^T, \ldots, v_m^T$ next to each other. Similarly, the notation is extended to sets of two row or column vectors. We can reorder (6.17) to

$$\begin{bmatrix} F_{qq}^o & F_{qp}^o & \{V_{q,i}^T B_i\} \\ F_{pq}^o & F_{pp}^o & \{V_{p,i}^T B_i\} \\ \{C_i^T V_{q,i-1}\} & \{C_i^T V_{p,i-1}\} & D \end{bmatrix} \begin{bmatrix} \{\Delta \bar{q}_{i-1}\} \\ \{\Delta \bar{p}_{i-1}\} \\ \Delta T \\ \Delta \gamma \end{bmatrix} = - \begin{bmatrix} \{V_{q,i}^T r_i\} \\ \{V_{p,i}^T r_i\} \\ s \\ n \end{bmatrix}. \tag{6.19}$$

Now, since $G_i V_{p,i-1} = V_{p,i} L_i$, F_{qp}^o disappears. At the limit cycle, $b_{T,i}$ is the tangent vector to the limit cycle in x_i and an eigenvector of M_i for the trivial Floquet multiplier 1. Since $\rho < 1$, $b_{T,i} \in \mathcal{U}_i$ and thus $V_{q,i}^T b_{T,i} = 0$. This term is small near the limit cycle and is usually neglected in actual computations. Note also that the block F_{qq}^o has precisely the same function as the block $V_q^T M V_q - I_q$ in (3.1), etc. In fact, for $m = 1$, (6.19) reduces to (3.1).

(6.19) is solved again in the spirit of the Moore–and–Spence-like approach. First we compute $\Delta \bar{q}_{i,r}$, $\Delta \bar{q}_{i,T}$ and $\Delta \bar{q}_{i,\gamma}$ from

$$F_{qq}^o \left[\{\Delta \bar{q}_{i-1,r}\} \quad \{\Delta \bar{q}_{i-1,T}\} \quad \{\Delta \bar{q}_{i-1,\gamma}\} \right] = - \left[\{V_{q,i}^T r_i\} \quad \{V_{q,i}^T b_{T,i}\} \quad \{V_{q,i}^T b_{\gamma,i}\} \right]. \tag{6.20}$$

It is possible to show that under assumption 6.3, the matrix F_{qq}^o is nonsingular. The proof is very similar to that of lemma 6.1. Since the terms $V_{q.i}^T b_{T,i}$ are usually small, one can often put $\Delta \bar{q}_{i,T} = 0$. Next we compute the $\Delta \bar{p}_i$'s, ΔT and $\Delta \gamma$ from

$$\begin{bmatrix} F_{pp}^0 & \{V_{p,i}^T B_i + F_{pq,i} V_{q,i-1} \left[\Delta \bar{q}_{i-1,T} \quad \Delta \bar{q}_{i-1,\gamma} \right]\} \\ \{C_i^T V_{p,i-1}\} & D + \sum_{i=1}^m C_i^T V_{q,i-1} \left[\Delta \bar{q}_{i-1,T} \quad \Delta \bar{q}_{i-1,\gamma} \right] \end{bmatrix} \begin{bmatrix} \{\Delta \bar{p}_{i-1}\} \\ \Delta T \\ \Delta \gamma \end{bmatrix} = \begin{bmatrix} \{V_{p,i}^T r_i + F_{pq,i} V_{q,i-1} \Delta \bar{q}_{i-1,r}\} \\ s + \sum_{i=1}^m c_{s,i}^T V_{q,i-1} \Delta \bar{q}_{i-1,r} \\ n + \sum_{i=1}^m c_{n,i}^T V_{q,i-1} \Delta \bar{q}_{i-1,r} \end{bmatrix}, \tag{6.21}$$

and finally we set

$$\Delta \bar{q}_i = \Delta \bar{q}_{i,r} + \Delta T \, \Delta \bar{q}_{i,T} + \Delta \gamma \, \Delta \bar{q}_{i,\gamma}.$$

Just as in the single shooting case, we can guarantee the nonsingularity of the P-system (6.21) if (6.19) is nonsingular and if the Q-systems (6.20) are solved with enough accuracy. If γ is kept fixed, we can omit the unknown $\Delta \bar{q}_{i,\gamma}$ from (6.20) and the row and column corresponding to the pseudo-arclength condition and the parametrizing equation γ from (6.19) and (6.21).

The Q-system(s) (6.20) can be solved by condensation based on a forward recursion. Using the recursion

$$\Delta \bar{q}_{i,*} = F_{qq,i} \Delta \bar{q}_{i-1,*} + V_{q,i}^T r_{i,*}, \quad i = 1, \ldots, m-1, \tag{6.22}$$

one can compute $\Delta \bar{q}_{m-1,*}$ in terms of $\Delta \bar{q}_{0,*}$ and the residuals. In (6.22), $r_{i,*}$ denotes r_i, $b_{T,i}$ or $b_{\gamma,i}$ depending on the right-hand side of (6.20) that is being used. After substituting this relation into the last q equations, we obtain the condensed system

$$(F_{qq,m} \cdots F_{qq,1} - I_q) \Delta \bar{q}_{0,*} = -\hat{r}_{q,*}, \tag{6.23}$$

with

$$\hat{r}_{q,*} = V_{q,0}^T r_{m,*} + F_{qq,m} V_{q,m-1}^T r_{m-1,*} \cdots + F_{qq,m} \cdots F_{qq,2} V_{q,1}^T r_{1,*}.$$

From this system, we can solve $\Delta \bar{q}_0$ and then compute the other $\Delta \bar{q}_i$'s using (6.22).

We can prove the following important property for the condensed system (6.23).

Lemma 6.4 *Let*

$$M_0 \left[\begin{array}{cc} X_{p,0} & X_{q,0} \end{array} \right] = \left[\begin{array}{cc} X_{p,0} & X_{q,0} \end{array} \right] \left[\begin{array}{cc} \Lambda_p & 0 \\ 0 & \Lambda_q \end{array} \right]$$

be the Jordan decomposition of M_0 such that $X_{p,0}$ contains all generalized eigenvectors corresponding to $\mu_1, \cdots, \mu_p$. Let $M_{q,0} = F_{qq,m} F_{qq,m-1} \cdots F_{qq,1}$ with $F_{qq,i}$ as defined in (6.18), then

$$M_{q,0}(V_{q,0}^T X_{q,0}) = (V_{q,0}^T X_{q,0}) \Lambda_q$$

is the Jordan decomposition of $M_{q,0}$.

Proof. Let $X_{q,i} = G_i \cdots G_1 X_{q,0}$. From lemma 6.2 follows that $X_{q,i}$ is a set of generalized eigenvectors of M_i for all eigenvalues contained in Λ_q. Now, since $V_{q,i}^T G_i V_{p,i-1} = 0$ and $V_{q,i} V_{q,i}^T = I - V_{p,i} V_{p,i}^T$, we can show

$$\begin{array}{rcl} M_{q,0} V_{q,0}^T X_{q,0} & = & V_{q,0}^T G_m V_{q,m-1} \cdots V_{q,1}^T G_1 V_{q,0} V_{q,0}^T X_{q,0} \\ & (1) & \\ & = & V_{q,0}^T G_m V_{q,m-1} \cdots V_{q,1}^T G_1 X_{q,0} - V_{q,0}^T G_m V_{q,m-1} \cdots V_{q,1}^T G_1 V_{p,0} V_{p,0}^T X_{q,0} \\ & (2) & \\ & = & V_{q,0}^T G_m V_{q,m-1} \cdots G_2 V_{q,1} V_{q,1}^T X_{q,1} \\ & = & \cdots \\ & = & V_{q,0}^T G_m V_{q,m-1} V_{q,m-1}^T X_{q,m-1} \\ & = & V_{q,0}^T G_m X_{q,m-1} - V_{q,0}^T G_m V_{p,m-1} V_{p,m-1}^T X_{q,m} \\ & (3) & \\ & = & V_{q,0}^T M_0 X_{q,0} \\ & = & (V_{q,0}^T X_{q,0}) \Lambda_q. \end{array}$$

(1): $V_{q,0} V_{q,0}^T = I - V_{p,0} V_{p,0}^T$
(2): $V_{q,1}^T G_1 V_{p,0} = 0$
(3): Definition of $X_{q,m-1}$: $G_m X_{q,m-1} = G_m(G_{m-1} \cdots G_1 X_{q,0}) = M_0 X_{q,0}$. □

Corollary 6.5 *The spectral radius of the matrix $M_{q,0}$ is the modulus of the eigenvalue μ_{p+1}, i.e., $r_\sigma(M_{q,0}) = |\mu_{p+1}|$.*

As discussed in [4], compacting the Q-subsystem with forward recursion is generally stable since $r_\sigma(F_{qq,m} \cdots F_{qq,1}) < \rho < 1$. Remark however that in extreme cases, some components may grow along part of the limit cycle and then shrink again. This may cause problems in the recursion. A possible cure for this problem would be to add these modes to the $\mathcal{U}$-subspace—at least if there would be an easy procedure to isolate them—and solve the P-system using a good linear system solver, e.g., Gauss elimination with full pivoting. (6.23) is system of dimension $q \times q = (N-p) \times (N-p)$ and hence very high-dimensional. We want to avoid computing the matrices $F_{qq,i}$ and factoring the product. In analogy to the single shooting case, we solve (6.23) approximately using an iterative method. The other $\Delta\bar{q}_{i,*}$'s are computed by the recursion (6.22). In section 6.2.3, we will discuss two methods for solving the Q-system: Picard iteration and the GMRES method.

There are several possibilities to solve the P-system (6.21). This is already a rather small system compared to the original system and can be solved directly using Gauss elimination. Note that the P-system is very similar to the linear system (6.7). Hence we can use the two techniques discussed in section 6.1.2 to solve this system. If $|\mu_p|$ is close enough to 1, the system can be condensed to a $(p+2) \times (p+2)$-system in a stable way by only using a backward recursion, otherwise one has to use a combination of forward and backward recursion to compact the system. We will elaborate on this problem in section 6.2.4.

6.2.2 Monitoring the convergence

It is easy to generalize the convergence analysis of section 3.2.2 and the discussion in section 3.4 to the multiple shooting case.

Let us assume again that the P-system is solved exactly and that all matrix–vector products in the following relations are computed with high accuracy. Let

$$\begin{aligned} Q_i r_{i,r} &= Q_i G_i \Delta q_{i-1,r} - \Delta q_{i,r} + Q_i r_i, \\ Q_i r_{i,T} &= Q_i G_i \Delta q_{i-1,T} - \Delta q_{i,T} + Q_i b_{T,i}, \\ Q_i r_{i,\gamma} &= Q_i G_i \Delta q_{i-1,\gamma} - \Delta q_{i,\gamma} + Q_i b_{\gamma,i}, \end{aligned} \tag{6.24}$$

with $\Delta q_m := \Delta q_0$. It is possible to generalize (3.10) and (3.11) to

$$\begin{aligned} &Q_i r_i(x_{i-1} + \Delta q_{i-1} + \Delta p_{i-1}, x_i + \Delta q_i + \Delta p_i, T + \Delta T, \gamma + \Delta\gamma) = \\ &\qquad Q_i r_{i,r} + \Delta T\, Q_i r_{i,T} + \Delta\gamma\, Q_i r_{i,\gamma} + Q_i G_i \Delta p_{i-1} \\ &\qquad + \mathrm{O}(\Delta p_{i-1}, \Delta p_i, \Delta q_{i-1}, \Delta q_i, \Delta T, \Delta\gamma)^2 \end{aligned} \tag{6.25}$$

and

$$\left\{ \begin{aligned} &P_i r_i(x_{i-1} + \Delta q_{i-1} + \Delta p_{i-1}, x_i + \Delta q_i + \Delta p_i, T + \Delta T, \gamma + \Delta\gamma) \\ &\qquad = \mathrm{O}(\Delta p_{i-1}, \Delta p_i, \Delta q_{i-1}, \Delta q_i, \Delta T, \Delta\gamma)^2, \\ &s(x_0 + \Delta q_{i-1} + \Delta p_{i-1}, \ldots, x_m + \Delta q_m + \Delta p_m, T + \Delta T, \gamma + \Delta\gamma) \\ &\qquad = \mathrm{O}(\Delta p_{i-1}, \Delta p_i, \Delta q_{i-1}, \Delta q_i, \Delta T, \Delta\gamma)^2, \\ &n(x_0 + \Delta q_{i-1} + \Delta p_{i-1}, \ldots, x_m + \Delta q_m + \Delta p_m, T + \Delta T, \gamma + \Delta\gamma) \\ &\qquad = \mathrm{O}(\Delta p_{i-1}, \Delta p_i, \Delta q_{i-1}, \Delta q_i, \Delta T, \Delta\gamma)^2. \end{aligned} \right. \tag{6.26}$$

The remarks made in section 3.2.2 are also applicable to the multiple shooting case. After computing the new residuals at each of the mesh points, we can compute the higher order terms since all other quantities are easily derived from the basis iterations and the iterations of the Q-system solvers as we shall see later and require no new matrix–vector products. Hence we can use (6.25) and (6.26) for an a posteriori analysis of the convergence results just as before.

The a priori use of the relations to determine the convergence goals for the basis iteration process and the Q-system solvers is also possible. First we need to estimate the size of the Δp_i's, ΔT and $\Delta\gamma$, e.g., using

$$\begin{aligned} \|\Delta p_{i-1}\| &\approx \max\left\{\|r_{i-1}\|, \|r_i\|\right\}, \\ \Delta T &\approx \max_i \frac{\|r_i\|}{\|b_T\|}, \\ \Delta\gamma &\approx \max_i \frac{\|r_i\|}{\|b_\gamma\|}. \end{aligned}$$

which is an extension of (3.17) and (3.18) and are very rough estimates. Of course, one can also extend the estimates to take into account the phase- and pseudo-arclength conditions. Next we determine the estimated higher-order terms and set the convergence goal at each of the multiple shooting mesh points separately using the method described in section 3.4. As a result of this step, we obtain a goal for the residual at each individual mesh point. We then have two options. First, we can use the largest of these residual goals as the goal in all mesh points, i.e., we only try to decrease the maximum of the residual norms at each of the mesh points. This strategy allows that in some mesh points, the residual grows. The disadvantage of this strategy is that since it allows a growth of the residual in some mesh points, the higher order terms may also increase significantly and cause trouble at later iteration steps. Another option is to use individual convergence goals at each of the mesh points. Here we try to reduce the residuals at each of the mesh points. In this case we need to specify convergence thresholds at each of the mesh points for the Q-system solvers and the basis iteration process.

Note that if one consistently notes high higher-order terms at a given mesh point, one should consider to split the interval ending at that mesh point.

6.2.3 Solving the Q-system

In this section, we extend the Picard iteration scheme and the GMRES method discussed in section 3.5 to the multiple shooting case. In both approaches, we start from a method that solves the condensed Q-system (6.23). This system is very similar to the Q-system in the single shooting approach. The scheme is then rewritten to compute the other unknowns Δq_i and the terms $G_i \Delta q_{i-1}$ needed in the coupling between the Q- and P-system and the error analysis without new matrix–vector products.

Picard iteration

As discussed in section 6.2.1 (formula (6.23)), each of the systems contained in (6.20) can be reduced to

$$(F_{qq,m} \cdots F_{qq,1} - I_q)\Delta \bar{q}_{0,*} = -\hat{r}_{q,*} \tag{6.27}$$

using the forward recursion (6.22). $\hat{r}_{q,*}$ is easily computed using the following algorithm:

```
$\hat{r}_{q,*} \leftarrow 0$
for $i = 1$ to $m$ do
    $\hat{r}_{q,*} \leftarrow F_{qq,i}\hat{r}_{q,*} + V_{q,i}^T r_{i,*}$
end for.
```
(6.28)

In this algorithm, $r_{i,*}$ denotes either r_i, $b_{T,i}$ or $b_{\gamma,i}$. Note that this step requires $m-1$ matrix–vector products (with $F_{qq,2}$ to $F_{qq,m}$). These products were not needed in the single shooting case. From corollary 6.5, we learn $r_\sigma(F_{qq,m}\cdots F_{qq,1}) = \left|\mu_{p+1}\right| < \rho < 1$, so we can solve (6.27) using the Picard iteration scheme already used in the single shooting case:

```
$\Delta\bar{q}_{0,*} \leftarrow 0$ (or any other starting value)
for $j = 1$ to $l$ do
    $\Delta\bar{q}_{0,*} \leftarrow F_{qq,m}\cdots F_{qq,1}\Delta\bar{q}_{0,*} + \hat{r}_{q,*}$        (6.29)
end for
```

Once we have computed $\Delta\bar{q}_{0,*}$, the other unknowns can be computed from the forward recursion

$$\Delta\bar{q}_{i,*} \leftarrow F_{qq,i}\Delta\bar{q}_{i-1} + V_{q,i}^T r_{i,*}, \quad i = 1,\ldots,m-1. \tag{6.30}$$

These three recursions can be easily integrated into one overall scheme.

Algorithm 6.1 *Multiple shooting Picard iteration, short Q-vectors*

```
for $j = 0$ to $l$ do
    if $j = 0$ then
        $\Delta\bar{q}_{0,*} \leftarrow 0$ (or any other starting value)
    else
        $\Delta\bar{q}_{0,*} \leftarrow F_{qq,m}\Delta\bar{q}_{m-1,*} + V_{q,0}^T r_{m,*}$
    end if
    for $i = 1$ to $m-1$ do
        $\Delta\bar{q}_{i,*} \leftarrow F_{qq,i}\Delta\bar{q}_{i-1,*} + V_{q,i}^T r_{i,*}$
    end for
end for
```

The first run of the outer loop basically constructs the condensed residual (except for the last term). In each of the following iterations, a Picard step (6.29) is taken and the new $\Delta\bar{q}_0$ is propagated to the other unknowns $\Delta\bar{q}_i$ using (6.30).

This simple Picard scheme needs several modifications. First, we wish to avoid the use of the bases $V_{q,i}$ by immediately computing the N-dimensional vectors $\Delta q_{i,*} = V_{q,i}\Delta\bar{q}_{i,*}$. Therefore we premultiply the lines in algorithm 6.1 involving $\Delta\bar{q}_{i,*}$ with $V_{q,i}$. We slightly rearrange the order of operations to store the terms $G_i\Delta q_{i-1,*}$ ($i = 1,\ldots,m$) needed to build the P-system and to analyze the convergence. Furthermore, we wish to stop the algorithm if the algorithm has converged to a predetermined threshold instead of after a fixed number of Picard iteration steps. We also wish to extend the algorithm to use a nonzero starting value for all $\Delta q_{i,*}$. The latter is done by first adapting the right-hand sides $r_{i,*}$ and then starting the algorithm with all zero starting values. We obtain the following algorithm.

Algorithm 6.2 *Picard iteration for multiple shooting (N-dimensional vectors)*

```
Input:
    Starting values $\Delta q_{i,*}^{[0]}$, $i = 0,\ldots,m-1$
    Optionally: $G_i\Delta q_{i-1,*}^{[0]}$, $i = 0,\ldots,m-1$
    Convergence thresholds $\varepsilon_i$, $i = 1,\ldots,m$
    Bases $V_{p,i}$ for the projectors, right-hand sides $r_{i,\cdot *}$ and a routine to compute
    $G_i v$.
Output:
    $\Delta q_{i,*}$, $i = 0, \ldots, m-1$, approximate solutions of (6.20) for one of the right-
    hand sides.
    $(G\Delta q)_i := G_i \Delta q_{i-1,*}$, $i = 1, \ldots, m$
begin
    for $i = 1$ to $m$ do
        Compute $G_i \Delta q_{i-1}^{[0]}$ (if not given) and store in $(G\Delta q)_i^{[0]}$.
        $r_{i,*} \leftarrow Q_i \left( r_{i,*} + G_i \Delta q_{i-1}^{[0]} - \Delta q_i^{[0]} \right)$
    end for
    if $\forall i : \|r_{i,*}\| < \varepsilon_i$ then
        for $i = 1$ to $m$ do
            $\Delta q_{i-1} \leftarrow \Delta q_{i-1}^{[0]}$
            $(G\Delta q)_i \leftarrow (G\Delta q)_i^{[0]}$
        end for
        return
    end if
    $j \leftarrow 0$; $(G\Delta q)_m = 0$
    repeat
        if $j \neq 0$ or $r_{i,*} = 0$, $i = 1, \ldots, m-1$ then
            $\Delta q_{0,*} \leftarrow Q_0 (G\Delta q)_m + Q_0 r_{m,*}$
        else
            $\Delta q_{0,*} = 0$
        end if
        $(G\Delta q)_1 \leftarrow G_1 \Delta q_{0,*}$
        for $i = 1$ to $m-1$ do
            $\Delta q_{i,*} \leftarrow Q_i (G\Delta q)_i + Q_i r_{i,*}$
            $(G\Delta q)_{i+1} \leftarrow G_{i+1} \Delta q_{i,*}$
        end for
        $j \leftarrow j + 1$
    until $\|r_{m,*} + Q_0 (G\Delta q)_m - \Delta q_0\| < \varepsilon_m$
    for $i = 1$ to $m$ do
        $\Delta q_{i-1,*} \leftarrow \Delta q_{i-1,*} + \Delta q_{i-1,*}^{[0]}$
        $(G\Delta q)_i \leftarrow (G\Delta q)_i + (G\Delta q)_i^{[0]}$
    end for
end
```

The first iteration through the **repeat**-loop will compute values Δq_i, $i = 1, \ldots, m-1$, such that

$$Q_i (r_{i,*} + G_i \Delta q_{i-1,*} - \Delta q_{i,*}) = 0, \; i = 1, \ldots, m-1, \tag{6.31}$$

i.e., all linear residuals are put to zero except

$$Q_0 (r_{m,*} + G_m \Delta q_{m-1,*} - \Delta q_0), \tag{6.32}$$

which is just the condensed residual. This relation is preserved through all following iterations and also if we add the starting values again at the end. Hence, at a restart, the vectors $r_{i,*}$ will all be zero except $r_{m,*}$ if the basis has not changed and if we did at least one run through the **repeat**-loop at the previous run. The test $r_{i,*} = 0,\ i = 1,\dots,m-1$, avoids an unnecessary run trough the **repeat**-loop in this case. If the conditions in the test at the beginning of the **repeat**-loop are satisfied, a Picard step is taken (computing a new $\Delta q_{0,*}$) and corresponding values $\Delta q_i,\ i = 1,\dots,m-1$, are computed such that (6.31) holds. The condensed residual (6.32) converges asymptotically with linear asymptotic convergence factor $\left|\mu_{p+1}\right|$. Notice once more that this algorithm is usually more expensive than in the single shooting case for the same number of iterations. Before we can do the first Picard step, we need to compute the condensed residual. The exception to this rule happens when the starting values satisfy (6.31). In all other cases, we need a series of multiplications with $G_i,\ i = 2,\dots,m$, which was not needed in the corresponding case for the single-shooting method (i.e., all $G_i\Delta q_{i-1,*}^{[0]}$'s known corresponds to $M\Delta q^{[0]}$ known in the single shooting case) to construct the condensed residual. For $m = 1$, the algorithm reduces to algorithm 3.1. Note that if we have nonzero starting values $\Delta q_i^{[0]}$ but do not know the matrix–vector products $G_i\Delta q_{i-1}^{[0]}$, it is probably better not to use the starting values for $\Delta q_{1,*}$ to $\Delta q_{m-1,*}$ altogether and skip the convergence test before the **repeat**-loop. Instead, we can immediately start the repeat loop with $\Delta q_{0,*} = \Delta q_0^{[0]}$ instead of $\Delta q_{0,*} = 0$. In this case, we should also skip the **for**-loop at the end of algorithm 6.2. After one run through the **repeat**-loop we obtain identical results as in algorithm 6.2. The only difference is that in algorithm 6.2, we may find out that the starting values are accurate enough by computing only m matrix–vector products. If we modify algorithm 6.2 as suggested above and skip the convergence test at the beginning, we have computed the condensed residual after m matrix–vector products and values for $\Delta q_{i,*}$ satisfying (6.31). The condensed residual can be larger than each of the individual residuals. Hence it may happen that the initial residuals for the nonzero starting values each satisfy the convergence criterion at that point while the condensed residual and the values of $\Delta q_{i,*}$ computed during the first run through the **repeat**-loop do not. In this case, the modified algorithm requires more work. However, if the initial starting values are not accurate enough, we perform m unnecessary matrix–vector products in algorithm 6.2.

The GMRES method

We motivated the use of GMRES instead of Picard iteration in section 3.5.2. Since our Picard scheme basically solves the condensed system for $\Delta\bar{q}_{0,*}$ and computes the other $\Delta\bar{q}_{i,*}$'s using the forward recursion (6.22), the motivation also holds for the multiple shooting case. We will adapt the GMRES method of section 3.5.2 in a similar way as the Picard scheme such that it solves the condensed system for $\Delta q_{0,*}$ using the GMRES method and computes the other unknowns at no significant extra cost.

Suppose we have a starting value $\Delta q_{0,*}^{[0]}$. In the first step of the GMRES method , we will compute corresponding updates $\Delta q_{i,*}^{[0]}$ and the condensed residual $\hat{r}_{q,*}$ using the forward recursion

```
$\Delta q_{0,*}^{[0]} \leftarrow$ starting value
**for** $i = 1$ **to** $m-1$ **do**
    $\Delta q_{i,*}^{[0]} \leftarrow Q_i \left( G_i \Delta q_{i-1,*}^{[0]} + r_{i,*} \right)$        (6.33)
**end for**
$\hat{r}_{q,*} \leftarrow Q_0 \left( r_{m,*} + G_m \Delta q_{m-1,*}^{[0]} - \Delta q_{0,*}^{[0]} \right).$
```

(Note that we use N-dimensional vectors in this formula and not $(N-p)$-dimensional vectors as in (6.28)!) The recursion (6.33) puts all residuals (6.31) to zero except (6.32) which is again just the condensed residual $\hat{r}_{q,*}$. If starting values are given for all $\Delta q_{i,*}$, $i = 0, \ldots, m-1$, we can proceed as discussed at the end of section 6.2.3: if the products $G_i \Delta q_{i-1,*}$, $i = 1, \ldots, m$, are also known, we first check whether the starting values satisfy the convergence criterion at each of the multiple shooting meshpoints and correct the right-hand sides (and after the GMRES steps, the computed values for $\Delta q_{i,*}$, $i = 0, \ldots, m-1$), otherwise we only use $\Delta q_{0,*}^{[0]}$ and neglect the other starting values. If after the correction all right-hand sides $r_{i,*}$ are zero except $r_{m,*}$, we proceed with $\Delta q_{i,*}^{[0]} = 0$, $i = 0, \ldots, m-1$, and $\hat{r}_{q,*} = \hat{r}_{m,*}$ and do not execute (6.33). We will suppose that we only have a starting value for $\Delta q_{0,*}^{[0]}$ in the remainder of this section. We will apply the GMRES method to the condensed Q-system. Hence we need to construct an orthonormal basis $V_{0,l+1} = \left[\begin{array}{ccc} v_{0,1} & \cdots & v_{0,l+1} \end{array} \right] \in \mathbb{R}^{N \times (l+1)}$ for the Krylov subspace $\mathcal{K}_{l+1}(Q_0 M_0, -\hat{r}_{q,*})$ such that

$$v_{0,1} := -\frac{\hat{r}_{q,*}}{\|\hat{r}_{q,*}\|} \tag{6.34}$$

and

$$Q_0 M_0 V_{0,l} = V_{0,l+1} \bar{H}_l \tag{6.35}$$

where $\bar{H}_l \in \mathbb{R}^{(l+1) \times l}$ is a $(l+1) \times l$ upper Hessenberg matrix. During the construction of the basis, we shall save all information that is necessary to compute the other unknowns $\Delta q_{i,*}$ without an extra run through the forward recursion (6.22). The GMRES update is found by computing the least-squares solution of the projected system

$$\left(\bar{H}_l - \bar{I}_l \right) y_0 = \|\hat{r}_{q,*}\| \, e_1, \tag{6.36}$$

with e_1 the first column of the $(l+1) \times (l+1)$ unit matrix and $y_0 \in \mathbb{R}^l$. Finally we set $\Delta q_{0,*} = \Delta q_{0,*}^{[0]} + V_{0,l} y_0$. and compute the other unknowns using the forward recursion

$$\Delta q_{i,*} = Q_i (G_i \Delta q_{i-1,*} + r_{i,*}).$$

To construct all $\Delta q_{i,*}$'s simultaneously at no extra cost, we construct a basis in each multiple shooting meshpoint. These bases $V_{0,l+1}$, $V_{1,l}$, ..., $V_{m-1,l}$ satisfy the relationship

$$\begin{array}{rcl} Q_1 G_1 V_{0,l} & = & V_{1,l} H_{1,l} \\ & \vdots & \\ Q_{m-1} G_{m-1} V_{m-2,l} & = & V_{m-1,l} H_{m-1,l} \\ Q_0 G_m V_{m-1,l} & = & V_{0,l+1} \bar{H}_{m,l} \end{array} \tag{6.37}$$

with H_1, …, $H_{m-1} \in \mathbb{R}^{l\times l}$, $\bar{H}_m \in \mathbb{R}^{(l+1)\times l}$, and $v_{0,1} = -\hat{r}_{q,*}/\left\|\hat{r}_{q,*}\right\|$. In this relation, $V_{0,l+1}$ must define an orthonormal basis, but we are free to choose nonorthogonal bases $V_{i,l}$, $i > 0$. A very simple but efficient choice is taking $H_1 = \cdots = H_{m-1} = I_l$, which leads to the following algorithm.

Algorithm 6.3 *Arnoldi process for the construction of the Krylov subspaces for the GMRES(l) method to solve the Q-system.*

```
    Input: $\hat{r}_{q,*}$
    Output: $V_{i,l} = \begin{bmatrix} v_{i,1} & \cdots & v_{i,l} \end{bmatrix}$ $(i = 1,\ldots,m-1)$ and $V_{0,l+1} = \begin{bmatrix} v_{0,1} & \cdots & v_{0,l+1} \end{bmatrix}$
    such that (6.34) and (6.37) hold.
    $v_{0,1} \leftarrow -\dfrac{\hat{r}_{q,*}}{\left\|\hat{r}_{q,*}\right\|}$
    for $j = 1$ to $l$ do
        for $i = 1$ to $m-1$ do
            $v_{i,j} \leftarrow Q_i G_i v_{i-1,j}$
        end for
        $v_{0,j+1} \leftarrow Q_0 G_m v_{m-1,j}$
        for $k = 1$ to $j$ do
            $\bar{H}_m(k,j) \leftarrow <v_{0,j+1}, v_{0,k}>$
            $v_{0,j+1} \leftarrow v_{0,j+1} - \bar{H}_m[k,j] v_{0,k}$
        end for
        $\bar{H}_m[j+1,j] \leftarrow \left\|v_{0,j+1}\right\|_2$
        $v_{0,j+1} \leftarrow v_{0,j+1}/\bar{H}_m[j+1,j]$
    end for
```

In this case, the bases $V_{1,l}$, …, $V_{m-1,l}$ are not orthonormal and

$$Q_0 G_m Q_{m-1} G_{m-1} \cdots Q_1 G_1 V_{0,l} = Q_0 G_m \cdots G_1 V_{0,l} = V_{0,l+1} \bar{H}_{m,l}.$$

The procedure can be made numerically more stable by constructing orthonormal bases $V_{i,l}$. Then H_1, …, H_{m-1} are upper triangular matrices and

$$Q_0 G_m Q_{m-1} G_{m-1} \cdots Q_1 G_1 V_{0,l} = Q_0 G_m \cdots G_1 V_{0,l} = V_{0,l+1} \bar{H}_{m,l} H_{m-1,l} \cdots H_{1,l}.$$

This more complicated procedure is not really needed when using algorithm 6.3 in the GMRES(l)-method, but is certainly useful when using Arnoldi's method to compute the dominant eigenvalues of the monodromy matrix. The eigenvalue computation is much more sensitive to the basis quality as the GMRES method. Also, as we have seen in the previous chapter, it is not a good idea to concentrate all information in one matrix if small eigenvalues are to be computed.

After constructing bases $V_{i,l}$ and the relationship (6.37), we build the projected GMRES system

$$\left(\bar{H}_l - \bar{I}_l\right) y_0 = (\bar{H}_{m,l} H_{m-1,l} \cdots H_{1,l} - \bar{I}_l) y_0 = \left\|\hat{r}_{q,*}\right\| e_1 \tag{6.38}$$

(which is trivial if $H_{1,l} = \cdots = H_{m-1,l} = I_l$). In this formula, $\bar{I}_l$ is the $(l+1)\times l$- matrix consisting of the $l \times l$ unit matrix with one additional row of zeros.

The least-squares solution of (6.38) can be computed using the approach presented in [103]. Now let

$$\begin{aligned} y_i &= H_{i,l} y_{i-1} \in \mathbb{R}^l, \quad i = 1, \ldots, m-1, \\ y_m &= \bar{H}_{m,l} y_{m-1} \in \mathbb{R}^{l+1}. \end{aligned}$$

We have

$$\Delta q_{0,*} = V_{0,l} y_0 + \Delta q_{0,*}^{[0]}.$$

The other unknowns can then be computed from

$$\Delta q_{i,*} = V_{i,l} y_i + \Delta q_{i,*}^{[0]}, \quad i = 1, \ldots, m-1. \tag{6.39}$$

Proof. By induction. From the GMRES procedure, we have $\Delta q_{0,*} = V_{0,l} y_0 + \Delta q_{0,*}^{[0]}$. Suppose now $\Delta q_{i-1,*} = V_{i-1,l} y_{i-1} + \Delta q_{i-1,*}^{[0]}$, then

$$\begin{aligned} \Delta q_{i,*} &= Q_i G_i \Delta q_{i-1,*} + Q_i r_{i,*} \\ &= Q_i G_i V_{i-1,l} y_{i-1} + Q_i \left(G_i \Delta q_{i-1,*}^{[0]} + r_{i,*} \right) \\ &= V_{i,l} H_{i,l} y_{i-1} + \Delta q_{i,*}^{[0]} \\ &= V_{i,l} y_i + \Delta q_{i,*}^{[0]} \end{aligned}$$

□

Hence we can compute the other unknown $\Delta q_{i,*}$'s without new matrix–vector products with G_i provided we stored the intermediate values $\Delta q_{i,*}^{[0]}$ from (6.33).

It is also possible to recover the terms $Q_i G_i \Delta q_{i-1,*}$ (needed in the convergence analysis) and the terms $V_{p,i}^T G_i \Delta q_{i-1,*}$ (needed to construct the P-system) from the GMRES procedure with little extra work. From (6.37) and (6.39) follows that

$$\begin{aligned} Q_i G_i \Delta q_{i-1,*} &= Q_i G_i V_{i-1,l} y_{i-1} + Q_i G_i \Delta q_{i-1,*}^{[0]} \\ &= V_{i,l} H_{i,l} y_{i-1} + Q_i G_i \Delta q_{i-1,*}^{[0]} \\ &= V_{i,l} y_i + Q_i G_i \Delta q_{i-1,*}^{[0]} \quad (i = 1, \ldots, m-1) \end{aligned} \tag{6.40}$$

and

$$\begin{aligned} Q_0 G_m \Delta q_{m-1} &= Q_0 G_m V_{m-1,l} y_{m-1} + Q_0 G_m \Delta q_{m-1,*}^{[0]} \\ &= V_{0,l+1} \bar{H}_{m,l+1} y_{m-1} + Q_0 G_m \Delta q_{m-1,*}^{[0]} \\ &= V_{0,l+1} y_m + Q_0 G_m \Delta q_{m-1,*}^{[0]}. \end{aligned} \tag{6.41}$$

The terms $Q_i G_i \Delta q_{i-1,*}^{[0]}$ and $Q_0 G_m \Delta q_{m-1,*}^{[0]}$ have already been computed during the construction of the condensed residual (6.33). Furthermore,

$$V_{p,i}^T G_i \Delta q_{i-1,*} = V_{p,i}^T G_i V_{i-1,l} y_{i-1} + V_{p,i}^T G_i \Delta q_{i-1,*}^{[0]}. \tag{6.42}$$

The products $V_{p,i}^T G_i V_{i-1,l}$ can be computed and stored during the construction of the basis in algorithm 6.3 and the terms $V_{p,i}^T G_i \Delta q_{i-1,*}^{[0]}$ are easily computed in (6.33). Hence the terms $V_{p,i}^T G_i \Delta q_{i-1,*}$, $i = 1, \ldots, m$, can also be computed easily. Of course, it is also possible to compute the terms $G_i \Delta q_{i-1,*}$, $i = 1, \ldots, m$.

Sometimes, a large basis is needed to obtain a sufficiently accurate result. In that case, one may wish to use restarted GMRES. At a restart, we already have a set of starting values satisfying (6.31) and there is no need to execute (6.33) again. Instead, we set

$$\hat{r}_{q,*} = Q_0 \left(r_{m,*} + G_m \Delta q_{m-1,*} - \Delta q_{0,*} \right).$$

We also need to store the quantities $Q_i G_i \Delta q_{i-1,*}$, $Q_0 G_m \Delta q_{m-1,*}$ and $V_{p,i}^T G_i \Delta q_{i-1,*}$, $i = 1, \ldots, m$, which were computed from (6.40), (6.41) and (6.42) respectively. These quantities are needed at the end of the new GMRES step when we compute their new values (and will then appear in the right-hand sides). ν steps of the GMRES(l) method thus require the same amount of matrix–vector products as the Picard iteration scheme with equal start data. Remark that the GMRES(l) method is guaranteed to perform at least as good as the l-steps Picard iteration. However, there is no guarantee that the same holds for ν GMRES(l) steps compared to νl Picard iteration steps.

Both the Picard and the GMRES method which we presented are based on solving the condensed system. One can wonder whether other approaches are possible, e.g., approaches that try to minimize

$$\begin{bmatrix} Q_1(r_1 + G_1 \Delta q_0 - \Delta q_1) \\ \vdots \\ Q_0(r_m + G_m \Delta q_{m-1} - \Delta q_0) \end{bmatrix}$$

over some search space for the vectors $\Delta q_0, \ldots, \Delta q_{m-1}$, thus distributing the error over all the residuals. Several questions arise, such as which search space should one use, what convergence properties can one expect, ... There are avenues for further improvements. However, we are convinced that the current methods work well in most practical circumstances.

Starting values

We can also generalize the discussion of section 3.5.3 on starting values. As in the single shooting case, we will add components to the computed solutions of the Q-systems in the least dominant directions of the P-space before a basis change and remove unnecessary components afterwards.

Let $V_{p,i} = [V_{p,1,i} \quad V_{p,2,i}]$ in analogy to section 3.5.3. Here $V_{p,1,i}$ is a basis for the maximal invariant subspace of M_i for all Floquet multipliers larger than or equal to 1 in modulus (or some value $\tilde{\rho}$ slightly smaller than 1). Let $V_{p,1,i} \in \mathbb{R}^{p_1}$ and $V_{p,2,i} \in \mathbb{R}^{p_2}$. In direct analogy to (3.24) we solve

$$\begin{bmatrix} F_1 & F_2 \\ & F_4 \end{bmatrix} \begin{bmatrix} \{\Delta \bar{q}_{p,i-1,*}\} \\ \{\Delta \bar{q}_{q,i-1,*}\} \end{bmatrix} = - \begin{bmatrix} \{V_{p,2,i}^T r_{i,*}\} \\ \{V_{q,i}^T r_{i,*}\} \end{bmatrix}$$

with

$$F_1 = \begin{bmatrix} V_{p,2,1}^T G_1 V_{p,2,0} & -I_{p_2} & \\ & \ddots & \ddots \\ -I_{p_2} & & V_{p,2,0}^T G_m V_{p,2,m-1} \end{bmatrix},$$

$$F_2 = \begin{bmatrix} V_{p,2,1}^T G_1 V_{q,0} & & \\ & \ddots & \\ & & V_{p,2,0}^T G_m V_{q,m-1} \end{bmatrix} \text{ and}$$

$$F_4 = \begin{bmatrix} V_{q,1}^T G_1 V_{q,0} & -I_q & \\ & \ddots & \ddots \\ -I_q & & V_{q,0}^T G_m V_{q,m-1} \end{bmatrix}.$$

The block $F_4 \{\Delta \bar{q}_{q,i-1,*}\} = -\{V_{q,i}^T r_{i,*}\}$ is precisely the Q-system that has been solved using the Picard or GMRES method. The additional components are computed by moving $F_2 \{\Delta \bar{q}_{q,i-1,*}\}$ to the right-hand side and solving the first blockrow of equations for $\{\Delta \bar{q}_{p,i-1,*}\}$. These components are then added to the corresponding $\Delta q_{i,*}$'s. Note that the augmented $\Delta q_{i,*}$'s also satisfy (6.31), now with $Q_i = I - V_{p,1,i} V_{p,1,i}^T$.

After a basis change, unnecessary components are removed using

$$\left\{ \begin{array}{l} \Delta q_{i-1,*}^{[1]} \leftarrow \left(I - V_{p,i-1} V_{p,i-1}^T\right) \Delta q_{i-1,*}^{[0]} \\ G_i \Delta q_{i-1,*}^{[1]} \leftarrow G_i \Delta q_{i-1,*} - \left(G_i V_{p,i-1}\right) V_{p,i-1}^T \Delta q_{i-1,*}^{[0]} \end{array} \right. \quad i = 1, \ldots, m.$$

The terms $G_i V_{p,i-1}$ are recovered from the basis computations. Note that if the basis has changed, the relation (6.31) is broken and a restart is more expensive. However, if the basis size has increased, it is very well possible that

$$\left\| Q_i \left(r_{i,*} + G_i \Delta q_{i-1,*} - \Delta_{i,*}\right) \right\|_2$$

has decreased and it does make sense to also update the matrix–vector products $G_i \Delta q_{i-1,*}$ and to test the starting values before starting the Picard iterations or the GMRES method.

Just as in the single shooting case, $\Delta q_{i,r} = 0$ and $\Delta q_{i,T} = 0$ are reasonable starting values at the beginning of a new Newton–Picard step. For the system with right-hand side $\{b_{\gamma,i}\}$, one should consider to use the last value of $\Delta q_{0,\gamma}$ from the previous iteration step. It is also straightforward to extend the method to compute start values from the unused components of the bases in the subspace iteration process.

6.2.4 Solving the P-system

There are several options to solve the P-system (6.21). One can use Gauss elimination or a least squares solver just as in the single shooting case. With the least squares solver, one can again consider to omit the pseudo-arclength and/or phase conditions. These methods can be applied to the unreduced $(mp+2) \times (mp+2)$ P-system; or one can first reduce the system to a $(p+2) \times (p+2)$-system. Usually, this reduction can be done in a numerically stable way by using backward recursion. However, if ρ in assumption 6.3 is very small, a combination of forward and backward recursion is needed.

Let us first discuss condensation using backward recursion. Let

$$\begin{bmatrix} \tilde{B}_i & \tilde{r}_i \end{bmatrix} = \begin{bmatrix} B_i & r_i \end{bmatrix} + G_i \begin{bmatrix} \Delta q_{i-1,T} & \Delta q_{i-1,\gamma} & \Delta q_{i-1,r} \end{bmatrix}, \quad i = 1, \ldots, m,$$

and

$$\begin{bmatrix} \tilde{D} & \begin{matrix}\tilde{s}\\ \tilde{n}\end{matrix} \end{bmatrix} = \begin{bmatrix} D & \begin{matrix}s\\ n\end{matrix} \end{bmatrix} + \sum_{i=1}^{m} C_i^T \begin{bmatrix} \Delta q_{i-1,T} & \Delta q_{i-1,\gamma} & \Delta q_{i-1,r} \end{bmatrix}.$$

From (6.21) we can derive the backward recursion

$$\begin{aligned} \Delta\bar{p}_{i-1} &= F_{pp,i}^{-1}\Delta\bar{p}_i - F_{pp,i}^{-1}V_{p,i}^T\tilde{b}_{T,i}\Delta T - F_{pp,i}^{-1}V_{p,i}^T\tilde{b}_{\gamma,i}\Delta\gamma - F_{pp,i}^{-1}V_{p,i}^T\tilde{r}_i, \\ & i = m, \ldots, 1, \ \Delta\bar{p}_m := \Delta\bar{p}_0. \end{aligned} \tag{6.43}$$

By using this recursion for $i = m-1, \ldots, 1$, $\Delta\bar{p}_i$, $i = 1, \ldots, m-1$, can be expressed in terms of $\Delta\bar{p}_0$, and after substitution of these relations in (6.43) for $i = 1$ and the last two equations of (6.21), we obtain the $(p+2)\times(p+2)$-system

$$\begin{bmatrix} I - \hat{F} & \hat{B} \\ \hat{C}^T & \hat{D} \end{bmatrix} \begin{bmatrix} \Delta\bar{p}_0 \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \hat{r} \\ \hat{s} \\ \hat{n} \end{bmatrix}, \tag{6.44}$$

with

$$\begin{aligned} \hat{F} &= F_{pp,1}^{-1}\cdots F_{pp,m}^{-1}, \\ \begin{bmatrix} \hat{B} & \hat{r} \end{bmatrix} &= F_{pp,1}^{-1}V_{p,1}^T \begin{bmatrix} \tilde{B}_1 & \tilde{r}_1 \end{bmatrix} + F_{pp,1}^{-1}F_{pp,2}^{-1}V_{p,2}^T \begin{bmatrix} \tilde{B}_2 & \tilde{r}_2 \end{bmatrix} + \cdots \\ & \quad + F_{pp,1}^{-1}\cdots F_{pp,m}^{-1}V_{p,m}^T \begin{bmatrix} \tilde{B}_m & \tilde{r}_m \end{bmatrix}, \\ \hat{C}^T &= C_1^TV_{p,0} + C_2^TV_{p,1}F_{pp,2}^{-1}\cdots F_{pp,m}^{-1} + \cdots + C_m^TV_{p,m-1}F_{pp,m}^{-1}, \\ \begin{bmatrix} \hat{D} & \begin{matrix}\hat{s}\\ \hat{n}\end{matrix} \end{bmatrix} &= \begin{bmatrix} \tilde{D} & \begin{matrix}\tilde{s}\\ \tilde{n}\end{matrix} \end{bmatrix} \\ & \quad - C_2^TV_{p,1}(F_{pp,2}^{-1}V_{p,2}^T \begin{bmatrix} \tilde{B}_2 & \tilde{r}_2 \end{bmatrix} + \cdots + F_{pp,2}^{-1}\cdots F_{pp,m}^{-1}V_{p,m}^T \begin{bmatrix} \tilde{B}_m & \tilde{r}_m \end{bmatrix}) \\ & \quad - \cdots - C_m^TV_{p,m-1}F_{pp,m}^{-1}V_{p,m}^T \begin{bmatrix} \tilde{B}_m & \tilde{r}_m \end{bmatrix}, \end{aligned} \tag{6.45}$$

or, by left-multiplying the first p equations of (6.44) with $-C_1^TV_{p,0}$ and adding to the last two equations,

$$\begin{aligned} \hat{C}^T &= C_1^TV_{p,0}F_{pp,1}^{-1}\cdots F_{pp,m}^{-1} + \cdots + C_m^TV_{p,m-1}F_{pp,m}^{-1}, \\ \begin{bmatrix} \hat{D} & \begin{matrix}\hat{s}\\ \hat{n}\end{matrix} \end{bmatrix} &= \begin{bmatrix} \tilde{D} & \begin{matrix}\tilde{s}\\ \tilde{n}\end{matrix} \end{bmatrix} \\ & \quad - C_1^TV_{p,0}(F_{pp,1}^{-1}V_{p,1}^T \begin{bmatrix} \tilde{B}_1 & \tilde{r}_1 \end{bmatrix} + \cdots + F_{pp,1}^{-1}\cdots F_{pp,m}^{-1}V_{p,m}^T \begin{bmatrix} \tilde{B}_m & \tilde{r}_m \end{bmatrix}) \\ & \quad - \cdots - C_m^TV_{p,m-1}F_{pp,m}^{-1}V_{p,m}^T \begin{bmatrix} \tilde{B}_m & \tilde{r}_m \end{bmatrix}. \end{aligned} \tag{6.46}$$

The coefficients (6.45) can be easily computed from the algorithm

Algorithm 6.4 *Condensation of the P-system via backwards recursion, computation of the coefficients.*

```
$\hat{F} \leftarrow I_{p\times p}$, $\hat{B} \leftarrow 0_{p\times 2}$, $\hat{C} \leftarrow 0_{p\times 2}$, $\hat{D} = \tilde{D}$, $\hat{r} \leftarrow 0_{p\times 1}$, $\hat{s} = \tilde{s}$ and $\hat{n} = \tilde{n}$.
for $i = m$ to $1$ step $-1$
    $\begin{bmatrix} \hat{F} & \hat{B} & \hat{r} \end{bmatrix} \leftarrow F_{pp,i}^{-1} \begin{bmatrix} \hat{F} & \hat{B} + V_{p,i}^T\tilde{B}_i & \hat{r} + V_{p,i}^T\tilde{r}_i \end{bmatrix}$
    if $i > 1$ then
        $\begin{bmatrix} \hat{C}^T & \hat{D} & \begin{matrix}\hat{s}\\ \hat{n}\end{matrix} \end{bmatrix} \leftarrow \begin{bmatrix} \hat{C}^T & \hat{D} & \begin{matrix}\hat{s}\\ \hat{n}\end{matrix} \end{bmatrix} + C_i^TV_{p,i-1} \begin{bmatrix} \hat{F} & -\hat{B} & -\hat{r} \end{bmatrix}$
```

```
else
    $\hat{C}^T \leftarrow \hat{C}^T + C_1^T V_{p,0}$
```

Omitting the test and always taking the then-path results in variant (6.46). Remark that we will never explicitly compute the inverses. Instead we solve the linear system

$$F_{pp,i} \left[\begin{array}{ccc} \hat{F}_{new} & \hat{B}_{new} & \hat{r}_{new} \end{array} \right] = \left[\begin{array}{ccc} \hat{F}_{old} & \hat{B}_{old} & \hat{r}_{old} \end{array} \right]. \qquad (6.47)$$

Because of the way we construct our bases (see section 6.3, we use Schur vectors ordered corresponding to decreasing modulus of the corresponding eigenvalues), the matrices $F_{pp,i}$, $i = 1, \ldots, m-1$, are upper triangular and $F_{pp,m}$ is upper block triangular with only 1×1 and 2×2 blocks on the diagonal. Hence solving the systems (6.47) only requires backsubstitutions and possibly solving some 2×2-systems if $i = m$. After computing $\Delta\bar{p}_0$, ΔT and $\Delta\gamma$, we can compute $\Delta\bar{p}_i$, $i = m-1, \ldots, 1$ from (6.43). $\sigma_r(F_{4,1}^{-1} \cdots F_{4,m}^{-1}) = \sigma_r(F_{4,m} \cdots F_{4,1})^{-1} = |\mu_p|^{-1}$ indicates the numerical stability of the recursion (6.43). In our computations, $|\mu_p|^{-1}$ is typically smaller than 10 or 100 and (6.43) is stable enough. If $|\mu_p|^{-1}$ would be so large that the P-system isn't solved accurately enough anymore, a combination of forward and backward recursion can be used to build the condensed system. Note that (6.21) has exactly the same structure as (6.7) and the procedure outlined in section 6.1.2 can be used. Remark that the P-system is already decoupled because of the way we compute our bases.

In extreme cases where under forward time integration, perturbations in certain directions shrink over part of the limit cycle but globally grow, neither backward nor combined forward and backward recursion will perform well. In this case, one should consider to solve the unreduced system using Gauss elimination with a very good pivoting strategy.

6.2.5 Why does a Jacobi variant not make sense?

In chapter 2 we also studied a Jacobi variant of our single-shooting based method. Tests in section 2.5 and the analysis in section 3.3 showed that its convergence behaviour was inferior to the Gauss–Seidel variants. However, the Jacobi variant with one Picard step and one subspace iteration step is appealing if integrating the nonlinear system (1.4) together with the variational equations (2.43) instead of doing all time integrations separately results in a large reduction of the computing time.

In the multiple shooting case it is not possible to do all the time integrations together even if one would neglect the terms F_{pq}^o in (6.19). It is possible to do the computation for the condensed residual (6.28) together with the time integrations for the residuals r_i. However, we still need to execute (6.30) to compute the other unknowns, and this can only be done once the condensed residual is computed. The only difference between a Jacobi and Gauss–Seidel scheme would be that the latter also needs the matrix–vector product $G_m \Delta q_{m-1}$. Hence a Jacobi variant, which would usually perform worse than a Gauss–Seidel variant, is not interesting in the multiple shooting case. Therefore we only developed extensions of the more robust $NPGS$ and CNP methods of chapter 3.

6.3 Computation of the basis

In this section, we discuss how one can build and update the bases for the spaces $\mathcal{U}_i$ efficiently and in a stable way. Recall that $\mathcal{U}_i$ is defined to be the space spanned by all generalized eigenvectors of $M_i = G_i \cdots G_1 G_m \cdots G_{i+1}$ corresponding to Floquet multipliers of modulus greater than ρ. Our basis computation process should fulfil the following requirements:

1. It should return orthonormal bases $V_{p,i}$, $i = 0, \dots, m-1$ of size p for the dominant invariant subspaces $\mathcal{U}_i$ or for a set of dominant invariant subspaces $\mathcal{U}_{eff,i}$ of dimension $p_{eff} > p$. Remark that for the convergence of the Picard scheme, it is very important that the subspaces are dominant invariant subspaces, i.e., the subspaces should capture the algebraic multiplicity of each eigenvalue correctly.

2. To build the P-system, we need the matrices $V_{p,i}^T G_i V_{p,i-1}$, $i = 1, \dots, m-1$. Furthermore, for the basis accuracy criterion which we will present in section 6.3.3, we also need a basis V_m for the space spanned by the columns of $G_m V_{p,m-1}$. We will therefore construct bases $V_{p.i}$ and matrices S_{i+1}, $i = 0,\ \dots,\ m$ such that

$$V_{p,i} S_i = G_i V_{p,i-1}, \quad i = 1,\ \dots,\ m \tag{6.48a}$$

and

$$S_{m+1} = V_{p,0}^T V_{p,m}. \tag{6.48b}$$

Then

$$V_{p,i}^T G_i V_{p,i-1} = S_i, \quad i = 1,\ \dots,\ m$$

and

$$V_{p,0}^T G_m V_{p,m-1} = V_{p,0}^T V_{p,m} S_m = S_{m+1} S_m,$$

so the submatrices of the P-system are easily computed.

Remark that the basis $V_{p,m}$ is different from the basis $V_{p,m}$ used in the previous section, where $V_{p,m} := V_{p,0}$. In this section, $V_{p,0}$ and $V_{p,m}$ are different bases for the same subspace at convergence of the subspace iterations. *From now on, for the remainder of this section, $V_{p,m} \neq V_{p,0}$!*

As in the single shooting case, we will study methods based on orthogonal subspace iteration. We will present some extensions of the subspace iteration process to compute all the required bases together without significant extra costs. First we will describe a straightforward extension. In section 6.3.2, we will discuss why this method may fail and derive a better algorithm based on the periodic Schur decomposition. In section 6.3.3, we will discuss criteria to judge the accuracy of subsets of the basis and finally in section 6.3.4 we will present the strategy used to determine the number of vectors used during the iterations.

6.3.1 Ordinary orthogonal subspace iteration

Let us assume p is known. As in the single shooting case, we again use $p_e > p$ vectors for the subspace iterations to accelerate the convergence of the subspace procedure and to check whether all required basis vectors are found. Suppose p_e is also known.

Let

$$V_e^{[0]} = \left[\begin{array}{ccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p_e}^{[0]} \end{array} \right],$$

and assume the vectors $v_1^{[0]}, \ldots, v_p^{[0]}$ are an initial guess for a basis of $\mathcal{U}_0$ and $v_{p+1}^{[0]}, \ldots, v_{p_e}^{[0]}$ guesses for the next dominant Schur vectors of M_0. Our first algorithm is obtained by slightly changing algorithm 2.1 (subspace iteration with projection, no locking) for single shooting such that the bases $V_{p,i}$, $i = 1, \ldots, m-1$ are computed at virtually no extra cost by using the relationship between bases of the dominant subspaces of M_i for different values of i given by lemma 6.2.

Algorithm 6.1 *Ordinary orthogonal subspace iteration with projection for multiple shooting.*

```
Input:
    • $V_e^{[0]} = \left[ v_1^{[0]} \cdots v_p^{[0]} \cdots v_{p_e}^{[0]} \right]$, where the vectors $v_1^{[0]}, \ldots, v_p^{[0]}$ are an
      initial guess for a basis of $\mathcal{U}_0$ and $v_{p+1}^{[0]}, \ldots, v_{p_e}^{[0]}$ guesses for the next dominant
      Schur vectors of $M_0$.
    • routine to compute $Mv$
Output:
    • Orthonormal bases $V_{p,i}$, $i = 0, \ldots, m-1$ of dimension $p_{eff} = p$, where
      $\mathrm{Span}(V_{p,i})$ is a good approximation for $\mathcal{U}_i$, an orthonormal basis $V_{p,m}$, and
      matrices $S_i$, $i = 1, \ldots, m+1$ such that $G_i V_{p,i-1} = V_{p,i} S_i$, $i = 1, \ldots, m$ and
      $V_{p,0}^T V_{p,m} = S_{m+1}$.
    • $V_e$, a good start basis to restart the subspace iteration process.
begin
    $V_e \leftarrow \mathrm{Ortho}(V_e^{[0]})$
    repeat
        $V_{e,0} \leftarrow V_e$
        for $i = 1$ to $m$
            $V_{e,i} \leftarrow G_i V_{e,i-1}$
        end for
        Compute $U = V_{e,0}^T M_0 V_{e,0} = V_{e,0}^T V_{e,m}$
        Compute the Schur vectors $Y = \left[ y_1 \cdots y_{p_e} \right]$ of $U$, order them accord-
        ing to decreasing modulus of the corresponding eigenvalue. Let $UY = YS$
        $V_e \leftarrow \mathrm{Ortho}(V_{e,m} Y)$
    until convergence of the first $p$ vectors
    for $i = 0$ to $m$
        $V_{e,i} \leftarrow V_{e,i} Y[1{:}p]$
        Compute the QR-factorization $V_{e,i} = V_{p,i} R_i$
    end for
    for $i = 1$ to $m$
        $S_i \leftarrow R_i R_{i-1}^{-1}$
        end for
        $S_{m+1} \leftarrow R_0^{-T} S[1{:}p, 1{:}p] R_m^{-1}$
    end
```

At the end of the repeat-loop, we have

$$G_i V_{e,i-1} = V_{e,i}, \quad i = 1, \ldots, m, \tag{6.49}$$

$$V_{e,0}^T V_{e,m} = U. \tag{6.50}$$

(6.49) is maintained in the following for-loops, but now

$$V_{e,0}^T V_{e,m} = S[1{:}p, 1{:}p]. \tag{6.51}$$

Furthermore, from (6.49) and (6.51) also follows

$$\begin{aligned} &G_i V_{p,i-1} R_{i-1} = V_{p,i} R_i \Rightarrow G_i V_{p,i-1} = V_{p,i} R_i R_{i-1}^{-1} = V_{p,i} S_i, \ i = 1, \ldots, m, \text{ and} \\ &(V_{p,0} R_0)^T V_{p,m} R_m = S \Rightarrow V_{p,0}^T V_{p,m} = R_0^{-T} S[1{:}p, 1{:}p] R_m^{-1} = S_{m+1}. \end{aligned}$$

Remark that algorithm 6.1 may need a lot of iterations or even not end at all if p_e is chosen too small because not all eigenvectors converge. We can cope with this by increasing p_e in such cases until the first p eigenvalues do converge. We used such a strategy in our multiple shooting code and will explain this in section 6.3.4. This remark also holds for the next two algorithms presented in this section.

6.3.2 Improved orthogonal subspace iteration

Algorithm 6.1 has serious drawbacks if $|\mu_1|$ is very large compared to $|\mu_{p_e}|$. The columns of $V_{e,m}$ tend to lie all into the direction of the most dominant eigenvectors and information about the smaller eigenvalues is lost. This problem can be avoided by reorthogonalizing the matrices $V_{e,i}$ after each multiplication with one of the matrices G_i and choosing the shooting intervals small enough such that the interesting components do not grow or shrink too much whenever a multiplication with G_i is performed. Therefore we replace the step $V_{e,i} \leftarrow G_i V_{e,i-1}$ in algorithm 6.1 with $V_{e,i} R_i \leftarrow G_i V_{e,i-1}$, where $V_{e,i} R_i$ is the QR-factorization of $G_i V_{e,i-1}$. The projected monodromy matrix U is then given by

$$U = V_{e,0}^T M_0 V_{e,0} = \left(V_{e,0}^T V_{e,m} \right) R_m \cdots R_1 := R_{m+1} R_m \cdots R_1.$$

A second problem arises in the computation of the eigenvalues and Schur vectors of U. The smaller eigenvalues are very inaccurate if $|\mu_1|$ is very large compared to $|\mu_{p_e}|$. Instead of explicitly constructing U and computing its Schur decomposition, we will compute the periodic Schur decomposition proposed in chapter 5. Let $(\{Y_i, \ i = 0, \ldots, m\}, \{S_i, \ i = 1, \ldots, m+1\})$ be the ordered periodic Schur decomposition of $\{R_i, \ i = 1, \ldots, m+1\}$, i.e.,

$$\begin{aligned} R_i Y_{i-1} &= Y_i S_i, \quad i = 1, \ldots, m, \\ R_{m+1} Y_m &= Y_0 S_{m+1}, \text{ and} \\ U Y_0 &= (R_{m+1} \cdots R_1) Y_0 = Y_0 (S_{m+1} \cdots S_1), \end{aligned}$$

then since $G_i V_{e,i-1} = V_{e,i} R_i,\ i = 1, \ldots, m$,

$$G_i(V_{e,i-1}Y_{i-1}) = V_{e,i}R_iY_{i-1} = (V_{e,i}Y_i)S_i$$

and thus

$$(V_{e,i}Y_i)^T G_i(V_{e,i-1}Y_{i-1}) = S_i.$$

Furthermore, since $V_{e,0}^T V_{e,m} = R_{m+1}$,

$$(V_{e,0}Y_0)^T(V_{e,m}Y_m) = Y_0^T R_{m+1} Y_m = Y_0^T Y_0 S_{m+1} = S_{m+1}.$$

Hence it is easy to obtain the matrices $F_{pp,i}$ from the basis computation process without new matrix–vector products. The equivalent of the step $V_e \leftarrow \text{Ortho}(V_{e,m}Y)$ at the end of the subspace iteration loop in algorithm 6.1 is

$$\begin{aligned} V_e &\leftarrow \text{Ortho}(G_m \cdots G_1 V_{e,0} Y_0) \\ &= \text{Ortho}(G_m \cdots G_2 V_{e,1} R_1 Y_0) \\ &= \text{Ortho}(G_m \cdots G_2 V_{e,1} Y_1 S_1) \\ &= \cdots \\ &= \text{Ortho}(V_{e,m} Y_m S_m \cdots S_1) = V_{e,m} Y_m \end{aligned}$$

since $S_m \cdots S_1$ is an upper triangular matrix and $V_{e,m}Y_m$ an orthonormal basis.

Using the above suggestions, one obtains the following algorithm.

Algorithm 6.2 *Stable orthogonal subspace iteration with projection for multiple shooting.*

```
    Input:
        • $V_e^{[0]} = \left[ v_1^{[0]} \ \cdots \ v_p^{[0]} \ \cdots \ v_{p_e}^{[0]} \right]$, where the vectors $v_1^{[0]}, \ldots, v_p^{[0]}$ are an
        initial guess for a basis for $\mathcal{U}_0$ and $v_{p+1}^{[0]}, \ldots, v_{p_e}^{[0]}$ guesses for the next dominant
        Schur vectors of $M_0$.
        • routine to compute $Mv$
    Output:
        • Orthonormal bases $V_{p,i},\ i = 0, \ldots, m-1$ of dimension $p_{eff} = p$, where
        Span($V_{p,i}$) is a good approximation for $\mathcal{U}_i$, an orthonormal basis $V_{p,m}$, and
        matrices $S_i,\ i = 1, \ldots, m+1$ such that $G_i V_{p,i-1} = V_{p,i} S_i,\ i = 1, \ldots, m$ and
        $V_{p,0}^T V_{p,m} = S_{m+1}$.
        • $V_e$, a good start basis to restart the subspace iteration process.
    begin
        $V_e \leftarrow$ Ortho($V_e^{[0]}$)
        repeat
            $V_{e,0} \leftarrow V_e$
            for $i = 1$ to $m$
                Compute the orthonormal basis $V_{e,i}$ and the upper triangular matrix
                $R_i$ such that $V_{e,i} R_{e,i} = G_i V_{e,i-1}$ (e.g., using modified Gramm–Schmidt
                orthogonalization).
            end for
            Compute $R_{e,m+1} = V_{e,0}^T V_{e,m}$
            Compute the ordered periodic Schur decomposition $(\{Y_{e,i},\ i = 0,\dots,m\}$, $\{S_{e,i},\ i = 1,\dots,m+1\})$ of $\{R_{e,i},\ i = 1,\dots,m+1\}$. (Ordering: according to decreasing modulus of the corresponding eigenvalue.)
            $V_e \leftarrow V_{e,m} Y_{e,m}$
        until convergence of the first $p$ vectors
        for $i = 0$ to $m$ do
            $V_{p,i} \leftarrow V_{e,i} Y_{e,i}[1{:}p]$
            $S_{i+1} \leftarrow S_{e,i+1}[1{:}p, 1{:}p]$
        end for
    end
```

In exact algorithmic, the algorithms 6.1 and 6.2 are fully equivalent, and so we do not need to prove the theoretical convergence properties again. However, in finite precision arithmetic, the algorithm 6.2 avoids the numerical pitfalls of algorithm 6.1 if strongly growing or shrinking components are computed.

Now it is even more interesting than in the single shooting case to also consider locking of vectors. In fact, one of the reasons to introduce multiple shooting is to be able to compute very unstable periodic solutions with very large Floquet multipliers. These will converge in very few iteration steps, while more iteration steps are needed to obtain the smaller Floquet multipliers with the desired accuracy. If there are very large Floquet multipliers, we can save a lot of matrix–vector products by using locking. We can extend algorithm 6.2 with locking as follows:

Algorithm 6.3 *Stable orthogonal subspace iteration with projection and locking for multiple shooting.*

```
    Input:
        • $V_e^{[0]} = \left[\ v_1^{[0]}\ \cdots\ v_p^{[0]}\ \cdots\ v_{p_e}^{[0]}\ \right]$, where the vectors $v_1^{[0]}, \dots, v_p^{[0]}$ are an initial guess for a basis for $\mathcal{U}_0$ and $v_{p+1}^{[0]}, \dots, v_{p_e}^{[0]}$ guesses for the next dominant Schur vectors of $M_0$.
        • routine to compute $Mv$
    Output:
        • Orthonormal bases $V_{p,i},\ i = 0,\dots,m-1$ of dimension $p_{eff}$, where $\mathrm{Span}(V_{p,i})$ is a good approximation for $\mathcal{U}_i$, or for another set of (larger) dominant invariant subspaces, an orthonormal basis $V_{p,m}$, and matrices $S_i,\ i = 1,\dots,m+1$ such that $G_i V_{p,i-1} = V_{p,i} S_i,\ i = 1,\dots,m$ and $V_{p,0}^T V_{p,m} = S_{m+1}$.
        • $V_e$, a good start basis to restart the subspace iteration process.
    begin
        $V_{e,m} \leftarrow V_e^{[0]}$
        $p_{eff} = 0$
        repeat
            $V_{e,0}[p_{eff}+1{:}p_e] \leftarrow V_{e,m}[p_{eff}+1{:}p_e]$
            Orthonormalize the column vectors of $V_{e,0}$ starting at column $p_{eff}+1$
            .for $i = 1$ to $m$ do
                Compute $V_{work} \leftarrow G_i V_{e,i-1}[p_{eff}+1{:}p_e]$
            Update the decomposition
                $V_{e,i}R_{e,i} = G_iV_{e,i-1} = \begin{bmatrix} G_iV_{e,i-1}[1{:}p_{eff}] & V_{work} \end{bmatrix} = \begin{bmatrix} V_{e,i}[1{:}p_{eff}]R_{e,i}[1{:}p_{eff},1{:}p_{eff}] & V_{work} \end{bmatrix}$
            by orthonormalizing the column vectors of $G_iV_{e,i-1}$ starting at column
            $p_{eff}+1$. Note that the first $p_{eff}$ columns of $V_{e,i}$ and the upper $p_{eff} \times$
            $p_{eff}$-block of $R_{e,i}$ are known from the previous iteration step.
        end for
        Update $R_{e,m+1} = V_{e,0}^TV_{e,m}$. Note that the upper left square block of size
        $p_{eff}$ is known from the previous iteration step!
        Compute the ordered periodic Schur decomposition ($\{Y_{e,i},\ i=0,\dots,m\}$,
        $\{S_{e,i},\ i=1,\dots,m+1\}$) of $\{R_{e,i},\ i=1,\dots,m+1\}$. (Ordering: according
        to decreasing modulus of the corresponding eigenvalue.)
        $p_{eff} \leftarrow$ the number of accurate basis vectors
        for $i=0$ to $m-1$ do
            $V_{e,i}[1{:}p_{eff}] \leftarrow V_{e,i}Y_{e,i}[1{:}p_{eff}]$
            $R_{e,i+1}[1{:}p_{eff},1{:}p_{eff}] \leftarrow S_{e,i+1}[1{:}p_{eff},1{:}p_{eff}]$
        end for
        $V_{e,m} \leftarrow V_{e,m}Y_m$
        $R_{e,m+1} \leftarrow S_{e,m+1}$
    until $p_{eff} \geq p$
    for $i=0$ to $m$ do
        $V_{p,i} \leftarrow V_{e,i}[1{:}p_{eff}]$
        $S_{i+1} \leftarrow S_{e,i+1}[1{:}p_{eff},1{:}p_{eff}]$
    end for
end
```

Just as in the single shooting case, we have chosen for a variant that allows small updates of locked vectors and changes in the order of already locked vectors.

In the following two sections we will further complete this algorithm with an accuracy criterion and with a strategy to determine p_e.

6.3.3 Checking the accuracy of the basis

From the error expansion discussed in section 6.2.2, we learn that we have to control the size of the terms $Q_iG_i\Delta p_{i-1}$ $(i=1,\dots,m-1)$ and $Q_0G_m\Delta p_{m-1}$. Because of the way we construct our bases,

$$Q_iG_iV_{p,i-1} = 0 \text{ for } i=1,\dots,m-1,$$

where the equality is true up to the roundoff errors made in the basis computation algorithm. However, if the basis algorithm has not converged yet,

$$Q_0G_mV_{p,m-1} \neq 0.$$

We will adapt the criterion discussed in section 2.4.4. The criterion is developed for the subspace iteration algorithm with locking. We will lock the first k vectors if

$$\zeta_k = \max_{\substack{\|\bar{p}_k\|_2=1 \\ \bar{p}_k\in\mathbb{R}^k}} \left\| \left(I - V_{e,0}[1{:}k]V_{e,0}[1{:}k]^T\right) G_mV_{e,m-1}[1{:}k]\bar{p}_k \right\|_2$$

decreases below a user-determined threshold ϵ_{sub}. To shorten the notations somewhat, let

$$S_{i,k} = S_i[1{:}k, 1{:}k]$$

and

$$V_{i,k} = V_{e,i}[1{:}k, 1{:}k].$$

ζ_k is the largest singular value of

$$(I - V_{0,k}V_{0,k}^T)G_m V_{m-1,k}.$$

The derivation of an expression for ζ_k in terms of the matrices $S_{m,k}$ and $S_{m+1,k}$ is very similar to the derivation in section 2.4.4. Let

$$\begin{aligned} Z_k &= (I - V_{0,k}V_{0,k}^T)G_m V_{m-1,k} \\ &= V_{m,k}S_{m,k} - V_{0,k}V_{0,k}^T V_{m,k}S_{m,k} \\ &= V_{m,k}S_{0,k} - V_{0,k}S_{m+1,k}S_{m,k}. \end{aligned}$$

$\sigma_{max}(Z_k)$ is also the square root of the largest eigenvalue of $Z_k^T Z_k$. Hence

$$\begin{aligned} \zeta_k^2 &= \lambda_{max}(Z_k^T Z_k) \\ &= \lambda_{max}\left((V_{m,k}S_{0,k} - V_{0,k}S_{m+1,k}S_{m,k})^T(V_{m,k}S_{0,k} - V_{0,k}S_{m+1,k}S_{m,k})\right) \\ &= \lambda_{max}(S_{m,k}^T S_{m,k} - (S_{m+1,k}S_{m,k})^T S_{m+1,k}S_{m,k}). \end{aligned}$$

Note that S_m is an upper triangular matrix while S_{m+1} is a block upper triangular matrix with 1×1 blocks according to the real eigenvalues of $R_{m+1} \cdots R_1$ in algorithm 6.3 and 2×2 blocks corresponding to pairs of complex conjugate eigenvalues. It is easy to show that if $S_{m+1}[k+1, k] = 0$,

$$S_{m,k}^T S_{m,k} = (S_m^T S_m)[1{:}k, 1{:}k]$$

and

$$S_{m+1,k}S_{m,k} = (S_{m+1}S_m)[1{:}k, 1{:}k].$$

If one first computes

$$\begin{aligned} A_1 &= S_m^T S_m, \\ A_2 &= S_{m+1}S_m, \end{aligned}$$

then

$$Z_k^T Z_k = A_1[1{:}k, 1{:}k] - A_2[1{:}k, 1{:}k]^T A_2[1{:}k, 1{:}k],$$

and $\zeta_k = \sqrt{\lambda_{max}(Z_k^T Z_k)}$ can be computed efficiently for different values of k. One can also construct the matrix Z_p and compute its largest singular value and the largest singular values of the matrices Z_k, $k < p$, as outlined in section 2.4.4. However, this option requires more memory and the higher accuracy of the singular value returned by this procedure is not really required. The accuracy criterion is used in algorithm 6.3 in a similar way as in section 2.4.4. p_{eff} is again determined to be the largest number for which $\zeta_k < \epsilon_{sub}$ $\forall k \leq p_{eff}$ with $S_{m+1}[k+1, k] = 0$. We always unlock all vectors at a each new Newton–Picard step to make full use of the possibilities of the a priori and a posteriori convergence analysis.

Note that at this moment we use the same criterion in the extra basis iterations used to compute the dominant Floquet multipliers accurately after the Newton–Picard iterations. In this case we are interested in the accurate computation of the Floquet multipliers themselves and not only of the corresponding dominant invariant subspaces. Ill-conditioned multiple eigenvalues may occur, e.g., at fold points or at a transition of a pair of complex conjugate eigenvalues to two real eigenvalues (or vice-versa). It is impossible to compute those Floquet multipliers with the same high accuracy as algebraically simple eigenvalues. Suspicious accuracy of some of the Floquet multipliers should be detected and reported to the user, but the continuation code should not fail. In our current code, this detection is not yet done very well. However, the subspace iterations coupled to the criterion on the accuracy of the subspaces managed to produce the correct number of eigenvalues in the neighbourhood of the exact eigenvalue.

6.3.4 Determining the basis size

We already studied how to compute the bases needed in the Newton–Picard method provided we know the size of the basis, and also discussed how to determine the accuracy of the computed bases. In section 2.4.4, we briefly discussed criteria to compute the basis size. However, we paid special attention to this point in our multiple shooting code. Hence we shall do over the discussion in more detail.

Let us first set the requirements for the size of the basis based on the needs of the Picard iteration, the computation of the dominant Floquet multipliers and the basis iterations themselves.

1. The requirements for the bases in the Newton–Picard procedure were set in assumption 6.3. Let ρ_{iter} be the user-determined threshold for ρ in assumption 6.3. We will perform a Newton–Picard iteration step as soon as all p_{iter} basis vectors for eigenvalues $> \rho_{iter}$ in modulus have converged to the required accuracy (determined from the convergence analysis in section 6.2.2). Failing to meet this goal may lead to problems such as slow convergence or no convergence at all of the Q-system solvers.

2. One is often also interested in obtaining good estimates for the dominant Floquet multipliers to study bifurcation phenomena along the computed branch. We require that all $p_{Floquet}$ eigenvalues $> \rho_{Floquet}$ are computed with a predetermined accuracy after the last Newton–Picard step. Remark that we are here interested in an accurate computation of the eigenvalues themselves and not the corresponding eigenspaces.

3. We also want to guarantee a fast enough convergence of the subspace iterations for the dominant vectors. Therefore, we use yet another threshold
$$\rho_{sub} \leq \min(\rho_{iter}, \rho_{Floquet}).$$
The p_{sub} (generalized) eigenvectors corresponding to eigenvalues $> \rho_{sub}$ in modulus are all included in the basis used for the subspace iterations. Failing to meet this goal is not as bad as in the above cases. The subspace iterations may converge slowly, but this is easily detected and corrected as we shall see furtheron.

Our strategy to meet the above requirements is based on adding p_{extra} additional vectors to the basis. At any point during the iterations, we aim to have p_{extra} Schur vectors with corresponding eigenvalues smaller than ρ_{sub}. The strategy to add and remove basis vectors to or from the basis is based on the observation that the more dominant eigenvalues and corresponding Schur vectors converge faster than the less dominant ones in subspace iteration with projection. The accuracy criterion for the subspaces used in the strategy to determine p_{iter}, $p_{Floquet}$ and p_{sub} is the same criterion as the one used to determine the number of vectors that should be locked during the subspace iterations, but with a weaker threshold. After every subspace iteration step we compute the set of vectors $\{\mu_1, \ldots, \mu_k\}$ that are accurate enough according to this threshold.

1. Let μ_i be the smallest eigenvalue in modulus larger than ρ_{sub} in the set $\{\mu_1, \ldots, \mu_k\}$. We require that there are at least $i + p_{extra}$ vectors in the basis, i.e., $i + p_{extra} \geq n_{vec}$, the total number of basis vectors used during the subspace iterations.

2. Suppose μ_k has converged to an eigenvalue smaller than $r_{hist}\rho_{sub}$ (where $r_{hist} \leq 1$). We proceed as explained in section 2.4.4. Let μ_j be the largest eigenvalue in the set $\{\mu_1, \ldots, \mu_k\}$ that is larger than $r_{hist}\rho_{sub}$. There should be at most p_{extra} vectors in the basis corresponding to smaller eigenvalues, i.e., $j + p_{extra} \leq n_{vec}$. The hysteresis factor r_{hist} is introduced to avoid adding vectors that are immediately removed again after some more subspace iteration steps or vice-versa because the bases evolve during the Newton–Picard iterations. It also avoids problems with the noise on the computation of ill-conditioned eigenvalues. We typically use $r_{hist} = 0.8$ or 0.9. Since this criterion may cause trouble if larger eigenvalues are initially underestimated and since using too much vectors in the basis for the subspace iterations only affects the performance and not the reliability, we only remove vectors once the subspace iteration procedure has converged.

3. We stop the subspace iterations as soon as all eigenvalues larger than ρ_{iter} (or $\rho_{Floquet}$ when we are computing the dominant Floquet multipliers once the periodic orbit has been computed) have converged to the predescribed basis accuracy. Checking this is done by assuring that the largest eigenvalue estimate that has not converged to the predescribed basis accuracy, has converged to a value smaller than ρ_{iter} (or $\rho_{Floquet}$) according to accuracy threshold used for the decision on adding and removing basis vectors, i.e., the subspace iterations in algorithm 6.3 are stopped if $k > p_{eff}$ and if $\left|\mu_{p_{eff}+1}\right| < \rho_{iter}$ or $\rho_{Floquet}$.

If the criterion to stop the basis iterations is fulfilled, requirement 1 and 2 are usually met. However, there is no absolute guarantee. This criterion is certainly not enough to assure that requirement 3 is also met if ρ_{sub} is smaller than ρ_{iter} or $\rho_{Floquet}$. Therefore we offer the user the option to require that at least one eigenvalue below ρ_{sub} in modulus has sufficiently converged (again using the same threshold as for the decision on adding and removing vectors). This option also gives more confidence that the first two requirements are fulfilled. The requirements may not be met if some of the more dominant eigenvalues are initially underestimated during the iterations. However, this is not very likely to occur in our applications, since we use projection (and so the more dominant eigenvalues converge faster than the less dominant ones) and usually have a very well spread-out spectrum.

If p_{extra} is chosen too small, the smaller eigenvalues may have trouble converging, e.g., if only half of a complex pair is included in the basis or if a cluster of eigenvalues near an eigenvalue with algebraic multiplicity > 1 is not fully captured. To avoid this problem, we increase p_{extra} with $p_{extra,+}$ (typically one or two) after ν_{sub} iterations, and we do this again every ν_{repeat} iterations. p_{extra} is only put back to its original value after finishing all the computations for the periodic orbit.

6.4 Further research

In this chapter, we introduced an extension of the Newton–Picard method to multiple shooting. Several topics deserve further investigation.

The implementation of the a priori and a posteriori convergence analysis needs further refinements. In particular, it is worth to develop good estimators for the higher order terms and the size of Δp_i, ΔT and $\Delta\gamma$ at the beginning of a Newton–Picard step. It would also be useful to combine the procedure with damping to further enhance the domain of attraction of the method. Damping is particularly useful to compute the first orbit on a branch or if one is interested in computing a single orbit at a given parameter value instead of a branch.

In this chapter, we made abstraction of all time integration aspects. To complete the method, one should add a procedure to select the multiple shooting meshpoints (1.27). Two aspects have to be taken into account:

- To obtain a large enough domain of attraction, it is important to keep the higher order terms small. Hence, if on an interval, the higher-order terms in the convergence analysis are consistently larger than on other intervals, the interval should be split.

- To make sure that the eigenvalues and basis vectors are computed accurately, we will limit the growth of the unstable modes over each interval. Information on this growth can be recovered from the matrices S_i in the subspace algorithm.

Besides a strategy to update the multiple shooting mesh, we also need a strategy to update the mesh used in the time integrator on each of the intervals. This mesh should also be kept fixed as much as possible, since the exact computed results depend on the mesh. Continuously changing the mesh may prevent the algorithm from converging.

Instead of solving (6.1), we can consider to solve the nonlinear system

$$\left\{\begin{array}{rcl} \varphi(x_0, \nabla s_1 T, \gamma) - x_1 &=& 0, \\ &\vdots& \\ \varphi(x_{m-1}, \nabla s_m T, \gamma) - x_m &=& 0, \\ x_m - x_0 &=& 0, \\ s(x_0, x_1, \ldots, x_{m-1}, T, \gamma) &=& 0, \\ n(x_0, x_1, \ldots, x_{m-1}, T, \gamma; \eta) &=& 0, \end{array}\right. \tag{6.52}$$

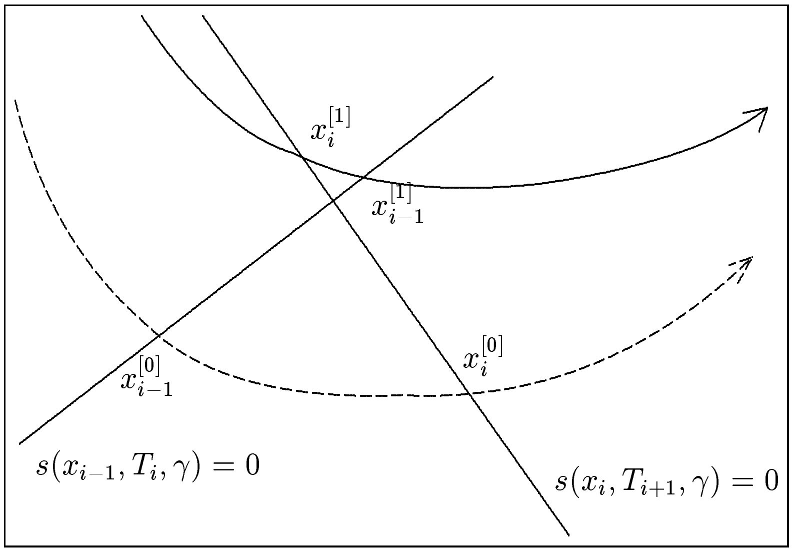


Figure 6.2: Failure of the geometric phase condition.

i.e., the boundary conditions are specified as a separate set of equations and an additional N-dimensional unknown x_m is introduced. This will not imply significant changes to the Newton–Picard procedure but may have some implementation advantages if the method is extended to the computation of bifurcation points. As we shall see in the next chapter, similar linearized systems will occur but with different boundary conditions. We chose to work out (6.1) instead of (6.52) because (6.1) reduces to the single shooting system (2.16) if $m = 1$.

Another approach, suggested to us by Dr. A. Khibnik (private communication), is the use of a "geometric phase condition". Instead of solving (6.1), we can solve the system

$$\left\{\begin{array}{rcl} \varphi(x_0, T_1, \gamma) - x_1 & = & 0, \\ s(x_0, T_1, \gamma) & = & 0, \\ & \vdots & \\ \varphi(x_{m-1} T_m, \gamma) - x_m & = & 0, \\ s(x_{m-1}, T_m, \gamma) & = & 0, \\ n(x_0, T_1, \ldots, x_{m-1}, T_m, \gamma; \eta) & = & 0. \end{array}\right.$$

The period of the orbit is $T = \sum_{i=1}^{m} T_i$. Each of the conditions $s(x_{i-1}, T_i, \gamma) = 0$ expresses a hypersurface on which the point (x_{i-1}, T_i, γ) may move. One option for these conditions is to express that x_{i-1} should lie in a hyperplane orthogonal to the starting orbit or another reference orbit. Such a strategy may lead to smaller changes of the points x_i from one orbit to another, in particular if a homoclinic orbit is approached. This may reduce the need for remeshing in the time integrator. A problem occurs if two of the hypersurfaces intersect between the starting and final orbits. This scenario is graphically represented in Figure 6.2. In this case, we would compute a negative value for T_i and this is impossible. Our applications usually result in very stiff systems that are extremely unstable under backward time integration. A strategy based on the geometric phase condition will need to incorporate a solution to this problem, e.g., by exchanging the points x_{i-1} and x_i or by replacing them by one point.

6.5 Conclusions

In this chapter, we discussed multiple shooting based methods. The first part of the chapter was devoted to traditional direct methods to solve the linearized systems in a Newton-based multiple shooting code. First we discussed a technique to solve such a system by condensing the system to a small system using forward recursion. This algorithm is unstable if the limit cycle is unstable. Next we discussed an algorithm based on the ideas of reorthogonalization and decoupling of [4]. In this method, the growing components are treated using backward recursion and the shrinking components using forward recursion. The decoupling is based on the periodic Schur decomposition discussed in the previous chapter. We discussed two possibilities to extend the method of [4] to be able to cope with the bordering from the phase- and pseudo-arclength conditions and derivatives with respect to the period or a parameter. These methods are stable but too expensive to be used for large-scale problems. However, coupled to the ideas behind our single shooting Newton–Picard methods, they give lead to a powerful new method and this is the subject of the second part of this chapter.

Our multiple shooting based techniques are based on the algebraic framework derived in chapter 3. We extended the $NPGS$ and CNP methods. We first generalized the decomposition. A different basis is needed in each multiple shooting point. However, these bases are related to each other and this relationship is fully exploited to obtain a technique that is not much more expensive than single shooting. The decomposition is very similar to the decomposition in the technique based on [4]. Next we extended the convergence analysis of section 3.2. We extended the Picard iteration scheme and the GMRES method to solve the Q-system and also discussed various techniques to solve the P-system. This system is very similar to the system obtained with the approach of [4]. We also extended the subspace iteration method with projection and locking to compute all bases simultaneously in a numerically very stable way. This algorithm fully exploits the relationship between the bases to compute all bases at a cost comparable to the single shooting case. The projection strategy is based on the periodic Schur decomposition. We also discussed an accuracy criterion for the bases and a strategy to determine the basis size.

We made an implementation using the subspace algorithm with projection and locking proposed in this chapter and Picard iterations to solve the Q-system. The unreduced P-system is solved using Gauss elimination with partial pivoting or an SVD-based least-squares solver.

The extension of the technique of [4] and the Newton–Picard multiple shooting technique, including the Q-system solvers and the new variants of subspace iteration, are original contributions of this thesis.

Chapter 7

Computing bifurcation points

This chapter is concerned with the extension of our Newton-Picard methods to the computation of various types of codimension-one bifurcation points: the period doubling point, the torus bifurcation point and the fold point. These bifurcation points were introduced in section 1.2. Our method uses extended systems that can be written down as extended boundary value problems and express conditions for an eigenvalue and corresponding eigenvector of the monodromy matrix. This approach is similar to the approach used in AUTO. We introduced this type of methods in section 1.6.2.

In the first part of this chapter, we work out a single-shooting based method to compute period doubling points. We present the method and prove that all subsystems that need to be solved are nonsingular if the method is implemented well. We discuss some particular implementation aspects and also present a testcase. In the remainder part of the chapter we suggest various extensions and alternatives. The extensions have not yet been thoroughly investigated but we wish to mention them anyway to demonstrate the power of the approach and as suggestions for further research. First we give some hints on how the method can be generalized to the torus bifurcation case. In this case we need to make changes to the Picard iteration and we prove the convergence of this scheme. Next we briefly mention the computation of fold points. Here one of the P-systems is singular and this causes trouble. The methods can also be generalized to multiple shooting. We end the chapter with a discussion on alternative approaches and with our conclusions.

7.1 Period doubling points

7.1.1 Derivation

Our method is based on the extended system (1.61). Single shooting applied to this system results in the nonlinear system

$$\begin{cases} r(x(0),T,\gamma) = \varphi(x(0),T,\gamma) - x(0) = 0, \\ s(x(0),T,\gamma) = 0, \\ (M(x(0),T,\gamma) + I)v = 0, \\ l^T v - 1 = 0. \end{cases} \tag{7.1}$$

This is a system of $2N + 2$ equations in the two N-dimensional vector unknowns $x(0)$ and v and the two scalar unknowns T and γ. At the solution, v is an eigenvector for the

Floquet multiplier -1. l is a vector that should not be orthogonal to the eigenvector of M for the Floquet multiplier -1 at the period doubling point. We can easily compute such a vector by taking a linear combination of the eigenvectors for eigenvalues close to -1 of $M(x(0)^{[0]}, T^{[0]}, \gamma^{[0]})$, the starting point for the computations. Our method will fully exploit the structure of (7.1).

Newton's method applied to (7.1) results in the repeated solution of linear systems

$$\begin{bmatrix} M-I & b_T & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & d_{s,\gamma} \\ M_x v & M_T v & M+I & M_\gamma v \\ 0 & 0 & l^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta v \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r(x(0),T,\gamma) \\ s(x(0),T,\gamma) \\ (M(x(0),T,\gamma)+I)v \\ l^T v - 1 \end{bmatrix}, \qquad (7.2)$$

where

$$\begin{bmatrix} M-I & b_T & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & d_{s,\gamma} \\ M_x v & M_T v & M+I & M_\gamma v \\ 0 & 0 & l^T & 0 \end{bmatrix} = \frac{\partial(r, s, (M+I)v, l^T v)}{\partial(x(0), T, v, \gamma)}.$$

One can solve this system by first computing Δx_r, Δx_γ, ΔT_r and ΔT_γ from

$$\begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta x_r & \Delta x_\gamma \\ \Delta T_r & \Delta T_\gamma \end{bmatrix} = - \begin{bmatrix} r & b_\gamma \\ s & d_{s,\gamma} \end{bmatrix}, \qquad (7.3)$$

then computing Δv and $\Delta\gamma$ from

$$\begin{bmatrix} M+I & M_\gamma v + M_x v \Delta x_\gamma + M_T v \Delta T_\gamma \\ l^T & 0 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} (M+I)v + M_x v \Delta x_r + M_T v \Delta T_r \\ l^T v - 1 \end{bmatrix}, \qquad (7.4)$$

and finally computing

$$\begin{aligned} \Delta x(0) &= \Delta x_r + \Delta\gamma\, \Delta x_\gamma, \\ \Delta T &= \Delta T_r + \Delta\gamma\, \Delta T_\gamma. \end{aligned}$$

This algorithm avoids the computation of the full matrix $M_x v$ and requires only two matrix–vector products with this matrix. However, it requires the construction of the monodromy matrix and we wish to avoid this in our method.

We extend assumption 2.1 to our particular case.

Assumption 7.1 *Let $y^* = (x(0)^*, T^*, v^*, \gamma^*)$ denote an isolated solution to (7.1), and let $\mathcal{B}$ be a small neighbourhood of y^*. Let $M(y) = \frac{\partial\varphi}{\partial x(0)}(y)$ for $y \in \mathcal{B}$ and denote its eigenvalues by μ_i, $i = 1, \ldots, N$. Assume that for all $y \in \mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1$$

and that no eigenvalue has modulus ρ; i.e., for all $y \in \mathcal{B}$,

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

Bases V_p and V_q and corresponding orthogonal projectors P and Q are constructed as before. $\Delta x(0)$ and Δv have the unique decomposition

$$\begin{array}{l}\Delta x(0) = V_p\Delta\bar{x}_p + V_q\Delta\bar{x}_q,\ \Delta x_p = V_p\Delta\bar{x}_p = P\Delta x(0),\ \Delta x_q = V_q\Delta\bar{x}_q = Q\Delta x(0),\\ \Delta v = V_p\Delta\bar{v}_p + V_q\Delta\bar{v}_q,\ \Delta v_p = V_p\Delta\bar{v}_p = P\Delta v,\ \Delta v_q = V_q\Delta\bar{v}_q = Q\Delta v\end{array} \tag{7.5}$$

with Δx_p, Δx_q, Δv_p, $\Delta v_q \in \mathbb{R}^N$, $\Delta\bar{x}_p$, $\Delta\bar{v}_p \in \mathbb{R}^p$ and $\Delta\bar{x}_q$, $\Delta\bar{v}_q \in \mathbb{R}^{N-p} = \mathbb{R}^q$.

After substitution of (7.5) into (7.2) and multiplying the first and the third set of equations at the left hand side with $[V_q\ \ V_p]^T$, we get

$$\begin{bmatrix} V_q^TMV_q - I_q & 0 & 0 & 0 & 0 & V_q^Tb_\gamma \\ V_p^TMV_q & V_p^TMV_p - I_p & V_p^Tb_T & 0 & 0 & V_p^Tb_\gamma \\ c_s^TV_q & c_s^TV_p & d_{s,T} & 0 & 0 & d_{s,\gamma} \\ V_q^TM_xvV_q & V_q^TM_xvV_p & V_q^TM_Tv & V_q^TMV_q + I_q & 0 & V_q^TM_\gamma v \\ V_p^TM_xvV_q & V_p^TM_xvV_p & V_p^TM_Tv & V_p^TMV_q & V_p^TMV_p + I_p & V_p^TM_\gamma v \\ 0 & 0 & 0 & l^TV_q & l^TV_p & 0 \end{bmatrix} \begin{bmatrix} \Delta\bar{x}_q \\ \Delta\bar{x}_p \\ \Delta T \\ \Delta\bar{v}_q \\ \Delta\bar{v}_p \\ \Delta\gamma \end{bmatrix} = -\begin{bmatrix} V_q^Tr \\ V_p^Tr \\ s \\ V_q^T(M+I)v \\ V_p^T(M+I)v \\ d^Tv - 1 \end{bmatrix}. \tag{7.6}$$

We again used $V_q^TV_p = 0$, $V_p^TV_q = 0$, $V_q^TMV_p = 0$ for a perfect basis and we neglected the term $V_q^Tb_T$. The matrix of this system has some similarities with the matrix of (2.21). We use the Moore–and–Spence-like approach to solve this system. From the first q equations, we get

$$\Delta\bar{x}_q = -\left(V_q^TMV_q - I_q\right)^{-1}\left[V_q^Tr + \Delta\gamma\, V_q^Tb_\gamma\right].$$

Hence we set

$$\Delta\bar{x}_q = \Delta\bar{x}_{q,r} + \Delta\gamma\Delta\bar{x}_{q,\gamma} \tag{7.7}$$

and solve $\bar{x}_{q,r}$ and $\Delta\bar{x}_{q,\gamma}$ from

$$\left[V_q^TMV_q - I_q\right]\left[\ \Delta\bar{x}_{q,r}\quad \Delta\bar{x}_{q,\gamma}\ \right] = -\left[\ V_q^Tr\quad V_q^Tb_\gamma\ \right]. \tag{7.8}$$

We can only compute $\Delta\bar{x}_q$ once $\Delta\gamma$ is known. Then we substitute (7.7) in the next $p+1$ equations of (7.6) and get

$$\begin{bmatrix} V_p^TMV_p - I_p & V_p^Tb_T \\ c_s^TV_p & d_{s,T} \end{bmatrix}\begin{bmatrix} \Delta\bar{x}_p \\ \Delta T \end{bmatrix} = -\begin{bmatrix} V_p^T\left(r + M\Delta x_{q,r}\right) + \Delta\gamma\, V_p^T\left(b_\gamma + M\Delta x_{q,\gamma}\right) \\ \left(s + c_s^T\Delta x_{q,r}\right) + \Delta\gamma\left(d_{s,T} + c_s^T\Delta x_{q,\gamma}\right) \end{bmatrix}.$$

Now we set

$$\begin{array}{rcl}\Delta\bar{x}_p &=& \Delta\bar{x}_{p,r} + \Delta\gamma\Delta\bar{x}_{q,\gamma},\\ \Delta T &=& \Delta T_r + \Delta\gamma\Delta T_\gamma,\end{array} \tag{7.9}$$

and compute $\Delta\bar{x}_{p,r}$, $\Delta\bar{x}_{q,\gamma}$, ΔT_r and ΔT_γ from

$$\begin{array}{l}\begin{bmatrix} V_p^TMV_p - I_p & V_p^Tb_T \\ c_s^TV_p & d_{s,T} \end{bmatrix}\begin{bmatrix} \Delta\bar{x}_{p,r} & \Delta\bar{x}_{q,\gamma} \\ \Delta T_r & \Delta T_\gamma \end{bmatrix} = \\ \quad -\begin{bmatrix} V_p^T\left(r + M\Delta x_{q,r}\right) & V_p^T\left(b_\gamma + M\Delta x_{q,\gamma}\right) \\ s + c_s^T\Delta x_{q,r} & d_{s,\gamma} + c_s^T\Delta x_{q,\gamma} \end{bmatrix},\end{array} \tag{7.10}$$

e.g., using Gauss elimination. We set

$$\Delta x_r = V_q \Delta \bar{x}_{q,r} + V_p \Delta \bar{x}_{p,r}, \; \Delta x_\gamma = V_q \Delta \bar{x}_{q,\gamma} + V_p \Delta \bar{x}_{p,\gamma}. \tag{7.11}$$

After substituting (7.9) and (7.11) in the next q equations of (7.6), we get

$$\begin{aligned}\left[V_q^T M V_q + I_q\right]\left[\Delta \bar{v}_q\right] = -\, &\left[V_q^T \left((M+I)v + M_x v \Delta x_r + M_T v \Delta T_r\right) + \right.\\ &\left.\Delta\gamma\, V_q^T \left(M_\gamma v + M_x v \Delta x_\gamma + M_T v \Delta T_\gamma\right)\right].\end{aligned}$$

Hence we set

$$\Delta \bar{v}_q = \Delta \bar{v}_{q,r} + \Delta\gamma \Delta \bar{v}_{q,\gamma} \tag{7.12}$$

and solve $\Delta \bar{v}_{q,r}$ and $\Delta \bar{v}_{q,\gamma}$ from

$$\left[V_q^T M V_q + I_q\right]\left[\begin{array}{cc} \Delta \bar{v}_{q,r} & \Delta \bar{v}_{q,\gamma}\end{array}\right] = -\left[\begin{array}{cc} V_q^T g_r & V_q^T g_\gamma \end{array}\right] \tag{7.13}$$

with

$$\begin{aligned} g_r &= (M+I)v + M_x v \Delta x_r + M_T v \Delta T_r,\\ g_\gamma &= M_\gamma v + M_x v \Delta x_\gamma + M_T v \Delta T_\gamma. \end{aligned} \tag{7.14}$$

After substituting (7.9), (7.11) and (7.12) in the last $p+1$ equations of (7.6), we obtain the system

$$\left[\begin{array}{cc} V_p^T M V_p + I_p & V_p^T \left(g_\gamma + M \Delta v_{q,\gamma}\right) \\ l^T V_p & l^T \Delta v_{q,\gamma} \end{array}\right]\left[\begin{array}{c} \Delta \bar{v}_p \\ \Delta\gamma \end{array}\right] = -\left[\begin{array}{c} V_p^T \left(g_r + M \Delta v_{q,r}\right) \\ l^T (v + \Delta v_{q,r}) - 1 \end{array}\right]. \tag{7.15}$$

From this system, we can easily compute $\Delta \bar{v}_p$ and $\Delta\gamma$ (e.g., using Gauss elimination with partial pivoting). Finally we set

$$\begin{aligned} \Delta x(0) &= \Delta x_r + \Delta\gamma \Delta x_\gamma,\\ \Delta T &= \Delta T_r + \Delta T_\gamma,\\ \Delta v &= \Delta v_{q,r} + \Delta\gamma \Delta v_{q,\gamma} + V_p \Delta \bar{v}_p. \end{aligned}$$

The Q-systems (7.8) and (7.13) are solved using Picard iteration. To avoid the need for the basis V_q, we immediately compute the N-dimensional updates $\Delta x_{q,*} = V_q \Delta \bar{x}_{q,*}$ and $\Delta v_{q,*} = V_q \Delta \bar{v}_{q,*}$. For $\Delta x_{q,*}$, we use algorithm 3.1. To compute $\Delta v_{q,*}$, we modified this algorithm slightly to use the Picard iteration

$$\Delta v_{q,*} \leftarrow -QM \Delta v_{q,*} - Q g_*.$$

It is also possible to generalize the GMRES method.

It is clear that this algorithm should work provided the basis is good enough, $V_q^T b_T$ (this term was neglected) is sufficiently small and the Q-systems (7.8) and (7.13) are solved with enough accuracy. The algorithm can be modified to take the term $V_q^T b_T$ into account. It remains to show that the linear systems (7.8), (7.10), (7.13) and (7.15) all have nonsingular matrices if (7.6) is nonsingular.

7.1.2 Mathematical motivation

Theorem 1.11 states the conditions for a non-degenerate simple period doubling point. In this section, we show that under these conditions, the extended system (7.1) has a nonsingular Jacobian matrix at the solution point and we also show that the subsystems (7.8), (7.10), (7.13) and (7.15) are all nonsingular if the exact solutions of the first three of these systems are used to construct (7.15).

Conditions for the nonsingularity of the Jacobian matrix of the extended system (7.1) are given by lemma 7.2. We will omit the $*$'s from our notation in the following two lemmas since all quantities are computed at the period doubling point.

Lemma 7.2 *Suppose the conditions from theorem 1.11 hold, i.e., $(x(0), T, \gamma)$ is a period doubling point, 1 and -1 are algebraically simple eigenvalues of $M(x(0), T, \gamma) = \varphi_x$, and the phase condition satisfies $c_s^T b_T \neq 0$. Suppose that the vector l is not orthogonal to the eigenspace of M for the eigenvalue -1. Suppose w_{-1} is a left eigenvector of M for the eigenvalue -1, and suppose v_{-1} is the corresponding right eigenvector that satisfies $l^T v_{-1} = 1$. Suppose*

$$k = w_{-1}^T \left[M_\gamma v_{-1} - M_{(x,T)} v_{-1} \begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}^{-1} \begin{bmatrix} b_\gamma \\ d_{s,\gamma} \end{bmatrix} \right] = w_{-1}^T g_\gamma \neq 0,$$

i.e., the eigenvalue -1 crosses the unit circle with nonzero speed. In this formula, g_γ is given by (7.14) and it is assumed that the exact solutions to (7.8) and (7.10) at the period doubling point are used to construct g_γ. Then:

$$J = \begin{bmatrix} M-I & b_T & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & d_{s,\gamma} \\ M_x v_{-1} & M_T v_{-1} & M+I & M_\gamma v_{-1} \\ 0 & 0 & l^T & 0 \end{bmatrix}$$

is nonsingular.

Proof. $(x(0), T, v_{-1}, \gamma)$ solves (7.1).

Since 1 is an algebraically simple eigenvalue of M, $b_T \notin \text{Range}(M-I)$. Furthermore we assumed that $c_s^T b_T \neq 0$. Hence

$$\begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}$$

is nonsingular (lemma 1.12). According to lemma 2.8 from [63], J is nonsingular <u>iff</u>

$$\begin{aligned} J_c &= \begin{bmatrix} M+I & M_\gamma v_{-1} \\ l^T & 0 \end{bmatrix} - \begin{bmatrix} M_x v_{-1} & M_T v_{-1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}^{-1} \begin{bmatrix} 0_{N\times p} & b_\gamma \\ 0_{1\times p} & d_{s,\gamma} \end{bmatrix} \\ &= \begin{bmatrix} M+I & M_\gamma v_{-1} - M_{(x,T)} v_{-1} \begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}^{-1} \begin{bmatrix} b_\gamma \\ d_{s,\gamma} \end{bmatrix} \\ l^T & 0 \end{bmatrix} \\ &= \begin{bmatrix} M+I & g_\gamma \\ l^T & 0 \end{bmatrix} \end{aligned}$$

is nonsingular. -1 is a simple eigenvalue of M and $\text{Rank}(M + I) = N - 1$. The transversality condition $k = w_{-1}^T g_\gamma \neq 0$ implies that $g_\gamma \notin \text{Range}(M + I)$. Hence $\text{Rank}\begin{bmatrix} M+I & g_\gamma \end{bmatrix} = N$. Since $l^T v_{-1} \neq 0$, $l \notin \text{Range}(M^T + I)$ and $\text{Rank}(J_c) = N + 1$. Hence J_c and J are nonsingular. □

We will now show that under the conditions of the above lemma, the subsystems (7.8), (7.10), (7.13) and (7.15) have nonsingular matrices.

Lemma 7.3 *Suppose the conditions from lemma 7.2 hold. Furthermore suppose the subsystems (7.8), (7.10) and (7.13) are solved exactly. Then at the period doubling point, the matrices*

$$V_q^T M V_q - I_q, \tag{7.16}$$

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix}, \tag{7.17}$$

$$V_q^T M V_q + I_q \tag{7.18}$$

and

$$\begin{bmatrix} V_p^T M V_p + I_q & V_p^T \left(g_\gamma + M\Delta v_{q,\gamma}\right) \\ l^T & l^T \Delta v_{q,\gamma} \end{bmatrix} \tag{7.19}$$

with g_γ given by (7.14) and $\Delta v_{q,\gamma}$ given by (7.13) are nonsingular.

Proof. The matrix

$$M_1 = \begin{bmatrix} M - I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}$$

is the matrix of the linearized single shooting system and is nonsingular under the conditions of the theorem. Furthermore,

$$\tilde{V} = \begin{bmatrix} V_q & V_p & 0_{N\times 1} \\ 0_{1\times q} & 0_{1\times p} & 1 \end{bmatrix}$$

is an orthogonal matrix, so

$$\tilde{V}^T M_1 \tilde{V} = \begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_q & c_s^T V_p & d_{s,T} \end{bmatrix}$$

is nonsingular and thus also (7.16) and (7.17).

Since $r_\sigma(V_q^T M V_q) < \rho < 1$, all eigenvalues of (7.18) lie in a circle with radius $\rho < 1$ around 1, so (7.18) is also a nonsingular matrix.

The nonsingularity of (7.19) can be shown as follows. From the proof of lemma 7.2 follows that the matrix

$$J_c = \begin{bmatrix} M + I & g_\gamma \\ d^T & 0 \end{bmatrix}$$

is nonsingular. Hence

$$\tilde{V}^T J_c \tilde{V} = \begin{bmatrix} V_q^T M V_q + I_q & 0 & V_q^T g_\gamma \\ V_p^T M V_q & V_p^T M V_p + I_p & V_p^T g_\gamma \\ l^T V_q & l^T V_p & 0 \end{bmatrix}$$

is also a nonsingular matrix. From lemma 2.5, (7.13) and the nonsingularity of (7.18) follows that

$$\begin{aligned} J_v &= \begin{bmatrix} V_p^T M V_p + I_p & V_p^T g_\gamma \\ l^T V_p & 0 \end{bmatrix} - \begin{bmatrix} V_p^T M V_q \\ l^T V_q \end{bmatrix} \left(V_q^T M V_q + I_q\right)^{-1} \begin{bmatrix} 0_{q\times p} & V_q^T g_\gamma \end{bmatrix} \\ &= \begin{bmatrix} V_p^T M V_p + I_p & V_p^T (g_\gamma + M\Delta v_{q,\gamma}) \\ l^T V_p & l^T \Delta v_{q,\gamma} \end{bmatrix} \end{aligned}$$

with

$$\Delta v_{q,\gamma} = V_q \left(V_q^T M V_q - I_q\right)^{-1} V_q^T g_\gamma$$

is nonsingular. J_v is just the matrix of the linear system (7.19). □

The nonsingularity is only proved at the period doubling point and under the assumption that (7.8), (7.10) and (7.13) are solved exactly. However, nonsingular matrices remain nonsingular under small perturbations. We have good confidence that the actual subsystems will also be nonsingular if enough Picard iterations are performed to solve the Q-systems and if a good starting value is used. This is confirmed by our test results.

7.1.3 Implementation aspects

The implementation aspects are not so different from our single shooting method for periodic solutions. The basis is computed in precisely the same way. The implementation of the Q-system solvers is also done in the same way. To solve the P-system, we used Gauss elimination with pivoting. Least-squares methods can also be used. As we have seen before, it is better not to omit the phase condition in that case. We have not yet done any experiments with least-squares methods to solve the P-systems in the context of the computation of period doubling points.

A new element in the implementation is the computation of the terms $M_\gamma v$, $M_T v$, $M_x v \Delta x_r$ and $M_x v \Delta x_\gamma$. These terms can be computed using finite differences, e.g.,

$$\begin{aligned} M_x v \Delta x_* &= \varphi_{xx} v \Delta x_* \approx \frac{\begin{array}{c}\varphi(x(0)+h_1 v + h_2 \Delta x, T, \gamma) + \varphi(x(0), T, \gamma) \\ -\varphi(x(0)+h_2\Delta x, T, \gamma) - \varphi(x(0)+h_1 v, T, \gamma)\end{array}}{h_1 h_2} \\ M_\gamma v &= \varphi_{x\gamma} v \approx \\ &\frac{\varphi(x(0)+h_1 v, T, \gamma+h_3) + \varphi(x(0), T, \gamma) - \varphi(x(0), T, \gamma+h_3) - \varphi(x(0)+h_1 v, T, \gamma)}{h_1 h_3}, \\ M_T v &= \varphi_{xT} v \approx \frac{f(\varphi(x(0),T,\gamma) + h_4 M v, \gamma) - f(\varphi(x(0),T,\gamma),\gamma)}{h_4}. \end{aligned} \tag{7.20}$$

In our tests, we experienced that the accuracy of these terms has a big influence on the convergence speed. If (7.20) is used, a good choice of the differentiation stepsizes h_1, h_2, h_3 and h_4 is very important. We have not yet tried to use higher-order formulas or to derive variational equations, although this is certainly worth to be done.

If first-order finite differences are used to compute b_γ and for the matrix–vector products with the monodromy matrix, if (7.20) is used and if all Q-systems are solved using

l Picard steps, our method needs $4l + 8$ time integrations besides the time integrations for the construction of the basis:

- One time integration is needed for the evaluation of $\varphi(x(0), T, \gamma)$.
- One additional time integration is required for the computation of Mv: $\varphi(x(0) + h_1 v, T, \gamma)$.
- One additional time integration is used for the computation of b_γ: $\varphi(x(0), T, \gamma + h_3)$.
- One additional time integration is needed for the computation of $M_\gamma v$: $\varphi(x(0) + h_1 v, T, \gamma + h_3)$. The other three terms have already been computed.
- The computation of $M_x v \Delta x_r$ and $M_x v \Delta x_\gamma$ requires 2×2 additional time integrations: $\varphi(x(0) + h_2 \Delta x_*, T, \gamma)$ and $\varphi(x(0) + h_1 v + h_2 \Delta x_*, T, \gamma)$. The other terms have already been computed.
- $4 \times l$ time integrations are done in the Picard iterations.

This contrasts to $N + 8$ time integrations for the full Newton method: the eight time integrations discussed above—we can compute Mv using M but need to compute $\varphi(x(0) + h_1 v, T, \gamma)$ anyway to compute $M_\gamma v$—and N additional time integrations to construct M.

7.1.4 Test results

As a test of the method, we computed some period doubling points in the model (3.40-3.41) discussed in section 3.8 (although with a coarser discretization than for the results in section 3.8). As a starting point for the computations, we used a point on the branch of periodic solutions or a perturbed point. We used the Schur vector for the eigenvalue closest to -1 as a start value for the vector v. Figure 7.1 shows the convergence behaviour for a period doubling point with largest Floquet multipliers

$$\begin{aligned} &\mu_1 = +1.000, \\ &\mu_2 = -1.000, \\ &\mu_{3,4} = +0.173 \pm 0.354i, \\ &\mu_5 = +0.158. \end{aligned}$$

15 discretization points were used. At the start point, the Floquet multiplier closest to -1 was around -0.9. Only two vectors were used in the basis V_p. Hence the expected asymptotic convergence rate of the Picard scheme is around 0.4, which corresponds to the curves for $l = 1$ in Figure 7.1. In this case, the convergence speed of the Picard iteration was the limiting factor for the overall convergence speed. If five Picard steps are used, it is clear that other factors limit the overall convergence speed since we notice little or no difference with four Picard steps. Note that at convergence,

$$\begin{aligned} &\|v\|_2 \approx \|v_p\|_2 \approx 1.0133, \\ &\|v_q\|_2 \approx 1.2\ 10^{-6}, \\ &\|(M + I)v\|_2 \approx 5.0\ 10^{-14}, \\ &\|(M + I)v_p\|_2 \approx 7.8\ 10^{-6}. \end{aligned}$$

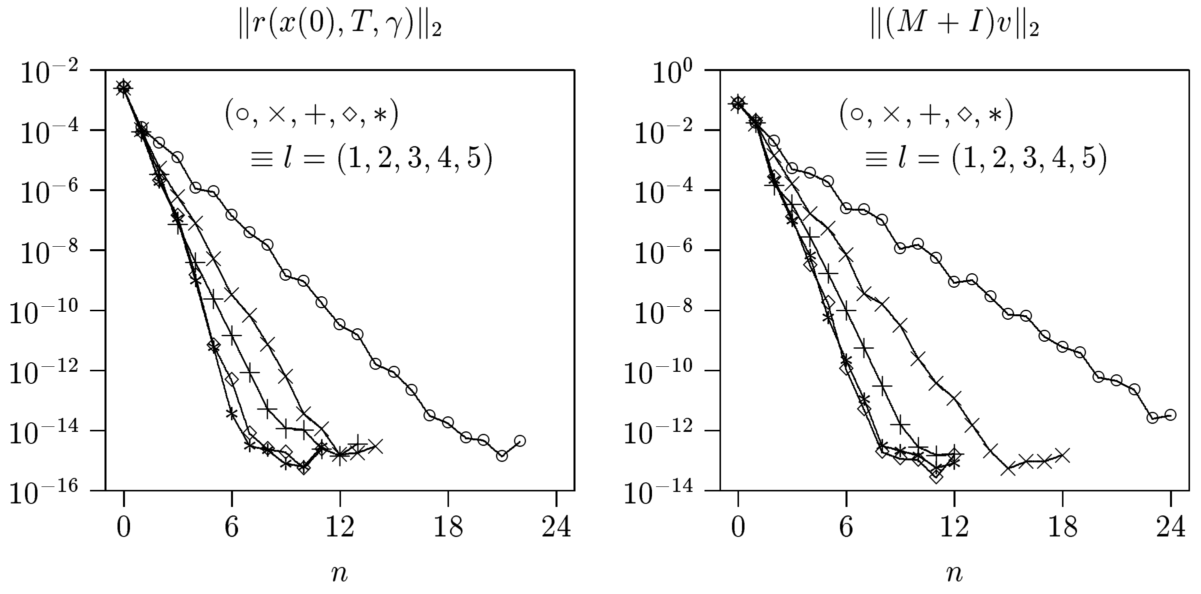


Figure 7.1: Convergence evolution for different numbers of Picard steps per Newton-Picard step. 2 basis vectors were used. $|\mu_3| \approx 0.4$.

Since the basis is not a perfect basis, v also has some small components in the space $\mathcal{U}^\perp$. In fact, v is a better approximation to the eigenvector for the eigenvalue -1 than the approximation that can be computed from the projected matrix (this is not necessarily v_p!). The accuracy that can be obtained with our method is independent of the accuracy of the basis. The latter only determines the convergence speed of the Newton–Picard procedure.

As we already mentioned earlier, the accuracy to which the terms $M_x v \Delta x_*$, $M_\gamma v$ and $M_T v$ are computed has a large influence on the convergence speed. Some fine-tuning of the differentiation stepsize h_i in (7.20) was needed to produce these results.

7.2 Torus bifurcation points

To compute torus bifurcation points using single shooting, we suggest the use of (1.65) with the normalization conditions (1.66). Hence we have to solve the nonlinear system

$$\left\{\begin{array}{l} r(x(0),T,\gamma) = \varphi(x(0),T,\gamma) - x(0) = 0, \\ s(x(0),T,\gamma) = 0, \\ (M - \cos(\theta) I)\, v + \sin(\theta) w = 0, \\ -\sin(\theta) v + (M - \cos(\theta) I)\, w = 0, \\ l_v^T v - 1 = 0, \\ l_w^T w = 0. \end{array}\right. \tag{7.21}$$

This is a system of $3N+3$ equations in the three N-dimensional vector unknowns $x(0)$, v and w and the three scalar unknowns T, γ and θ. At the solutions, $v + iw$ is an eigenvector for the Floquet multiplier $e^{i\theta}$ and $v - iw$ for the eigenvalue $e^{-i\theta}$.

Newton's method results in the repeated solution of linear systems

$$\begin{bmatrix} M-I & b_T & 0 & 0 & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & 0 & 0 & d_{s,\gamma} \\ M_x v & M_T v & M-cI & sI & sv+cw & M_\gamma v \\ M_x w & M_T w & -sI & M-cI & -cv+sw & M_\gamma w \\ 0 & 0 & l_v^T & 0 & 0 & 0 \\ 0 & 0 & 0 & l_w^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta v \\ \Delta w \\ \Delta\theta \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} r(x(0),T,\gamma) \\ s(x(0),T,\gamma) \\ (M-cI)\,v+sw \\ -sv+(M-cI)\,w \\ l_v^T v-1 \\ l_w^T w \end{bmatrix}, \tag{7.22}$$

where $c=\cos(\theta)$ and $s=\sin(\theta)$. The derivation of the Newton–Picard method is similar to the period doubling case. We will only stress the differences. The Newton–Picard method starts from a slight generalization of assumption 2.1 similar to assumption 7.1. Bases V_p and V_q and projectors P and Q are constructed as before. We set

$$\begin{array}{l} \Delta x(0)=V_p\Delta\bar{x}_p+V_q\Delta\bar{x}_q,\ \Delta x_p=V_p\Delta\bar{x}_p=P\Delta x(0),\ \Delta x_q=V_q\Delta\bar{x}_q=Q\Delta x(0), \\ \Delta v=V_p\Delta\bar{v}_p+V_q\Delta\bar{v}_q,\ \Delta v_p=V_p\Delta\bar{v}_p=P\Delta v,\ \Delta v_q=V_q\Delta\bar{v}_q=Q\Delta v, \\ \Delta w=V_p\Delta\bar{w}_p+V_q\Delta\bar{w}_q,\ \Delta w_p=V_p\Delta\bar{w}_p=P\Delta w,\ \Delta w_q=V_q\Delta\bar{w}_q=Q\Delta w. \end{array} \tag{7.23}$$

(7.23) is substituted in (7.22) and the N-dimensional blockrows are premultiplied with $[V_q\ \ V_p]^T$. The unknowns are ordered as $\Delta\bar{x}_q$, $\Delta\bar{x}_p$, ΔT, $\Delta\bar{v}_q$, $\Delta\bar{w}_q$, $\Delta\bar{v}_p$, $\Delta\bar{w}_p$, $\Delta\theta$ and $\Delta\gamma$. The subblock

$$\begin{bmatrix} M-cI & sI & sv+cw \\ -sI & M-cI & -cv+sw \end{bmatrix}$$

becomes

$$\begin{bmatrix} V_q^TMV_q-cI_q & sI_q & 0 & 0 & sV_q^Tv+cV_q^Tw \\ -sI_q & V_q^TMV_q-cI_q & 0 & 0 & -cV_q^Tv+sV_q^Tw \\ V_p^TMV_q & 0 & V_p^TMV_p-cI_p & sI_p & sV_p^Tv+cV_p^Tw \\ 0 & V_p^TMV_q & -sI_p & V_p^TMV_p-cI_p & -cV_p^Tv+sV_p^Tw \end{bmatrix}. \tag{7.24}$$

At the solution and with a perfect basis, $V_q^Tv=0$ and $V_q^Tw=0$. If the starting values for v and w are chosen carefully, V_q^Tv and V_q^Tw will be small and it will often be possible to neglect these terms. However, a truly robust method should take these terms into account.

The solution procedure proceeds as in the period doubling case. We set (7.7) and solve (7.8). Next we set (7.9), solve (7.10) and compute (7.11). After substituting (7.9) and (7.11) in the next $2q$ equations of the projected system, we get

$$\begin{bmatrix} V_q^TMV_q-cI_q & sI_q \\ -sI_q & V_q^TMV_q-cI_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_q \\ \Delta\bar{w}_q \end{bmatrix} = -\tilde{V}_q^T(g_r+\Delta\theta g_\theta+\Delta\gamma g_\gamma)$$

with

$$\begin{aligned} g_r &= \begin{bmatrix} (M-cI)v+sw \\ -sv+(M-cI)w \end{bmatrix} + \begin{bmatrix} M_x v \\ M_x w \end{bmatrix} \Delta x_r + \begin{bmatrix} M_T v \\ M_T w \end{bmatrix} \Delta T_r, \\ g_\theta &= \begin{bmatrix} sv+cw \\ -cv+sw \end{bmatrix}, \\ g_\gamma &= \begin{bmatrix} M_\gamma v \\ M_\gamma w \end{bmatrix} + \begin{bmatrix} M_x v \\ M_x w \end{bmatrix} \Delta x_\gamma + \begin{bmatrix} M_T v \\ M_T w \end{bmatrix} \Delta T_\gamma \end{aligned}$$

and

$$\tilde{V}_q = \begin{bmatrix} V_q & 0 \\ 0 & V_q \end{bmatrix}, \ \tilde{V}_p = \begin{bmatrix} V_p & 0 \\ 0 & V_p \end{bmatrix}.$$

Hence we set

$$\begin{aligned} \Delta\bar{v}_q &= \Delta\bar{v}_{q,r} + \Delta\theta\Delta\bar{v}_{q,\theta} + \Delta\gamma\Delta\bar{v}_{q,\gamma}, \\ \Delta\bar{w}_q &= \Delta\bar{w}_{q,r} + \Delta\theta\Delta\bar{w}_{q,\theta} + \Delta\gamma\Delta\bar{w}_{q,\gamma}, \end{aligned}$$

and solve $\Delta\bar{v}_{q,*}$ and $\Delta\bar{w}_{q,*}$ from

$$\begin{bmatrix} V_q^T M V_q - cI_q & sI_q \\ -sI_q & V_q^T M V_q - cI_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_{q,r} & \Delta\bar{v}_{q,\theta} & \Delta\bar{v}_{q,\gamma} \\ \Delta\bar{w}_{q,r} & \Delta\bar{w}_{q,\theta} & \Delta\bar{w}_{q,\gamma} \end{bmatrix} = -\tilde{V}_q^T \begin{bmatrix} g_r & g_\theta & g_\gamma \end{bmatrix}. \tag{7.25}$$

Note that in practise, one can often set $\Delta\bar{v}_{q,\theta} = \Delta\bar{w}_{q,\theta} = 0$. After solving (7.25), we construct and solve the P-system

$$\begin{bmatrix} V_p^T M V_p - cI_p & sI_p & \tilde{V}_p^T \hat{g}_\theta & \tilde{V}_p^T \hat{g}_\gamma \\ -sI_p & V_p^T M V_p - cI_p & & \\ l_v^T V_p & 0 & l_v^T \Delta v_{q,\theta} & l_v^T \Delta v_{q,\gamma} \\ 0 & l_w^T V_p & l_w^T \Delta w_{q,\theta} & l_w^T \Delta w_{q,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_p \\ \Delta\bar{v}_q \\ \Delta\theta \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \tilde{V}_p^T \hat{g}_r \\ l_v^T(v+\Delta v_{q,r}) - 1 \\ l_w^T(w+\Delta w_{q,r}) \end{bmatrix} \tag{7.26}$$

with

$$\hat{g}_* = g_* + \begin{bmatrix} M-cI & sI \\ -sI & M-cI \end{bmatrix} \begin{bmatrix} \Delta v_{q,*} \\ \Delta w_{q,*} \end{bmatrix}.$$

The system (7.26) is a small $(2p+2)\times(2p+2)$ system and can be solved using a direct method. System (7.25) however is a large $2q \times 2q$ system. We can derive a Picard scheme to solve this system as follows: (7.25) is equivalent to

$$\begin{bmatrix} cI_q & -sI_q \\ sI_q & cI_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_{q,*} \\ \Delta\bar{w}_{q,*} \end{bmatrix} = \begin{bmatrix} V_q^T M V_q & 0 \\ 0 & V_q^T M V_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_{q,*} \\ \Delta\bar{w}_{q,*} \end{bmatrix} + \tilde{V}_q^T g_*.$$

Hence we suggest to use the iteration

$$\begin{aligned} \begin{bmatrix} \Delta\bar{v}_{q,*} \\ \Delta\bar{w}_{q,*} \end{bmatrix} &\leftarrow \begin{bmatrix} cI_q & sI_q \\ -sI_q & cI_q \end{bmatrix} \left(\begin{bmatrix} V_q^T M V_q & 0 \\ 0 & V_q^T M V_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_{q,*} \\ \Delta\bar{w}_{q,*} \end{bmatrix} + \tilde{V}_q^T g_* \right) \\ &= \begin{bmatrix} cV_q^T M V_q & sV_q^T M V_q \\ -sV_q^T M V_q & cV_q^T M V_q \end{bmatrix} \begin{bmatrix} \Delta\bar{v}_{q,*} \\ \Delta\bar{w}_{q,*} \end{bmatrix} + \begin{bmatrix} cV_q^T & sV_q^T \\ -sV_q^T & cV_q^T \end{bmatrix} g_*. \end{aligned} \tag{7.27}$$

To prove the contractivity of this scheme, we first state the following lemma.

Lemma 7.4 *Let $A \in \mathbb{C}^{N\times N}$. Suppose $AX = X\Lambda$ is the Jordan decomposition of A with $X, \Lambda \in \mathbb{C}^{N\times N}$. Suppose $c = \cos(\theta)$ and $s = \sin(\theta)$. Then*

$$\begin{bmatrix} cA & sA \\ -sA & cA \end{bmatrix} \begin{bmatrix} (s-ic)V & (s+ic)V \\ (c+is)V & (c-is)V \end{bmatrix} = \\ \begin{bmatrix} (s-ic)V & (s+ic)V \\ (c+is)V & (c-is)V \end{bmatrix} \begin{bmatrix} (c+is)\Lambda & 0 \\ 0 & (c-is)\Lambda \end{bmatrix}. \tag{7.28}$$

Proof. The proof of this lemma is simple. It is sufficient to compute the matrix products at the left-hand side and the right-hand side of (7.28), substitute AV with $V\Lambda$ and compare the results. □

If the matrix A is diagonalizable then (7.28) is the Jordan decomposition of

$$\begin{bmatrix} cA & sA \\ -sA & cA \end{bmatrix}. \tag{7.29}$$

Otherwise some further transformations are needed to transform the complex upper diagonal entries in

$$\begin{bmatrix} (c+is)\Lambda & 0 \\ 0 & (c-is)\Lambda \end{bmatrix}$$

to 1. In any case, if λ is an eigenvalue of A with algebraic multiplicity k, then $e^{i\theta}\lambda$ and $e^{-i\theta}\lambda$ are eigenvalues of (7.29) with algebraic multiplicity k. Hence we have

Corollary 7.5 *For the usual choice of the basis V_q, we have*

$$r_\sigma\left(\begin{bmatrix} cV_q^TMV_q & sV_q^TMV_q \\ -sV_q^TMV_q & cV_q^TMV_q \end{bmatrix}\right) = r_\sigma\left(V_q^TMV_q\right) = |\mu_{p+1}|.$$

Hence the Picard scheme (7.27) is contractive. This shows that the Newton–Picard method can also be extended to the computation of torus bifurcation points. Compared to the period doubling case, the main difficulty is the more complex structure of the Q-system (7.25) compared to (7.13). A new Picard iteration scheme is needed. Every Picard step with (7.27) takes twice as much work as in the period doubling case and there is also an additional right-hand side (although we expect that this right-hand side can often be neglected). We have not yet implemented the method. However, we do not expect any particular new problems.

7.3 Fold point

At a fold point, the monodromy matrix has an algebraically double eigenvalue 1. Generically, the geometric multiplicity is one, but in rare cases, two linearly independent eigenvectors exist. To compute a fold point using single shooting, we propose to use the extended system

$$\left\{\begin{array}{l} r(x(0),T,\gamma) = \varphi(x(0),T,\gamma) - x(0) = 0, \\ s(x(0),T,\gamma) = 0, \\ (M(x(0),T,\gamma) - I)v + \alpha f(\varphi(x(0),T,\gamma),\gamma) = 0, \\ l_1^T v = 0, \\ l_2^T v - 1 = 0. \end{array}\right. \tag{7.30}$$

Alternatively, one can use $f(x(0),\gamma)$ instead of $f(\varphi(x(0),T,\gamma),\gamma)$. Both choices are equivalent at the solution. System (7.30) relies on the fact that $f(x(0),\gamma)$ is an eigenvector of $M(x(0),T,\gamma)$ at a periodic solution for the trivial Floquet multiplier 1. Note that this is only approximately true after time discretization, but it is usual practise to not introduce another set of equations that define the conditions for the eigenvector for the trivial Floquet multiplier 1. This extended system is similar to the system used in AUTO94. In (7.30), l_1 should not be orthogonal to $f(x(0),\gamma)$. This condition is used to assure that v is a second eigenvector linearly independent of $f(x(0),\gamma)$ or a generalized eigenvector and does not lie in the direction of $f(x(0),\gamma)$. $l_1 = f(x(0),\gamma)$ is a good choice, but if one wishes to avoid a nonlinear condition, one can evaluate $f(x(0),\gamma)$ in the start point and use the result for the value of l_1. l_2 should not be parallel to l_1. It is a good idea to choose l_2 orthogonal to l_1. If any approximation to a second eigenvector or a generalized eigenvector is available, one can use this approximation to construct the vector l_2. An alternative to the condition $l_2^T v = 1$ is the nonlinear condition $v^T v = 1$.

Newton's method applied to (7.30) results in the repetitive solution of linear systems

$$\begin{bmatrix} M-I & b_T & 0 & 0 & b_\gamma \\ c_s^T & d_{s,T} & 0 & 0 & d_{s,\gamma} \\ M_x v & M_T v & M-I & b_T & M_\gamma v \\ 0 & 0 & l_1^T & 0 & 0 \\ 0 & 0 & l_2^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \\ \Delta v \\ \Delta \alpha \\ \Delta \gamma \end{bmatrix} = - \begin{bmatrix} r \\ s \\ (M-I)v + \alpha f \\ l_1^T v \\ l_2^T v - 1 \end{bmatrix}, \tag{7.31}$$

where $f = f(\varphi(x(0),T,\gamma),\gamma)$. Note that at the solution point, the matrix

$$\begin{bmatrix} M-I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}$$

has one zero eigenvalue and

$$\begin{bmatrix} v + \dfrac{c_s^T v + \alpha\, d_{s,T}}{c_s^T b_T} b_T \\ \alpha \end{bmatrix}$$

is the corresponding eigenvector. Hence we cannot use a variant of (7.3)-(7.4) to solve (7.31). Several solutions have been proposed in the literature to solve such systems while exploiting the structure, see, e.g., [60, 83]. However, not all of these methods are easily combined with the Newton–Picard procedure.

We suggest the following method. Suppose $A \in \mathbb{R}^{N\times N}$ and $b_1 \in \mathbb{R}^N$. Suppose A is nonsingular or A has an eigenvalue 0 of geometric multiplicity one and $b_1 \notin \text{Range}(A)$.

Let $B \in \mathbb{R}^{N\times N}$, r_1, r_2, b_2, $c_2 \in \mathbb{R}^N$ and $n \in \mathbb{R}$. Consider the linear system

$$\begin{bmatrix} A & 0 & b_1 \\ B & A & b_2 \\ 0 & c_2^T & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ \gamma \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ n \end{bmatrix}. \tag{7.32}$$

Suppose the matrix in (7.32) is nonsingular. This implies some additional conditions which depend on whether A is nonsingular or not. (7.32) has the same structure as (7.31) and corresponds to the linearized system at or near a non-degenerate fold point if A is (nearly) singular. One can solve (7.32) as follows.

1. First compute a nonzero solution (x_β, γ_β) to

$$\begin{bmatrix} A & b_1 \end{bmatrix} \begin{bmatrix} x_\beta \\ \gamma_\beta \end{bmatrix} = 0_{N\times 1} \tag{7.33}$$

Note that if A is nonsingular or if A has a zero eigenvalue of geometric multiplicity one and $b_1 \notin \text{Range}(A)$,

$$\text{Rank}(\begin{bmatrix} A & b_1 \end{bmatrix}) = N.$$

The normalized solution to (7.33) is the last right singular vector of $[A \;\; b_1]$. If A is singular then $x_\beta = v_0$, $\gamma_\beta = 0$ with v_0 an eigenvector for the zero eigenvalue satisfies (7.33). There are several options to compute a nonzero solution to (7.33). For instance, one can compute the singular value decomposition of $[A \;\; b_1]$ and use the last right singular vector as a nonzero solution $(x_\beta^T, \gamma_\beta)$ to (7.33). This is an expensive method, but numerically very stable. Another option is adding a random normalization condition

$$l_x^T x_\beta + l_\gamma \gamma_\beta = 1$$

to (7.33). Note that this condition will fail if the two directions $[l_x^T \;\; l_\gamma]^T$ and $[x_\beta^T \;\; \gamma_\beta]^T$ are orthogonal.

2. Next compute a vector $x_r \in \mathbb{R}^N$ and a scalar γ_r that solve

$$\begin{bmatrix} A & b_1 \end{bmatrix} \begin{bmatrix} x_r \\ \gamma_r \end{bmatrix} = -r_1. \tag{7.34}$$

(7.34) has a one-parameter family of solutions from which we pick an arbitrary solution, e.g., the least-squares solution. This solution is obtained if we add the condition

$$x_\beta^T x_r + \gamma_\beta \gamma_r = 0$$

to the system since the least-squares solution is orthogonal to the last singular vector (this follows from (3.29)).

Note that if we set

$$\begin{aligned} x &= x_r + \beta x_\beta, \\ \gamma &= \gamma_r + \beta \gamma_\beta, \end{aligned} \tag{7.35}$$

then

$$\begin{bmatrix} A & b_1 \end{bmatrix} \begin{bmatrix} x \\ \gamma \end{bmatrix} = -r_1.$$

$\{(x,\gamma) \mid x = x_r + \beta x_\beta,\ \gamma = \gamma_r + \beta\gamma_\beta\}$ is the one-parameter family of solutions to (7.34). For one particular value of β, it is possible to also satisfy the other equations of (7.32).

3. To determine v and β, we substitute (7.35) in the last $N+1$ equations of (7.32). After some rewriting, we obtain

$$\begin{bmatrix} A & b_2\gamma_\beta + Bx_\beta \\ c_2^T & 0 \end{bmatrix} \begin{bmatrix} v \\ \beta \end{bmatrix} = - \begin{bmatrix} r_2 + b_2\gamma_r + Bx_r \\ n \end{bmatrix}. \tag{7.36}$$

It is clear from the procedure that if (7.32) is a nonsingular system, system (7.36) must have a unique solution. Consider the case of a singular A. A necessary condition for the nonsingularity of the matrix in (7.36) is

$$b_2\gamma_\beta + Bx_\beta \notin \text{Range}(A), \tag{7.37}$$

or equivalently,

$$\text{Rank}\left(\begin{bmatrix} A & b_2\gamma_\beta + Bx_\beta \end{bmatrix}\right) = N.$$

Suppose ψ_0 is a left eigenvector of A and v_0 a right eigenvector for the zero eigenvalue. Then $x_\beta = v_0$ and $\gamma_\beta = 0$ is a solution to (7.33) and the condition (7.37) is equivalent to

$$\psi_0^T\left(b_2\gamma_\beta + Bx_\beta\right) \neq 0 \Leftrightarrow \psi_0^T B v_0 \neq 0. \tag{7.38}$$

If (7.32) results from the computation of a fold point on a steady-state solution branch of (1.4), $B = f_{xx}v_0$ at the fold point and (7.38) is precisely the nondegeneracy condition.

After solving (7.36), we can compute x and γ using (7.35).

This procedure works if A is nonsingular and if A has a geometrically simple eigenvalue 0 with $b_1 \notin \text{Range}(A)$. This is important since at the fold point, A is singular and $b_1 \notin \text{Range}(A)$, but during the Newton iterations, A is nonsingular (although close to singular). The procedure is well-defined in both cases and solves (7.32) accurately in both cases.

We have implemented this procedure in Matlab version 5 and tested the routine with random systems of the form (7.32) (both with nonsingular and singular matrices A). The results were comparable with the results produced by the Matlab version 5 *LU* decomposition routine applied to the system (7.32) in the sense that the residuals were of comparable size.

It is very easy to apply the Newton–Picard idea to this procedure. We start from an assumption similar to assumption 7.1, construct bases V_p and V_q and projectors P and Q as before and set

$$\begin{aligned} &\Delta x(0) = V_p\Delta\bar{x}_p + V_q\Delta\bar{x}_q,\ \Delta x_p = V_p\Delta\bar{x}_p = P\Delta x(0),\ \Delta x_q = V_q\Delta\bar{x}_q = Q\Delta x(0), \\ &\Delta v = V_p\Delta\bar{v}_p + V_q\Delta\bar{v}_q,\ \Delta v_p = V_p\Delta\bar{v}_p = P\Delta v,\ \Delta v_q = V_q\Delta\bar{v}_q = Q\Delta v, \end{aligned}$$

We substitute these equations in (7.31) and premultiply the N-dimensional blockrows with $[V_q\ \ V_p]^T$. We again $V_q^T b_T$.

To solve the resulting system, we first have to find a nonzero solution to

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \end{bmatrix} \begin{bmatrix} \Delta \bar{x}_{q,\beta} \\ \Delta \bar{x}_{p,\beta} \\ \Delta T_\beta \\ \Delta \gamma_\beta \end{bmatrix} = 0_{(N+1)\times 1}. \tag{7.39}$$

Therefore, we first construct

$$\Delta \bar{x}_{q,\beta,\gamma} = -\left(V_q^T M V_q - I_q\right)^{-1} V_q^T b_\gamma \tag{7.40}$$

and then compute a nonzero solution to

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T (b_\gamma + M \Delta x_{q,\beta,\gamma}) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} + c_s^T \Delta x_{q,\beta,\gamma} \end{bmatrix} \begin{bmatrix} \Delta \bar{x}_{p,\beta} \\ \Delta T_\beta \\ \Delta \gamma_\beta \end{bmatrix} = 0_{(p+1)\times 1}. \tag{7.41}$$

It is possible to show (using lemma 2.6) that the rank of the matrix in (7.41) is $p+1$ if the phase condition is well chosen and if the fold point is a non-degenerate one. To compute $\Delta \bar{x}_{p,\beta}$, ΔT_β and $\Delta \gamma_\beta$, we can compute the last right singular vector of the matrix in (7.41) or add a random normalization condition. Finally,

$$\Delta x_\beta = V_p \Delta \bar{x}_{p,\beta} + \Delta \gamma_\beta V_q \Delta \bar{x}_{q,\beta,\gamma}.$$

One can normalize the solution $(\Delta x_\beta, \Delta T_\beta, \Delta \gamma_\beta)$, but this is not strictly necessary.

Next we have to compute a particular solution to

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \end{bmatrix} \begin{bmatrix} \Delta \bar{x}_{q,r} \\ \Delta \bar{x}_{p,r} \\ \Delta T_r \\ \Delta \gamma_r \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \end{bmatrix}. \tag{7.42}$$

We suggest to add the condition

$$\Delta x_\beta^T (V_q \Delta \bar{x}_{q,r} + V_p \Delta \bar{x}_{p,r}) + \Delta T_\beta \Delta T_r + \Delta \gamma_\beta \Delta \gamma_r = 0 \tag{7.43}$$

to (7.42). This results in the least-squares solution of (7.42). (7.42-7.43) can be solved by first solving

$$(V_q^T M V_q - I_q) \Delta \bar{x}_{q,r,r} = -V_q^T r, \tag{7.44}$$

then solving

$$\begin{gathered} \begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T (b_\gamma + M \Delta x_{q,\beta,\gamma}) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} + c_s^T \Delta x_{q,\beta,\gamma} \\ \Delta x_\beta^T V_p & \Delta T_\beta & \Delta \gamma_\beta + \Delta x_\beta^T \Delta x_{q,\beta,\gamma} \end{bmatrix} \begin{bmatrix} \Delta \bar{x}_{p,r} \\ \Delta T_r \\ \Delta \gamma_r \end{bmatrix} = \\ - \begin{bmatrix} V_p^T (r + M \Delta x_{q,r,r}) \\ s + c_s^T \Delta x_{q,r,r} \\ \Delta x_\beta^T \Delta x_{q,r,r} \end{bmatrix}, \end{gathered} \tag{7.45}$$

and finally setting

$$\Delta x_r = V_p \Delta \bar{x}_{p,r} + V_q(\Delta \bar{x}_{q,r,r} + \Delta\gamma_r \Delta \bar{x}_{q,\beta,\gamma}).$$

Now we have

$$\begin{aligned} \Delta x(0) &= \Delta x_r + \beta \Delta x_\beta, \\ \Delta T &= \Delta T_r + \beta \Delta T_\beta, \\ \Delta \gamma &= \Delta \gamma_r + \beta \Delta \gamma_\beta, \end{aligned}$$

and we still have to determine Δv, $\Delta\alpha$ and β. These unknowns are solved from the system

$$\begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & V_q^T g_\beta \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T g_\beta \\ l_1^T V_q & l_1^T V_p & 0 & 0 \\ l_2^T V_q & l_2^T V_p & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta \bar{v}_q \\ \Delta \bar{v}_p \\ \Delta\alpha \\ \beta \end{bmatrix} = - \begin{bmatrix} V_q^T g_r \\ V_p^T g_r \\ l_1^T v \\ l_2^T v - 1 \end{bmatrix}, \tag{7.46}$$

where

$$\begin{aligned} g_\beta &= M_x v \Delta x_\beta + M_T v \Delta T_\beta + M_\gamma v \Delta \gamma_\beta, \\ g_r &= (M - I)v + \alpha f + M_x v \Delta x_r + M_T v \Delta T_r + M_\gamma v \Delta\gamma_r. \end{aligned}$$

First $\Delta \bar{v}_{q,r}$ and $\Delta \bar{v}_{q,\beta}$ are computed from

$$\left[V_q^T M V_q - I_q\right] \left[\begin{array}{cc} \Delta \bar{v}_{q,r} & \Delta \bar{v}_{q,\beta} \end{array}\right] = - \left[\begin{array}{cc} V_q^T g_r & V_q^T g_\beta \end{array}\right], \tag{7.47}$$

next $\Delta \bar{v}_p$, $\Delta\alpha$ and β are solved from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T(g_\beta + M \Delta v_{q,\beta}) \\ l_1^T V_p & 0 & l_1^T \Delta v_{q,\beta} \\ l_2^T V_p & 0 & l_2^T \Delta v_{q,\beta} \end{bmatrix} \begin{bmatrix} \Delta \bar{v}_p \\ \Delta\alpha \\ \beta \end{bmatrix} = - \begin{bmatrix} V_p^T(g_r + M\Delta v_{q,r}) \\ l_1^T(v + \Delta v_{q,r}) \\ l_2^T(v + \Delta v_{q,r}) - 1 \end{bmatrix}, \tag{7.48}$$

and finally, we compute

$$\begin{aligned} \Delta v &= V_p \Delta \bar{v}_p + (\Delta v_{q,r} + \beta \Delta v_{q,\beta}), \\ \Delta x(0) &= \Delta x_r + \beta \Delta x_\beta, \\ \Delta T &= \Delta T_r + \beta \Delta T_\beta, \\ \Delta\gamma &= \Delta\gamma_r + \beta \Delta\gamma_\beta. \end{aligned}$$

This method requires the solution of four Q-systems: (7.40), (7.44) and (7.47) with two different right-hand sides. The cost per iteration step is comparable to the cost for the method for period doubling points.

We have not yet implemented and tested this method. However, the good results of the Matlab experiments reported above and the similarities of the various subsystems with those in the period doubling case gives us good confidence that the above derivation is a solid basis for a Newton–Picard method for the computation of fold points. Further investigations are required to generalize the convergence analysis of section 3.2.2 to this case and to determine how accurate the solution $(\Delta x_\beta,\ \Delta T_\beta,\ \Delta\gamma_\beta)$ to (7.39) must be computed for the method to work well.

7.4 Multiple shooting

The methods discussed in this chapter can also be extended to multiple shooting. As an example, we will study the method for period doubling points. The extended system (7.1) corresponds to the system that is obtained if single shooting is used to solve the boundary value problem (1.61). We can also solve this boundary value problem using multiple shooting. Consider again the mesh (1.27). Multiple shooting applied to (1.61) results in

$$\left\{\begin{array}{l} r_1 := \varphi(x_0, \nabla s_1 T, \gamma) - x_1 = 0, \\ \qquad\qquad\qquad\vdots \\ r_m := \varphi(x_{m-1}, \nabla s_m T, \gamma) - x_0 = 0, \\ s(x_0, \ldots, x_{m-1}, T, \gamma) = 0, \\ r_{e,1} := M(x_0, \nabla s_1 T, \gamma) v_0 - v_1 = 0, \\ \qquad\qquad\qquad\vdots \\ r_{e,m-1} := M(x_{m-2}, \nabla s_{m-1} T, \gamma) v_{m-2} - v_{m-1} = 0, \\ r_{e,m} := M(x_{m-1}, \nabla s_m T, \gamma) v_{m-1} + v_0 = 0, \\ n(v_0) = l^T v_0 - 1 = 0. \end{array}\right. \tag{7.49}$$

Of course, one can also use another normalization equation $n(v_0) = 0$ or $n(v_0, \ldots, v_{m-1}) = 0$. After the Newton linearization, we obtain the linear system

$$\begin{bmatrix} G_1 & -I & & b_{T,1} & & & & b_{\gamma,1} \\ & \ddots & \ddots & \vdots & & & & \vdots \\ -I & & G_m & b_{T,m} & & & & b_{\gamma,m} \\ c_{s,1}^T & \cdots & c_{s,m}^T & d_{s,T} & & & & d_{s,\gamma} \\ G_{1,x} v_0 & & & G_{1,T} v_0 & G_1 & -I & & G_{1,\gamma} v_0 \\ & \ddots & & \vdots & & \ddots & \ddots & \vdots \\ & & G_{m,x} v_{m-1} & G_{m,T} v_{m-1} & +I & & G_m & G_{m,\gamma} v_{m-1} \\ & & & & l^T & & & \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \vdots \\ \Delta x_{m-1} \\ \Delta T \\ \Delta v_0 \\ \vdots \\ \Delta v_{m-1} \\ \Delta \gamma \end{bmatrix} = - \begin{bmatrix} r_1 \\ \vdots \\ r_m \\ s \\ r_{e,1} \\ \vdots \\ r_{e,m} \\ n \end{bmatrix} \tag{7.50}$$

with

$$\begin{aligned} G_{i,x} &= \frac{\partial M(x_{i-1}, \nabla s_i T, \gamma)}{\partial x_{i-1}}, \\ G_{i,T} &= \frac{\partial M(x_{i-1}, \nabla s_i T, \gamma)}{\partial T}, \\ G_{i,\gamma} &= \frac{\partial M(x_{i-1}, \nabla s_i T, \gamma)}{\partial \gamma}. \end{aligned}$$

The block

$$\begin{bmatrix} G_1 & -I & \\ & \ddots & \ddots \\ -I & & G_m \end{bmatrix}$$

corresponds to the block $M - I$ in (7.2), the block

$$\begin{bmatrix} G_1 & -I & \\ & \ddots & \ddots \\ +I & & G_m \end{bmatrix}$$

to $M + I$, etc.

To derive a Newton–Picard procedure, one needs to adapt assumption 6.3 slightly (in the spirit of assumption 7.1). Bases and projectors are constructed as in section 6.2.1. The variables x_i and v_i are decomposed according to

$$\begin{aligned} x_i &= V_{p,i}\bar{x}_{p,i} + V_{q,i}\bar{x}_{q,i}, \quad x_{p,i} = V_{p,i}\bar{x}_{p,i} = P_i x_i, \quad x_{q,i} = V_{q,i}\bar{x}_{q,i} = Q_i x_i \\ v_i &= V_{p,i}\bar{v}_{p,i} + V_{q,i}\bar{v}_{q,i}, \quad v_{p,i} = V_{p,i}\bar{v}_{p,i} = P_i v_i, \quad v_{q,i} = V_{q,i}\bar{v}_{q,i} = Q_i v_{i.} \end{aligned}$$

These formulas are inserted in (7.50) and blockrows containing G_i are premultiplied with $[V_{q,i} \ \ V_{p,i}]^T$. After some reordering and neglecting the terms $V_{q,i}^T b_{T,i}$, we obtain the system

$$\begin{bmatrix} F_{qq}^{o-} & & & & & \{V_{q,i}^T b_{\gamma,i}\} \\ F_{pq}^{o-} & F_{pp}^{o-} & \{V_{p,i}^T b_{T,i}\} & & & \{V_{p,i}^T b_{\gamma,i}\} \\ \{c_{s,i}^T V_{q,i-1}\} & \{c_{s,i}^T V_{p,i-1}\} & d_{s,T} & & & d_{s,\gamma} \\ F_{xqq}^{o} & F_{xqp}^{o} & F_{Tq} & F_{qq}^{o+} & & F_{\gamma q} \\ F_{xpq}^{o} & F_{xpp}^{o} & F_{Tp} & F_{pq}^{o+} & F_{pp}^{o+} & F_{\gamma p} \\ & & & \{l_i^T V_{q,i-1}\} & \{l_i^T V_{p,i-1}\} & \end{bmatrix}$$
$$\begin{bmatrix} \{\Delta\bar{x}_{q,i-1}\} \\ \{\Delta\bar{x}_{p,i-1}\} \\ \Delta T \\ \{\Delta\bar{v}_{q,i-1}\} \\ \{\Delta\bar{v}_{p,i-1}\} \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \{V_{q,i}^T r_i\} \\ \{V_{p,i}^T r_i\} \\ s \\ \{V_{q,i}^T r_{e,i}\} \\ \{V_{p,i}^T r_{e,i}\} \\ n \end{bmatrix} \qquad (7.51)$$

with

$$F_{qq}^{o-} = \begin{bmatrix} F_{qq,1} & -I_q & \\ & \ddots & \ddots \\ -I_q & & F_{qq,m} \end{bmatrix}, \ F_{qp}^{o-} = \begin{bmatrix} F_{qp,1} & & \\ & \ddots & \\ & & F_{qp,m} \end{bmatrix},$$

$$F_{pq}^{o-} = \begin{bmatrix} F_{pq,1} & & \\ & \ddots & \\ & & F_{pq,m} \end{bmatrix}, \ F_{pp}^{o-} = \begin{bmatrix} F_{pp,1} & -I_p & \\ & \ddots & \ddots \\ -I_p & & F_{pp,m} \end{bmatrix}$$

$$F_{qq}^{o+} = \begin{bmatrix} F_{qq,1} & -I_q & \\ & \ddots & \ddots \\ +I_q & & F_{qq,m} \end{bmatrix}, \ F_{qp}^{o+} = F_{qp}^{o-},$$

$$\begin{aligned}
F_{pq}^{o+} = F_{pq}^{o-},\ F_{pp}^{o+} &= \begin{bmatrix} F_{pp,1} & -I_p & \\ & \ddots & \ddots \\ +I_p & & F_{pp,m} \end{bmatrix} \\
F_{Tq} = \left\{V_{q,i}^T G_{i,T} v_{i-1}\right\},\ F_{\gamma q} &= \left\{V_{q,i}^T G_{i,\gamma} v_{i-1}\right\}, \\
F_{Tp} = \left\{V_{p,i}^T G_{i,T} v_{i-1}\right\},\ F_{\gamma p} &= \left\{V_{p,i}^T G_{i,\gamma} v_{i-1}\right\}, \\
l_1 = l,\ l_2, \ldots, l_m &= 0_{N\times 1}
\end{aligned}$$

and $F_{qq,i}$, $F_{qp,i}$, $F_{pq,i}$ and $F_{pp,i}$ as in (6.18). Each of these blocks correspond to the block in (7.6) at the same position and the solution method of (7.6) is easily generalized to (7.51).

First, $\Delta\bar{x}_{q,i-1,r}$ and $\Delta\bar{x}_{q,i-1,\gamma}$ are solved from

$$F_{qq}^{o-} \left[\ \{\Delta\bar{x}_{q,i-1,r}\} \quad \{\Delta\bar{x}_{q,i-1,\gamma}\} \ \right] = -\left[\ \left\{V_{q,i}^T r_i\right\} \quad \left\{V_{q,i}^T b_{\gamma,i}\right\} \ \right],$$

next $\Delta\bar{x}_{p,i-1,r}$, $\Delta\bar{x}_{p,i-1,\gamma}$, ΔT_r and ΔT_γ are computed from

$$\begin{aligned}
&\begin{bmatrix} F_{pp}^{o-} & \left\{V_{p,i}^T b_{T,i}\right\} \\ \left\{c_{si}^T V_{q,i-1}\right\} & d_{s,T} \end{bmatrix} \begin{bmatrix} \{\Delta\bar{x}_{p,i-1,r}\} & \{\Delta\bar{x}_{p,i-1,\gamma}\} \\ \Delta T_r & \Delta T_\gamma \end{bmatrix} = \\
&\quad -\begin{bmatrix} \left\{V_{p,i}^T\left(r_i + G_i \Delta x_{q,i-1,r}\right)\right\} & \left\{V_{p,i}^T\left(b_{\gamma,i} + G_i \Delta x_{q,i-1,\gamma}\right)\right\} \\ s + \sum_{i=1}^m c_{s,i}^T \Delta x_{q,i-1,r} & d_{s,\gamma} + \sum_{i=1}^m c_{s,i}^T \Delta x_{q,i-1,\gamma} \end{bmatrix}.
\end{aligned}$$

We can then set

$$\begin{aligned}
\Delta x_{i,r} &= V_p \Delta\bar{x}_{p,i,r} + V_q \Delta\bar{x}_{q,i,r}, \\
\Delta x_{i,\gamma} &= V_p \Delta\bar{x}_{p,i,\gamma} + V_q \Delta\bar{x}_{q,i,\gamma}.
\end{aligned}$$

After substitution of these expressions in the lower $Nm+1$ equations of (7.51), we can compute $\Delta\bar{v}_{q,i,r}$ and $\Delta\bar{v}_{q,i,\gamma}$ from

$$F_{qq}^{o+} \left[\ \{\Delta\bar{v}_{q,i-1,r}\} \quad \{\Delta\bar{v}_{q,i-1,\gamma}\} \ \right] = -\left[\ \left\{V_{q,i}^T g_{i,r}\right\} \quad \left\{V_{q,i}^T g_{i,\gamma}\right\} \ \right] \tag{7.52}$$

with

$$\begin{aligned}
g_{i,r} &= r_{e,i} + G_{i,x} v_{i-1} \Delta x_{i-1,r} + G_{i,T} v_{i-1} \Delta T_r, \\
g_{i,\gamma} &= G_{i,\gamma} v_{i-1} + G_{i,x} v_{i-1} \Delta x_{i-1,\gamma} + G_{i,T} v_{i-1} \Delta T_\gamma.
\end{aligned}$$

It is possible to generalize the Picard iteration or the GMRES method to take into account the $+$-sign in the last row of F_{qq}^{o+} instead of the $-$-sign. For instance, in algorithm 6.1 we need to replace the line

$$\Delta\bar{q}_{0,*} \leftarrow F_{qq,m} \Delta\bar{q}_{m-1,*} + V_{q,0}^T r_{m,*}$$

with

$$\Delta\bar{v}_{q,0,*} \leftarrow -F_{qq,m} \Delta\bar{v}_{q,m-1,*} - V_{q,0}^T g_{m,*}.$$

Finally, we need to solve the P-system

$$\begin{aligned}
&\begin{bmatrix} F_{pp}^{o+} & \left\{V_{p,i}^T\left(g_{i,\gamma} + F_{pq,i} \Delta v_{q,i-1,\gamma}\right)\right\} \\ \left\{l_i^T V_{p,i-1}\right\} & l^T \Delta v_{q,0,\gamma} \end{bmatrix} \begin{bmatrix} \{\Delta\bar{v}_{p,i-1}\} \\ \Delta\gamma \end{bmatrix} = \\
&\quad -\begin{bmatrix} \left\{V_{p,i}^T\left(g_{i,r} + F_{pq,i} \Delta v_{q,i-1,r}\right)\right\} \\ l^T(v_0 + \Delta v_{q,0,r}) - 1 \end{bmatrix}
\end{aligned}$$

and compute

$$\begin{array}{rcl}\Delta x_i & = & \Delta x_{i,r} + \Delta\gamma \Delta x_{i,\gamma}, \\ \Delta T & = & \Delta T_r + \Delta\gamma \Delta T_\gamma, \\ \Delta v_i & = & V_{p,i}\Delta \bar{v}_{p,i} + V_q \left(\Delta v_{q,i,r} + \Delta\gamma \Delta v_{q,i,\gamma}\right).\end{array}$$

In a similar way, we can also adapt the method for the computation of torus bifurcation points by solving the boundary value problem (1.67) and the method for fold points by using (1.64). Both methods are easily generalized by replacing each block in the single shooting case with the corresponding block of the projected linearized multiple shooting system. For the torus bifurcation case, a further adaptation of the Picard iteration scheme is needed. All of these methods need further analysis and testing. However, it is clear that our Newton–Picard approach is very powerful and useful for the efficient computation of various types of bifurcation points in large-scale problems.

7.5 Alternative approaches

7.5.1 Indirect methods

Our Newton–Picard methods can compute the dominant Floquet multipliers accurately at a reasonable cost. Hence we can easily use the Floquet multipliers to locate bifurcation points on branches of periodic solutions using indirect methods. Indirect methods are less suited to compute a curve of bifurcation points of a given type but are very well suited to locate a bifurcation point on a branch of periodic solutions of lower-codimension bifurcation points or periodic solutions. They are appealing because of their simplicity. Indirect methods do not need new solvers, which are often specific for a particular type of bifurcation point and hard to write, but use the existing solver for the computation of the branch and can be implemented in such a way that the same code can be used for various types of bifurcation points (also higher codimension bifurcation points on curves of codimension one bifurcation points) by passing the test function as an argument to the routine. However, these methods are usually not as efficient as a good direct method. We do not yet have much experience with such approaches but do believe that they are sometimes an interesting option if a test function exists that can be computed at a reasonable cost (e.g., a test function based on the dominant Floquet multipliers).

7.5.2 Extended systems based on scalar test functions.

In this approach, the pseudo-arclength condition is replaced by a scalar test function. The structure of the system (2.16) is maintained and the Newton–Picard solver does not need to be changed a lot. The problem with this approach is the evaluation of the test function and the computation of the derivatives with respect to T and γ and the scalar products of the derivatives with respect to the initial state $x(0)$ and a given vector. For instance,

$$\det\left(M(x(0), T, \gamma) + I\right) = 0 \tag{7.53}$$

is a scalar condition for a period doubling point. However, it is not possible to evaluate (7.53) without first constructing the monodromy matrix. This is precisely what we try to avoid in our method because of the high cost. This approach is only appealing if we have a test function that can be evaluated at a reasonable cost, does not require the computation of the monodromy matrix, can be differentiated cheaply with respect to T and γ and allows the computation of scalar products of the derivative with respect to $x(0)$ and a given vector at a reasonable cost.

7.5.3 Methods based on the projected system

All eigenvalues of M close to the unit circle are preserved in the projected monodromy matrix $V_p^T M V_p$. Furthermore, lemma 2.6 shows that a condition of the type "a vector does or does not lie in the range of M" can also be carried over to the projected system. Hence we can construct many extended system (either a scalar extension or a vector extension) using the projected quantities instead of the quantities in the N-dimensional space. At first sight, this is a cheap approach since a smaller matrix is used in the extended system. However, the approach has some serious disadvantages:

- The basis V_p depends on $x(0)$, T and γ. Hence derivatives of V_p are needed. One needs to assure that changes in the basis are continuous, which is difficult because of the reordering.
- The result can only be as accurate as the projected monodromy matrix and other projected quantities needed in the determining system, e.g.,

  $$V_p^T \left(b_\gamma - M V_q \left(V_q^T M V_q - I_q \right)^{-1} V_q^T b_\gamma \right).$$

  In the solvers developed in this chapter, the accuracy of the basis and projected quantities is one of the factors that influences the convergence speed. However, there is little or no influence on the accuracy of the final result. If extended systems based on projected quantities are used, one will need to compute the basis and other quantities used in the extended system with much higher precision.

A method of this type is used in [45]. Jarausch and Mackens extend their condensed Newton–supported Picard method to compute fold points and simple and double bifurcation points in a similar way in [57]. It is clear that this type of methods is complex (because derivatives of the basis are needed) and quite expensive (because we need to compute the basis and sometimes other quantities also with very high precision). Hence we believe that this approach should only be considered if no other affordable alternative is available.

7.6 Conclusions

In this chapter, we discussed some extensions of our Newton–Picard methods to compute bifurcation points. We derived a method for period doubling points and showed that the method will work for a non-degenerate period doubling point provided the various subsystems are solved accurately enough. We discussed the implementation aspects. At each step, four Q-systems need to be solved compared to one for the $NPGS$ and two for the CNP method, and there are also some additional time integrations needed for the computation of certain derivatives. We also presented a testcase.

To demonstrate the power of the approach, we also considered the computation of torus bifurcation points and fold points. The former requires a special Picard iteration scheme and we proved the asymptotic convergence of this scheme. In the method for the computation of fold points, the first P-system is singular. We presented a special approach to cope with this problem. The approach can be used without the Newton–Picard procedure but is also very well suited for use with this method. We also showed how the various methods can be combined with multiple shooting. The analysis of the methods described in the second part of this chapter is not yet complete and testing is necessary. However, we believe that the approaches are the foundation for the development of truly robust methods.

At the end of the chapter, we also discussed some alternative approaches.

Much work remains to be done in this area. First, some of the methods need further analysis. All methods should be combined with the convergence analysis techniques of section 3.2.2 and 6.2.2 to develop truly robust solvers. We believe that our approach can also be extended easily to the path-following of bifurcation points in a two-parameter space and the computation of many other higher codimension bifurcation phenomena. It should be verified which bifurcation phenomena can be computed using the approach of extended systems based on the solution of boundary value problems similar to (1.61), (1.67) and (1.64) (or equivalently, extended systems based on matrix–vector products with the monodromy matrix). For the other phenomena or for more complex phenomena, it is certainly worth to study some of the alternatives in more detail.

Our approach to apply the Newton–Picard method to the computation of bifurcation points of periodic solutions is an original contribution of this thesis. Jarausch and Mackens considered the use of their condensed Newton–supported Picard method for the computation of fold points, simple transcritical bifurcation points and double bifurcation points of steady-state solution branches in [57], but their method relies on the full decoupling in the condensed Newton–supported Picard method and uses extended systems based on the low-dimensional projected system.

Chapter 8

Conclusions and suggestions for future work

8.1 Conclusions

In this thesis, we presented a family of robust and efficient methods to compute branches of periodic solutions of systems of partial differential equations with low-dimensional dynamics based on the ideas behind the recursive projection method [107] and the condensed Newton–supported Picard method [56, 57, 58]. We constructed methods based on single- and multiple shooting. Special attention was paid to applications in bifurcation analysis. For that purpose, we extended the method to compute bifurcation points accurately and embedded robust strategies to compute the dominant Floquet multipliers.

This thesis makes several important contributions:

- We extended the recursive projection method to the computation and continuation of periodic solutions using single shooting. This required dealing with the extra bordering of the linearized systems caused by the phase condition. Moreover, we took an original starting point that made us discover new and more robust variants of the method that take into account a coupling term that is important for nonnormal problems but neglected in [107]. We paid special attention to the robustness of the method without neglecting the efficiency aspect. This leads to different trade-offs compared to previous methods and several extensions: the use of multiple Picard steps (and even a varying number) at each Newton–Picard step, the use of GMRES to solve the high-dimensional subsystems, the use of orthogonal subspace iteration with projection and locking instead of ordinary orthogonal subspace iteration and a different and more robust strategy to determine the basis size. The latter is particularly important if the method is to be used in bifurcation analysis. One of our variants also preserves more information on bifurcation points in the low-dimensional subsystem than the methods of [107]. We also developed an algebraic framework in which all our methods fit. This framework allows us to understand the convergence behaviour, even with an imperfect basis, and is not limited to the asymptotic regime. We exploited the framework to develop very robust variants of our method.

To the best of our knowledge, only in [55] a similar method based on single shooting is studied. However, there are many important differences with our approach: a different basis is used which is harder to compute, there is no choice for the phase condition and parametrizing equation, no distinction is made between the system variables, system parameters and period and less information on bifurcations can be recovered from the method. Moreover, only the basic ideas are studied.

- We appreciated the potential of the periodic Schur decomposition [11] for the accurate computation of Floquet multipliers in multiple shooting techniques, some collocation methods and finite difference methods. The periodic QR algorithm can be applied to the matrices that appear in the linearized system in some cases or to matrices that appear during the solution of the linearized systems with some particular solvers. For larger problems, the method can be combined with a special variant of subspace iteration with locking and projection to compute the dominant Floquet multipliers.

- We generalized our most powerful single-shooting based method to multiple shooting. This method combines the ideas behind our single-shooting based method with the periodic Schur decomposition and the ideas of reorthogonalization and decoupling of the multiple shooting method in [4]. The generalization required extensions to the Picard iteration scheme and the GMRES method to allow for the simultaneous computation of all unknowns at about the same cost as in the single shooting case. We also developed a special version of orthogonal subspace iteration with projection and locking to compute the bases accurately and efficiently.

  As far as we know this is the first time that the Newton–Picard idea is applied to multiple shooting.

- We developed an efficient single-shooting based direct method to compute period doubling bifurcation points using the Newton–Picard idea. Furthermore, we have indicated how similar methods can be developed for the computation of torus bifurcation and fold points and how these methods can be extended to multiple shooting. We have discussed some alternatives to conclude that our method is certainly not a bad one.

  In [57], Jarausch and Mackens study a method to compute fold points and single or double bifurcation points along steady-state branches. Their approach is again quite different from ours and also relies on the decoupling of the low- and high-dimensional subsystems that occurs in their problems if the parameters are kept fixed. They define an extended system based on a projected Jacobian. We pointed out that such a method is cumbersome since derivatives of the basis are needed. Jarausch and Mackens do not really discuss this point in their paper. They use numerical differences to compute the derivatives of the projected Jacobian but do not point out how they assure that basis changes are continuous.

At the end of the chapters 6 and 7, we already pointed out that there are still many open questions. We are convinced that the method has great potential and can be further extended to other problem types (e.g., DDEs) and other underlying numerical techniques such as some collocation methods. We also believe that further investigation of the use of the method for steady-state problems is needed.

At this point, one can wonder whether it wouldn't make sense to apply a Krylov method (GMRES for instance) directly to the linearized single- or multiple-shooting system instead of using the rather difficult to implement Newton–Picard method. We have little experience with such methods, but nevertheless we believe that our approach has several advantages:

- It is not clear what the spectrum of the linearized system looks like. For the single shooting method with the parameter fixed, one obtains the matrix

  $$\begin{bmatrix} M - I & b_T \\ c_s^T & d_{s,T} \end{bmatrix}. \tag{8.1}$$

  If μ_i are the eigenvalues of M at the periodic solution with $\mu_1 = 1$ the trivial Floquet multiplier, then it is possible to show that at the limit cycle, the eigenvalues of (8.1) are $(d_{s,T}/2) \pm \sqrt{c_s^T b_T + (d_{s,T}/2)^2}$, $\mu_2 - 1$, …, $\mu_N - 1$. If the GMRES method is applied to this system, the eigenvalues of M outside or near the unit circle may slow down the convergence. In fact, the Krylov space typically has to capture the outermost eigenvalues well enough before the method starts to converge well.

  We do not know an expression for the spectrum of

  $$\begin{bmatrix} M - I & b_T & b_\gamma \\ c_s^T & d_{s,T} & d_{s,\gamma} \\ c_n^T & d_{n,T} & d_{n,\gamma} \end{bmatrix}$$

  with an arbitrary parametrizing equation and it is also an open question what the spectrum looks like in the linearized multiple shooting system or in the methods for the computation of bifurcation points. Hence it is not clear at all how the GMRES method would perform if applied to these systems.

- Since we know little about the spectrum of the linearized system, we cannot easily recover the Floquet multipliers except for the matrix (8.1). For this matrix, the GMRES procedure will usually return estimates for some or all of the dominant eigenvalues that can be used as starting values for a postprocessing step.

- One can also look for methods that first split the linearized system in subsystems with a known spectrum. Furthermore, one can try to capture the outermost eigenvalues of these subsystems with a preconditioner. We believe that such methods would look a lot like our methods and doubt whether they would offer a big performance improvement. In fact, our methods combined with the GMRES solver for the high-dimensional subsystems are of this type. We have isolated all hard modes and removed them from the systems that are solved using the GMRES method.

Further research in this area and a thorough comparison with the Newton–Picard approach would certainly be useful.

8.2 Suggestions for future research

8.2.1 Overview of topics

In chapter 6 and 7 we already pointed out several aspects that deserve further attention. We just briefly recapitulate. The multiple shooting method may be further refined as follows.

- The implementation of the a priori and a posteriori analysis can be further refined. Better estimates for the size of the Δp_i's, ΔT and $\Delta\gamma$ are needed. Also, damping would be an useful extension to control the higher order terms. This is particularly important if a solution at a given parameter value is to be computed or to compute the first point on a branch. Of course, the same remarks hold for our single shooting based methods, but since our multiple shooting methods reduce to the corresponding single shooting method if only one interval is used, we think that it is not necessary to maintain a separate implementation.

- A strategy to determine the multiple shooting mesh intervals is needed to complete the method. Also, a good strategy for the remeshing of the time grid used by the integrators on each interval would be very useful.

- It may be better to base an implementation on an alternative multiple shooting system that expresses the periodic boundary conditions as a different set of equations. We do not expect that this will influence the method a lot.

- It is certainly worth to further investigate the approach of the "geometric phase condition".

Several refinements and extensions to the methods for computing bifurcation points should also be investigated.

- We only indicated how our single shooting based method for period doubling points can be extended to the computation of torus bifurcation points. These methods need further analysis and testing. The same holds for the extensions to multiple shooting.

- The methods should also be extended to path-following of bifurcation points in a two-parameter space.

- It should also be investigated which other bifurcation phenomena can be computed using similar methods.

- We mentioned some alternative approaches. Although the alternative direct methods based on a scalar extension or on an extended system defined using the low-dimensional projected system didn't seem to be very interesting for computation of the types of bifurcation points which we studied, they might be interesting for some other types.

Several other extensions of our work also come into mind.

- The algorithms for reordering of the periodic Schur decomposition should be further refined. Several methods for reordering of the Schur decomposition of a matrix have been developed and implemented in software packages such as LAPACK, see, e.g., [6, 33]. These methods should be generalized to the periodic Schur decomposition if possible.

- We have done some work on the control of the stepsize in a continuation code based on the evolution of bifurcation test functions along the branch. The stepsize is decreased if a possible zero crossing of the test function is approached. We combined such a strategy with an indirect method for the computation of bifurcation points. The indirect method was used to generate good starting values for a direct method and not to compute the bifurcation points to full accuracy. The results were very promising. The method reduced the number of branch jumps and we were able to detect two consecutive bifurcation points of the same type which otherwise would have remained undetected since the original continuation code did not generate a point in between them and the test function did not change sign . However, the method needs to be further worked out and coupled to our Newton–Picard solvers.

- The methods should be combined with grid refinement of both the space and the time grid. In our current techniques, the PDE system is immediately transformed to an ODE system and it is not verified whether the discretized system is a good approximation to the infinite-dimensional PDE system. Periodic solutions at different parameter values may require a different grid to approximate the solution of the continuous PDE model with the desired accuracy.

- We believe that there are other types of equations that have similar properties as our PDE systems. The Newton–Picard approach should be extended to those systems also. We already discussed an extension of our single shooting based method to DDEs but still need to adapt our multiple shooting code. Other types of equations, e.g., large systems of differential-algebraic equations, should also be considered.

- We pointed out that it is not a trivial task to develop an efficient Newton–Picard method to compute steady-state solutions of ODE systems, certainly not if one also wishes to detect bifurcation points. Further research in this area is certainly needed. We believe that combinations with methods to compute the rightmost eigenvalues of a matrix may lead to results.

- We are convinced that the Newton–Picard method can be extended to Gauss–Legendre collocation and will elaborate on this in section 8.2.2.

- Our algorithms should be implemented on parallel computers if systems with more than a few 100s of degrees of freedom have to be studied. We will discuss some aspects in section 8.2.3.

- A more user-friendly interface should be developed, see 8.2.4 for some comments.

This thesis is one of the first texts that studies the computation of periodic solutions of large-scale problems using classical simulation techniques and the Newton–Picard approach. It is clear that much work remains to be done in this area. This only confirms the great potential of the Newton–Picard approach.

8.2.2 Gauss–Legendre collocation methods

Gauss–Legendre collocation is a popular and robust method to compute periodic solutions of small systems. A good implementation is available in the software package AUTO. Although we think that multiple shooting combined with a very good time integrator and a good strategy for the selection of the mesh is able to compute periodic solutions in many practical models, in some cases, collocation might be more robust or cheaper. In this section we wish to indicate how the Newton–Picard procedure can be generalized to Gauss–Legendre collocation methods. The analysis is far from complete and mentions only the basic ideas. Hence we chose to include it as a topic for further research.

The Gauss–Legendre collocation method was described in section 1.4.3. If the method is applied to pseudo-arclength continuation of periodic solutions and if Newton's method is used to solve the resulting nonlinear system, we obtain a sequence of linear systems

$$\left[\begin{array}{cccccccccc}
A_{0,0,1} & A_{0,1,1} & \cdots & A_{0,n,1} & & & & & b_{T,0,1} & b_{\gamma,0,1} \\
\vdots & & & \vdots & & & & & \vdots & \vdots \\
A_{0,0,n} & A_{0,1,n} & \cdots & A_{0,n,n} & & & & & b_{T,0,n} & b_{\gamma,0,n} \\
 & & & \ddots & \ddots & \ddots & & & \vdots & \vdots \\
 & & & & & A_{m-1,0,1} & \cdots & A_{m-1,n,1} & b_{T,m-1,0} & b_{\gamma,m-1,0} \\
 & & & & & \vdots & & \vdots & \vdots & \vdots \\
 & & & & & A_{m-1,0,n} & \cdots & A_{m-1,n,n} & b_{T,m-1,n} & b_{\gamma,m-1,n} \\
-I & & & & & & & I & & \\
c_{s,0,0} & c_{s,0,1} & \cdots & \begin{array}{c} c_{s,0,n}+ \\ c_{s,1,0} \end{array} & \cdots & \begin{array}{c} c_{s,m-2,n}+ \\ c_{s,m-1,0} \end{array} & \cdots & c_{s,m-1,n} & d_{s,T} & d_{s,\gamma} \\
c_{n,0,0} & c_{n,0,1} & \cdots & \begin{array}{c} c_{n,0,n}+ \\ c_{n,1,0} \end{array} & \cdots & \begin{array}{c} c_{n,m-2,n} \\ c_{n,m-1,0} \end{array} & \cdots & c_{n,m-1,n} & d_{n,T} & d_{n,\gamma}
\end{array}\right]
\left[\begin{array}{c} \Delta x_0 \\ \Delta x_{1/n} \\ \vdots \\ \Delta x_1 \\ \vdots \\ \Delta x_{m-1} \\ \vdots \\ \Delta x_m \\ \Delta T \\ \Delta\gamma \end{array}\right]
= -\left[\begin{array}{c} r_{0,1} \\ \vdots \\ r_{0,n} \\ \vdots \\ r_{m-1,1} \\ \vdots \\ r_{m-1,n} \\ r_m \\ s \\ n \end{array}\right]. \tag{8.2}$$

If $\int_0^1 g_s(x(\tau),T,\gamma)d\tau$ represents the integral phase condition and $\int_0^1 g_u(x(\tau),T,\gamma)d\tau$ the integral parametrizing equation, then

$$c_{*,i,j} = w_{i,j}\left.\frac{\partial g_*}{\partial x}\right|_{(x_{i+j/n}T,\gamma)}, \; i = 0,\dots,m-1, \; j = 0,\dots,n,$$

$$d_{s,T} = \sum_{i=0}^{m-1} \sum_{j=0}^{n} w_{i,j} \left. \frac{\partial g_s}{\partial T} \right|_{(x_{i+j/n}T,\gamma)} \text{, etc.,}$$

$$b_{T,i,k} = f(p_i(z_{i,k}), \gamma),$$

$$b_{\gamma,i,k} = T \left. \frac{\partial f}{\partial \gamma} \right|_{(p_i(z_{i,k}),\gamma)},$$

$$r_{i,k} = Tf(p_i(z_{i,k}), \gamma) - \left. \frac{dp_i}{ds} \right|_{s = z_{i,k}},$$

$$r_m = x_m - x_0$$

and $A_{i,j,k}$ given by (1.41). Here $p_i(s)$, $z_{i,k}$ and $x_{i+j/m}$ are defined as in section 1.4.3. It is possible to transform this system to the system

$$\begin{bmatrix}
G_{0,1} & -I & & & & & & & & & & \tilde{b}_{T,0,1} & \tilde{b}_{\gamma,0,1} \\
\vdots & & \ddots & & & & & & & & & \vdots & \vdots \\
G_{0,n} & & & -I & & & & & & & & \tilde{b}_{T,0,n} & \tilde{b}_{\gamma,0,n} \\
 & & & G_{1,1} & -I & & & & & & & \tilde{b}_{T,1,1} & \tilde{b}_{\gamma,1,1} \\
 & & & \vdots & & \ddots & & & & & & \vdots & \vdots \\
 & & & G_{1,n} & & & -I & & & & & \tilde{b}_{T,1,n} & \tilde{b}_{\gamma,1,n} \\
 & & & & & & \ddots & \ddots & \ddots & & & \vdots & \vdots \\
 & & & & & & & G_{m-1,1} & -I & & & \tilde{b}_{T,m-1,1} & \tilde{b}_{\gamma,m-1,1} \\
 & & & & & & & \vdots & & \ddots & & \vdots & \vdots \\
 & & & & & & & G_{m-1,n} & & & -I & \tilde{b}_{T,m-1,n} & b_{\gamma,m-1,n} \\
-I & & & & & & & & & & I & & \\
\tilde{c}_{s,0} & & & \tilde{c}_{s,1} & & & \cdots & \tilde{c}_{s,m-1} & & & & \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} \\
\tilde{c}_{n,0} & & & \tilde{c}_{n,1} & & & \cdots & \tilde{c}_{n,m-1} & & & & \tilde{d}_{n,T} & \tilde{d}_{n,\gamma}
\end{bmatrix}
\begin{bmatrix}
\Delta x_0 \\ \Delta x_{1/n} \\ \vdots \\ \Delta x_1 \\ \Delta x_{1+1/n} \\ \vdots \\ \Delta x_2 \\ \vdots \\ \Delta x_{m-1} \\ \Delta x_{(m-1)+1/n} \\ \vdots \\ \Delta x_m \\ \Delta T \\ \Delta \gamma
\end{bmatrix}
= -
\begin{bmatrix}
\tilde{r}_{0,1} \\ \vdots \\ \tilde{r}_{0,n} \\ \tilde{r}_{1,1} \\ \vdots \\ \tilde{r}_{1,n} \\ \vdots \\ \tilde{r}_{m-1,1} \\ \vdots \\ \tilde{r}_{m-1,n} \\ \tilde{r}_m \\ \tilde{s} \\ \tilde{n}
\end{bmatrix}. \tag{8.3}$$

Note that the algorithm for the reduction can be very unstable if no pivoting is used. However, we need the transformation only for theoretical purposes. The transformed system (8.3) can be solved by first solving

$$\begin{bmatrix} G_{0,n} & -I & & & & \tilde{b}_{T,0,n} & \tilde{b}_{\gamma,0,n} \\ & G_{1,n} & -I & & & \tilde{b}_{T,1,n} & \tilde{b}_{\gamma,1,n} \\ & & \ddots & \ddots & & \vdots & \vdots \\ & & & G_{m-1,1} & -I & \tilde{b}_{T,m-1,n} & \tilde{b}_{\gamma,m-1,n} \\ -I & & & & I & & \\ \tilde{c}_{s,0} & \tilde{c}_{s,1} & \cdots & \tilde{c}_{s,m-1} & & \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} \\ \tilde{c}_{n,0} & \tilde{c}_{n,1} & \cdots & \tilde{c}_{n,m-1} & & \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_{m-1} \\ \Delta x_m \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{0,n} \\ \tilde{r}_{1,n} \\ \vdots \\ \tilde{r}_{m-1,n} \\ \tilde{r}_m \\ \tilde{s} \\ \tilde{n} \end{bmatrix} \tag{8.4}$$

and then computing the other unknowns from the other equations of (8.3). Note that the condensed system (8.4) has precisely the same structure as the linearized system of the multiple shooting method (6.52). Hence we can solve this system approximately using the Newton–Picard procedure. To implement the procedure efficiently, one must find a way to compute the matrix–vector products $G_{i,k}v$, the scalar products $\tilde{c}_{*,i}v$ and the terms $\tilde{b}_{*,i,k}$, $\tilde{d}_{*,*}$ and $\tilde{r}_{i,j}$ efficiently without explicitly constructing the matrices $G_{i,k}$. We will show that these terms can be computed as the numerical solution of certain variational equations using a Gauss–Legendre implicit Runge–Kutta time integrator. Hence the problem is reformulated as a problem of efficiently implementing these time integration schemes. We do not yet have a solution to this problem. However, a search in the huge literature on numerical time integration methods for PDEs might reveal a solution.

Suppose we wish to integrate the variational equation

$$\frac{dv}{dt} = \left.\frac{\partial f}{\partial x}\right|_{(p(s),T,\gamma)} v + u(s) \tag{8.5}$$

(with $u(s)$ defined on $[0,1]$ and $p(s)$ the spline approximation to $x(s)$) on the interval $[s_i, s_{i+1}]$ with $v(s_i)$ given. If we solve this linear initial value problem using one step of the n-stage Gauss–Legendre implicit Runge–Kutta scheme, $v_{i+j/n} = v(s_{i+j/n})$, $j = 1, \ldots, n$ has to be solved from the linear system

$$\begin{bmatrix} A_{i,0,1} & A_{i,1,1} & \cdots & A_{i,n,1} \\ \vdots & & & \vdots \\ A_{i,0,n} & A_{i,1,n} & \cdots & A_{i,n,n} \end{bmatrix} \begin{bmatrix} v_i \\ v_{i+1/n} \\ \vdots \\ v_{i+1} \end{bmatrix} + \begin{bmatrix} u_{i,1} \\ \vdots \\ u_{i,n} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.6}$$

with v_i given and $u_{i,k} = u(z_{i,k})$. This system can be transformed to

$$\begin{bmatrix} G_{i,1} & -I & & \\ \vdots & & \ddots & \\ G_{i,n} & & & -I \end{bmatrix} \begin{bmatrix} v_i \\ v_{i+1/n} \\ \vdots \\ v_{i+1} \end{bmatrix} + \begin{bmatrix} \tilde{u}_{i,1} \\ \vdots \\ \tilde{u}_{i,n} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{8.7}$$

Of course the transformations needed to move from (8.6) to (8.7) are the same as the transformations applied to the $i+1^{\text{st}}$ block of kN rows of (8.2) to obtain (8.3) and the matrices $G_{i,k}$ in (8.7) and (8.3) are precisely the same. From (8.7) we get

$$v_{i+k/n} = G_{i,k}v_i + \tilde{u}_{i,k}, \; k = 1, \ldots, n. \tag{8.8}$$

Let us first suppose $u(s) \equiv 0$. Then

$$v_{i+k/n} = G_{i,k} v_i.$$

Hence the matrix–vector products $G_{i,k} v$ can be computed by integrating the variational equation

$$\frac{dv}{dt} = T \left. \frac{\partial f}{\partial x} \right|_{(p(s),\gamma)} v \tag{8.9}$$

on $[s_i, s_{i+1}]$ using one step with the implicit Gauss–Legendre Runge–Kutta formula corresponding to the collocation method used to compute $x(s)$ and with $v(s_i) = v$.

Now let

$$u(s) = f(p(s), \gamma).$$

Then $u_{i,k} = f(p(z_{i,k}), \gamma) = b_{T,i,k}$. Suppose $v_i = 0$. Then

$$v_{i+k/n} = \tilde{b}_{T,i,k}.$$

We can thus compute the vectors $\tilde{b}_{T,i,k}$ in (8.3) by integrating the variational equation

$$\frac{dv}{dt} = T \left. \frac{\partial f}{\partial x} \right|_{(p(s),\gamma)} v + f(p(s), \gamma)$$

on $[s_i, s_{i+1}]$ using $v(s_i) = 0$. Similarly, if

$$u(s) = T \left. \frac{\partial f}{\partial \gamma} \right|_{(p(s),\gamma)}$$

the vectors $\tilde{b}_{\gamma,i,k}$ are obtained and if

$$u(s) = T f(p(s), \gamma) - \frac{dp}{ds}$$

one obtains the vectors $\tilde{r}_{i,k}$.

To transform the last two rows in (8.2) to (8.3), we add $c_{*,i,k} \times$ the blockrow of (8.3) containing $G_{i,k}$ to the last two rows, i.e.,

$$\tilde{c}_{*,i} = c_{*,i,0} + \sum_{k=1}^{n} c_{*.i,k} G_{i,k},$$

$$\begin{bmatrix} \tilde{d}_{s,T} & \tilde{d}_{s,\gamma} & \tilde{s} \\ \tilde{d}_{n,T} & \tilde{d}_{n,\gamma} & \tilde{n} \end{bmatrix} = \begin{bmatrix} d_{s,T} & d_{s,\gamma} & s \\ d_{n,T} & d_{n,\gamma} & n \end{bmatrix} + \sum_{i=0}^{m-1} \sum_{k=1}^{n} \begin{bmatrix} c_{s,i,k} \\ c_{n,i,k} \end{bmatrix} \begin{bmatrix} \tilde{b}_{T,i,k} & \tilde{b}_{\gamma,i,k} & \tilde{r}_{i,k} \end{bmatrix}.$$

Note that

$$\tilde{c}_{*,i} v_i = c_{*,i,0} v_i + \sum_{k=1}^{n} c_{*,i,k} G_{i,k} v_i = \sum_{j=0}^{n} c_{*,i,j} v_{i+j/n}$$

with $v_{i+j/n}$ the result of the numerical integration of (8.9). Once the matrix–vector product $G_{i,n} V_p$ has been computed, the term $\tilde{c}_{*,i} V_p$ is also easily computed provided the values for $G_{i,k} V_p$ have also been stored, which is usual practise in a collocation code. Similarly, the terms $\tilde{d}_{*,*}$, $\tilde{s}$ and $\tilde{n}$ are easily computed once the terms $\tilde{b}_{T,i,k}$, $\tilde{b}_{\gamma,i,k}$ and $\tilde{r}_{i,k}$ have been computed. Note that the evaluation of these terms resembles a lot the evaluation of the derivatives of the integral phase condition for shooting methods discussed in section 3.7.

Finite difference methods lead to a linearized system similar to (8.2) and we are convinced that the procedure outlined above can also be applied to some of these methods. In fact, one can replace $\varphi(x(0), T, \gamma)$ in the single shooting system (2.16) with an arbitrary one-step time integration scheme (i.e., a time integration scheme that computes $x(s_{i+1})$ from $x(s_i)$ using only the values $x(s_i)$, $x(s_{i+1})$ and values $x(s)$ with $s \in]s_i, s_{i+1}[$ such as an implicit Runge–Kutta formula, the backward Euler method or the trapezium rule) and solve the huge nonlinear system for all unknown values $x(s_i)$ and T and γ simultaneously. After linearization a system similar to (8.2) is obtained. If the linear system (8.7) can be solved easily for all unknown values of v given v_i (e.g., using an analogy with the integration of certain variational equations), the Newton–Picard procedure can be applied. We expect that the analogy with the integration of certain variational equations exists for many of the classical time integration schemes, but we need to further investigate this point. This procedure avoids the accurate time integration of the nonlinear problem (1.4) and instead only requires the solution of linear systems (8.7) (often equivalent to the time integration of linear problems). Such a technique will likely be more robust than a single or multiple shooting technique using the same time integration scheme and the same time grid. Note that not all time integration schemes give lead to a system similar to (8.2). For instance, linear multistep methods also use values of $x(s)$ for $s < s_i$ to compute $x(s_{i+1})$ from $x(s_i)$. The method also does not directly apply to methods that use Krylov subspaces to approximate an exponential [34, 100]. Doing a time step with such a method does not give lead to linear systems like (8.6).

If the Newton–Picard method is applied to the condensed system (8.4), we will need to reorthogonalize a set of basis vectors at each mesh interval boundary of the collocation method. This is an expensive operation on parallel computers since several global communication steps are needed. To avoid too frequent reorthogonalizations, one can use a subgrid of the collocation time mesh for the Newton–Picard method. Let $\{\hat{s}_l\} \subset \{s_i\}$, $l = 0, \ldots, m_2$, with

$$\hat{s}_0 = s_0 < \hat{s}_1 < \cdots < \hat{s}_{m_2} = s_m.$$

The system (8.4) can be further condensed to a system in the unknowns $x(\hat{s}_l)$, T and γ. Again, we can show that the computation of the quantities in that system will correspond to the results of the integration of certain variational equations using the original grid $\{s_i\}$. The number and spreading of the mesh points in the fine grid $\{s_i\}$ is determined by the required accuracy of the time discretization. The position and number of the mesh points in the coarse mesh are determined such that the solution procedure is numerically stable. One should avoid that components grow too much on a mesh interval of the coarse grid. In fact, a similar strategy can be used in the multiple shooting method but is not so much needed since usually the number of mesh intervals is quite low. The fine multiple shooting mesh is used to control the higher-order terms while a coarser submesh can be used to control the numerical stability of the solver.

It is clear that collocation methods or other similar methods will only be efficient if the variational equation (8.5) can be integrated efficiently or the system (8.7) can be solved cheaply given v_i. Moreover, these methods are extremely memory-consuming. However, they can easily be combined with an integral phase condition and pseudo-arclength equation which is advantageous if adaptive time meshes are used and we expect that they are also more robust than shooting methods for hard problems. Whether a multiple shooting method or a collocation-like method should be used depends on the properties of the periodic solutions that are computed, the amount of memory that can be used and the available time integration codes. In fact, we can distinguish three families of methods that are worth to implement:

- The multiple shooting method of chapter 6 is interesting if low memory requirements are important.
- The Gauss–Legendre collocation method or similar methods based on other time integration schemes are interesting if enough memory is available, if the variational equations can be integrated efficiently (or if the linear system (8.7) can be solved cheaply given v_i) and if robustness is the primary concern.
- We believe that it might sometimes be interesting to consider a hybrid method that is basically a multiple shooting method but stores the whole trajectory (and optionally information on the Jacobian of $f(x, \gamma)$ or the linearized systems in the time integrator code at each point). Such a method only makes sense if the approach of this section cannot be used in combination with the time integration scheme (because the system (8.6) cannot be solved easily or because the technique simply does not give lead to a system similar to (8.2)). Compared to a traditional multiple shooting method, the hybrid method has the advantage that an integral phase condition and parametrizing equation can be implemented at a reasonable cost. Such a condition leads to an increased efficiency if adaptive meshes are used in the time integrator. It is also easier to compute functions along the orbit for display in a bifurcation diagram. The method is also interesting if space-time methods such as waveform relaxation are used to integrate (1.4) since good starting values are available at each of the mesh points of the time grid.

8.2.3 Implementation on parallel computers

Suppose we wish to compute branches of a PDE system on a d-dimensional spatial domain. This problem is actually a $d + 2$-dimensional problem since we wish to compute functions of the d space coordinates, the time t and the artificial pseudo-arclength parameter η or the parameter γ. It is clear that only using parallelism in the spatial directions will not result in a high parallel efficiency if a large number of processors is used and if the problem size and total computation cost are reasonable. One will need to use strategies that also introduce parallelism in the time direction and if possible some parallelism in the parameter direction.

The multiple shooting method does not introduce additional parallelism in the time direction compared to the single shooting method. The operations on successive multiple shooting mesh intervals are purely sequential since each operation on one interval depends on the result of an operation on the previous one. Of course, the same holds for the collocation and collocation-like techniques introduced in the previous section. Hence we should look for parallelism in the time integrator itself. We think that space-time methods that have parallelism in both the space and time direction should be used for the time integrations if one wishes to obtain a high efficiency on massively parallel computers. In a multiple shooting code, the multiple shooting mesh intervals should not only be determined by the desire to control the higher order terms but should also be large enough to assure a good parallel efficiency and yet small enough for a good convergence of the time integration method. Similarly, in a collocation-like technique, the size of the coarse mesh intervals should be well chosen to assure a good numerical stability, a good parallel efficiency and a good convergence of the time integration method. Time integration is done on the intervals of the intermediate grid and the reorthogonalizations for the Newton–Picard procedure at the mesh points of the coarsest grid.

Since usually, most of the time is spent in the time integration code, this piece of the code is clearly the first target for parallelization. Such a strategy is good enough if one or more of the following conditions hold:

- If only a few processors are used.
- If not too many reorthogonalizations are needed (i.e., the multiple shooting intervals are large). These are operations on individual vectors and not on time profiles and have a low parallel efficiency.
- If the problem size is sufficiently large.

If a better scalability is desired, one should also parallelize the operations in the Newton–Picard method that involve N-dimensional vectors. There are two basis operations: linear combinations (which parallelize well) and scalar products. The latter are used to compute norms, in the reorthogonalization procedure and to project vectors. This operation requires global communication and can reduce the parallel efficiency. The operations on matrices or vectors with a size proportional to p can still be carried out sequentially except if p would be very large. If a homogenous distributed memory computer is used (i.e., a network of equal nodes with equal processors), we can perform these operations on each of the processors simultaneously. If it cannot be guaranteed that all processors produce exactly the same output from exactly the same input data, then the computations should be carried out on one node and the results communicated to all other nodes.

8.2.4 User interface aspects

A code for bifurcation analysis is typically used in a cyclic sequence of three steps. First, the user edits some input files to give instructions to the code, next the code is run and finally, the results are combined with previous results and interpreted. For large problems, the code may be run on a different computer than the machine used to prepare the input files and study the results. In the current implementation of our Newton–Picard methods, input to the code is provided via a rather cryptic input file. The output is a set of text files and optionally a LaTeX report with information on the convergence of the solver. Data can be extracted from the output files and converted to a different format using scripts. The whole process is not very user-friendly and there is a clear need for a better interface.

In our opinion, the interface should consist of two parts. There should be a database to store computed solution paths, data on the convergence of the solver and bifurcation diagrams. Furthermore, a graphical front-end is needed that serves several purposes: visualizing the results, preparing the input for the next run and managing the database. Nowadays PC's running the Microsoft Windows NT operating system start to enter the workstation market previously totally controlled by vendors of UNIX systems. The same might happen with Apple Macintosh compatible systems once Apple finishes its next-generation operating system code-named Rhapsody since the processor architecture used in the Macintosh is very powerful. Hence the interface should not only target UNIX systems but should also be portable to other platforms. Let us study the options for the database and GUI components separately.

The graphical user interface (GUI) has several tasks. First it should offer a user-friendly interface to the command file for the continuation code. It should assist in choosing good defaults for the various parameters in the algorithm and be useful for both "expert users" that understand the underlying algorithms and are capable to chose good values for all parameters of the algorithms, and normal users who have little understanding about the underlying code. The GUI should also allow to manage the data in the database, e.g., omit curves that are no longer needed, add comments to curves or solution points or extract data for use with other packages. Furthermore the GUI should allow to visualize the computed results. It should be possible to construct various types of diagrams easily:

- Bifurcation diagrams. These diagrams represent the solution or functions of the solution versus one or more varying parameters. It should be possible to construct several 2D and 3D projections of such a diagram, select colors for the curves or bifurcation points, etc.

- Individual solutions. It should be possible to visualize the solution profile or a cross-section in function of the time. It is also useful to draw phase plots showing the solution at one point in space in function of the solution at a different point at the same time or at some time in the past, either using a fixed delay, a delay that is a function of the period, or, in a DDE model, a delay that is a function of one of the delays in the model. Being able to draw plots using a delay that depends on the period is very useful to study space-time symmetries in solutions.

- Special diagrams. For instance, making a diagram of the evolution of the Floquet multipliers along the computed branch is very useful to study the stability of the branch.

It should be easy to extend the GUI to include other types of diagrams or to incorporate parameters for new solvers.

There exist various options to develop such a GUI.

- One can use low-level portable graphics and user interface libraries. The continuation package CONTENT is written using such a library (Vibrant, developed at the

National Center for Biotechnology Information of the National Institute of Health, U.S.A.). A low-level approach often allows to write very efficient programs. However, a lot of coding is required. A special option of this kind is the use of the Java environment. Java is a programming language with an extensive run-time library that allows the development of programs that are portable to virtually all modern operating systems. The source code is compiled into "byte-codes", a kind of machine code, and the byte-codes are executed in a virtual machine. The byte-codes are platform independent. The Java run-time libraries are rapidly evolving. Currently, the environment is very well suited for the development of user interfaces. It is also possible to draw 2D plots. 3D graphics APIs will be added in the near future. There are still performance problems on many platforms since the byte-codes are often interpreted. However, the performance of the virtual machines is improving rapidly. Also, compilers are becoming available that compile the Java source code to a regular system-specific binary. However, the Java solution requires a lot of coding, particularly for the visualization of solutions.

- At the other side of the spectrum, one finds packages for scientific visualization. Some of these packages allow the user to write a user interface and to interface with code written in C or some other programming languages. An example of such a package is AVS/Express from Advanced Visual Systems, available for the more popular UNIX platforms and Microsoft Windows 95 and NT . These packages offer very powerful visualization techniques. However, they are often very expensive and not available at many sites.

- Matlab version 5 is a good intermediate solution. It offers a very high-level interpreted programming language, provides enough functionality to develop a good GUI and also has some built-in visualization techniques. It can also easily interface with Fortran or C routines in a quite portable way. Matlab is already in use at many places and is available on many different platforms. Hence it is a very interesting option.

The second interface element is the database. The database must store the computed equilibrium and periodic solutions efficiently and their order on the computed branches. For large models, lossy compression techniques could be used as long as the decompressed orbits or fixed points do not differ to much from the original ones. The decompressed solutions must still contain all important details and be accurate enough to be used again as a starting value for the code. The database should also be able to store additional data for special points, such as quantities needed to jump to a different branch. Often a lot of time is spent in the construction of bifurcation diagrams and other plots. It may take some time before the ideal variables and projections have been determined to study the solution branches. Hence it should also be possible to store the bifurcation diagrams or other plots. The database should take into account that there exist various conventions to represent data as byte strings. This is important since often the continuation code will be run on a different computer as the GUI. And finally, the database should be accessible by the GUI front end during computations to check the progress.

There exist two options to implement the database.

- The database routines can be implemented as a library that is linked in the continuation code and the GUI program. The data reside in one or more files on a filesystem that is shared between the machine used to run the GUI and the machine on which the code is run. Since file access is an element of the ANSI C library, it is possible to develop a very portable library. However, there are problems to synchronize access to the database files in a robust and efficient way. Locking of files or records in files is not defined in ANSI C and hence is not portable. In fact, some operating systems do not even fully support locking. This approach is used in older PC database packages, but is more and more superseded by the next approach.

- One can also use a client-server approach. A server process controls the access to the data. Clients (the GUI and the continuation code) communicate with the server process and do not directly write into or read from the data files. Nowadays, most commercial database systems are based on this approach. The main disadvantage of this approach is that it is harder to make a portable implementation because of the interprocess communication. Mechanisms for interprocess communication over a network are not part of the standard C library. A Java application could be an interesting option, certainly if the Java virtual machine is integrated into the operating system and if the performance of the virtual machine is improved. However, Java does not yet offer a solution to the client-side part of the database system since there is no portable solution yet to call Java routines from other languages. An other option is to see whether one of the available database systems can be used. However, these are often very expensive software packages, and again, they are not available at all sites.

It is clear that writing a user-friendly interface around the continuation code is not a trivial task. However, it is a requirement if the code is to be used widely by less experienced users.

Bibliography

[1] J. Argyris, G. Faust, and M. Haase. *An Exploration of Chaos — An Introduction for Natural Scientists and Engineers.* North Holland Amsterdam, 1994.

[2] V.A. Arnol'd. *Geometrical Methods in the Theory of Ordinary Differential Equations*, volume 250 of *Grundlehren der mathematischen Wissenschaften.* Springer-Verlag, New York, 1983.

[3] U. Ascher, J. Christiansen, and R.D. Russel. Collocation software for boundary value ODE's. *ACM Trans. of math. software*, 7(2):209–222, 1981.

[4] U.M. Ascher, R.M.M. Mattheij, and R.D. Russell. *Numerical solution of boundary value problems for ordinary differential equations*, volume 13 of *Classics in Applied Mathematics.* SIAM, 1995.

[5] A. Back, J. Guckenheimer, M. Myers, F. Wicklin, and P. Worfolk. dstool: Computer assisted exploration ov dynamical systems. *Computers and Mathematics*, 39(4):303–309, 1992.

[6] Z. Bai and J.W. Demmel. On a direct algorithm for computing invariant subspaces with specified eigenvalues. Technical report. LAPACK working note 38.

[7] A.K. Bangia, P.F. Batcho, I.G. Kevrekidis, and G.E. Karniadakis. Unsteady two-dimensional flows in complex geometries: Comparative bifurcation studies with global eigenfunction expansions. *SIAM J. Sci. Comput.*, 18(3):775–805, 1997.

[8] R.E. Bank. *PLTMG, a software package for solving elliptic partial differential equations: users' guide 7.0*, volume 15 of *Frontiers in Applied Mathematics.* SIAM, 1994.

[9] R.E. Bank and T.F. Chan. PLTMGC: A multi grid continuation program for parameterized nonlinear elliptic systems. *SIAM J. Sci. Stat. Comput.*, 7:540–559, 1986.

[10] G. Berkooz, P. Holmes, and J.L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annu. Rev. Fluid Mech.*, 25:539–575, 1993.

[11] A. Bojanczyk, G. Golub, and P. Van Dooren. The periodic Schur decomposition. Algorithms and applications. In F.T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures and Implementations III*, volume 1170 of *Proceedings of the*

SPIE, the International Society of Optical Engineering, pages 31–42, Bellingham, Washington, U.S.A., 1992.

[12] H.S. Brown, M.S. Jolly, I.G. Kevrekidis, and E.S. Titi. Use of approximate inertial manifolds in bifurcation calculations. In D. Roose, B. De Dier, and A. Spence, editors, *Continuation and Bifurcation: Numerical Techniques and Applications*, volume 313 of *NATO ASI Series C: Mathematical and Physical Sciences.* Kluwer Academic Publishers, 1990.

[13] K. Burrage, J. Erhel, B. Pohl, and A. Williams. A deflation technique for linear systems of equations. *SIAM J. Sci. Comput.*, 1997. To appear.

[14] K. Burrage, A. Wiliams, J. Erhel, and B. Pohl. The implementation of a generalized cross validation algorithm using deflation techniques for linear systems. *Journal of Applied and Numerical Mathematics*, 19:17–31, 1996.

[15] A.M. Castelfranco and H.W. Stech. Periodic solutions in a model of recurrent neural feedback. *SIAM J. Appl. Math.*, 47(3):573–588, 1987.

[16] T. Chan. Deflation techniques and block-elimination algorithms for solving bordered singular systems. *SIAM J. Sci. Stat. Comput.*, 5(1):121–134, 1984.

[17] T. Chan. Techniques for large sparse systems arising from continuation methods. In Küpper et al. [70], pages 116–128.

[18] J.-P. Chebab and R. Temam. Incremental unknowns for solving nonlinear eigenvalue problems: New multiresolution methods. *Num. Meth. PDE's*, 11:199–228, 1995.

[19] B. Childs, M. Scott, J.W. Daniel, E. Denman, and P. Nelson, editors. *Codes for Boundary-Value Problems in Ordinary Differential Equations*, number 76 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1979.

[20] B.D. Davidson. Large scale continuation and numerical bifurcation for partial differential equations. *SIAM J. Numer. Anal.*, 34(5):2008–2027, 1997.

[21] P.J. Davis and P. Rabinowitz. *Numerical Integration.* Blaisdell, Waltham, Massachussets, 1967.

[22] C. de Boor and B. Swartz. Comments on a comparison of global methods for two-point boundary value problems. *Math. Comp.*, 31:916–921, 1977.

[23] A.E. Deane, I.G. Kevrekidis, G.E. Karniadakis, and S.A. Orszag. Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders. *Phys. Fluids A*, 3(10):2337–2354, 1991.

[24] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall Series in Computational Mathematics. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[25] P. Deuflhard. Nonlinear equation solvers in bondary value problem codes. In Childs et al. [19].

[26] P. Deuflhard and G. Bader. Multiple shooting techniques revisited. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, volume 2 of *Progress in Scientific Computing*. Birkhäuser Verlag, 1983.

[27] E. Doedel. AUTO: Software for continuation and bifurcation problems in ordinary differential equations. Report Applied Mathematics, California Institute of Technology, Pasadena, USA, 1986.

[28] E. Doedel, H.B. Keller, and J.P. Kernevez. Numerical analysis and control of bifurcation problems: (I) Bifurcation in finite dimensions. *International Journal of Bifurcation and Chaos*, 1(3):493–520, 1991.

[29] E. Doedel, H.B. Keller, and J.P. Kernevez. Numerical analysis and control of bifurcation problems: (II) Bifurcation in infinite dimensions. *International Journal of Bifurcation and Chaos*, 1(4):745–772, 1991.

[30] E.J. Doedel. Finite difference collocation methods for nonlinear two point boundary value problems. *SIAM J. Numer. Anal.*, 16(2):173–185, 1979.

[31] E.J. Doedel, A.R. Champneys, T.J. Fairgrieve, Y.A. Kuznetsov, B. Sandstede, and X.-J. Wang. AUTO97: Continuation and bifurcation software for ordinary differential equations. Technical report, Department of Computer Science, Concordia University, 1997.

[32] E.J. Doedel and P.P.C. Leung. A numerical technique for bifurcation problems in delay differential equations. *Congr. Num.*, 34:225–237, 1982.

[33] J.J. Dongarra, S. Hammarling, and J.H. Wilkinson. Numerical considerations in computing invariant subspaces. Technical Report CS-90-117, University of Tennessee, 1990. LAPACK working note 25.

[34] W.S. Edwards, L.S. Tuckerman, R.A. Friesner, and D.C. Sorensen. Krylov methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 110:82–102, 1994.

[35] J. Elezgaray and A. Arneodo. Crisis-induced intermittent bursting in reaction-diffusion chemical systems. *Physical Review Letters*, 68(5):714–717, 1992.

[36] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1996.

[37] T. Fairgrieve. *The computation and use of Floquet multipliers for bifurcation analysis.* PhD thesis, Department of Computer Science, University of Toronto, 1994.

[38] T. F. Fairgrieve and A. D. Jepson. O.K. Floquet multipliers. *SIAM J. Numer. Anal.*, 28(5):1446–1462, 1991.

[39] M. Farkas. *Periodic motions*, volume 104 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1994.

[40] U. Feudel and W. Jansen. CANDSYS/QA — A software system for qualitative analysis of nonlinear dynamical systems. *International Journal of Bifurcation and Chaos*, 2(4):773–794, 1992.

[41] M.J. Friedman and E.J. Doedel. Numerical computation and continuation of invariant manifolds connecting fixed points. *SIAM J. Numer. Anal.*, 28(3):789–808, 1991.

[42] B. Garcia-Archilla and J. de Frutos. Time integration of the non-linear Galerkin method. *IMA Journal of Numerical Analysis*, 15:221–244, 1995.

[43] P. Glendinning. *Stability, instability and chaos.* Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 1994.

[44] G.H. Golub and C.F. Van Loan. *Matrix computations*, volume 3 of *John Hopkins series in the mathematical sciences.* John Hopkins University Press, Baltimore, 1983.

[45] W. Govaerts, J. Guckenheimer, and A. Khibnik. Defining functions for multiple Hopf bifurcations. *SIAM J. Numer. Anal.*, 34(3):1269–1288, 1997.

[46] M.D. Graham and I.G. Kevrekidis. Alternative approaches to the Karhunen-Loève decomposition for model reduction and data analysis. *Comp. Chem. Eng.*, 20:495, 1996.

[47] M. Grinfeld. Dynamics of a model equation in viscoelasticity. In *Proceedings Equadiff 91*, Singapore, 1993. World Scientific.

[48] J. Guckenheimer and P. Holmes. *Nonlinear oscillations, dynamical systems and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1983.

[49] K.P. Hadeler. Effective computation of periodic orbits and bifurcation diagrams in delay equations. *Numer. Math.*, 34:457–467, 1980.

[50] J.K. Hale. *Theory of Functional Differential Equations*, volume 3 of *Applied Mathematical Science.* Springer-Verlag, 1977.

[51] B.D. Hassard, N. Kazarinoff, and Y.H. Wan. *Theory and applications of Hopf bifurcation.* Cambridge University Press, Cambridge, 1981.

[52] C. Hirsch. *Numerical computation of internal and external flows. Volume 1: fundamentals of numerical discretisation.* John Wiley and Sons, Chichester, 1988.

[53] M. Holodniok, P. Knedlik, and M. Kubiček. Continuation of periodic solutions in parabolic differential equations. In T. Küpper, R. Seydel, and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Applications*, ISNM 79, pages 122–130. Birkhäuser, Basel, 1987.

[54] H. Jarausch. Zur numerischen Untersuchung von parabolischen Differentialgleichungen mit Hilfe einer adaptiven spektralen Zerlegung. Habilitation Thesis, RWTH Aachen - Institut für Geometrie und praktische Mathematik, 1991.

[55] H. Jarausch. Analyzing stationary and periodic solutions of systems of parabolic partial differential equations by using singular subspaces as reduced basis. Bericht 92, RWTH Aachen - Institut für Geometrie und praktische Mathematik, December 1993.

[56] H. Jarausch and W. Mackens. Numerical treatment of bifurcation branches by adaptive condensation. In Küpper et al. [70], pages 296–309.

[57] H. Jarausch and W. Mackens. Computing bifurcation diagrams for large nonlinear variational problems. In P. Deuflhard and B. Enguist, editors, *Large Scale Scientific Computing*, volume 7 of *Progress in Scientific Computing*. Birkhäuser Verlag, 1987.

[58] H. Jarausch and W. Mackens. Solving large nonlinear systems of equations by an adaptive condensation process. *Numer. Math.*, 50(6):633–653, 1987.

[59] A. Jennings and W.J. Stewart. A simultaneous iteration algorithm for real matrices. *ACM Trans. of math. software*, 7:184–198, 1981.

[60] A. Jepson and A. Spence. Folds in solutions of two parameter systems and their calculation. Part I. *SIAM J. Numer. Anal.*, 22(2):347–368, 1985.

[61] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software.* Prentice Hall, Englewood Cliffs, 1989.

[62] J. Kaplan and J. Yorke. Ordinary differential equations which yield periodic solutions of differential-delay equations. *J. Math. Anal. Appl.*, 48:317–324, 1974.

[63] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. In P. H. Rabinowitz, editor, *Applications of Bifurcation Theory*, New York, 1977. Academic Press.

[64] H. B. Keller and H. Von Sosen. New methods in CFD: DAE and RPM. In W. H. Hui, Y.-K. Kwok, and J. R. Chasnov, editors, *Proceedings of the First Asian CFD Conference*, pages 17–30, Hong Kong, 1995.

[65] H.B. Keller. *Methods for two-point boundary value problems.* Blaisdell, 1968.

[66] H.B. Keller and A.D. Jepson. Steady state and periodic solution paths: their bifurcations and computations. In Küpper et al. [70], pages 219–246.

[67] A.I. Khibnik, Yu.A. Kuznetsov, V.V. Levitin, and E.N. Nikolaev. *LOCBIF, version 2: Interactive LOCal BIFurcation Analyzer.* CAN Expertise Centre, Amsterdam, 1992.

[68] A.I. Khibnik, Yu.A. Kuznetsov, V.V. Levitin, and E.N. Nikolaev. Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps. *Physica D*, 62:360–371, 1993.

[69] M. Kubiček and M. Holodniok. Algorithms for determination of period-doubling bifurcation points in ordinary differential equations. *Journal of Computational Physics*, 70:203–217, 1987.

[70] T. Küpper, H. D. Mittelmann, and H. Weber, editors. *Numerical Methods for Bifurcation Problems*, volume 70 of *ISNM*. Birkhäuser, Basel, 1984.

[71] Y.A. Kuznetsov. *Elements of applied bifurcation theory*, volume 112 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1995.

[72] Yu.A. Kuznetsov and V.V. Levitin. CONTENT, a multiplatform continuation environment. Tecnical report, CWI, Amsterdam, The Netherlands, 1996.

[73] M. Lentini and V. Pereyra. An adaptive finite-difference solver for nonlinear two-point boundary value problems with mild boundary layers. *SIAM J. Numer. Anal.*, 14:91–111, 1977.

[74] K. Lust and D. Roose. Newton–Picard methods with subspace iteration for computing periodic solutions of partial differential equations. *Zeitschrift für Angewandte Mathematik und Mechanik*, pages 605–606, 1996. Supplement 2 (Applied Analysis) of the special volumes devoted to the proceedings of ICIAM 95.

[75] K. Lust, D. Roose, A. Spence, and A.R. Champneys. An adaptive Newton–Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.*, 1996. Accepted for publication.

[76] T. Luzyanina, K. Engelborghs, K. Lust, and D. Roose. Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations. *International Journal of Bifurcation and Chaos*, 1997. To appear.

[77] T. Luzyanina and D. Roose. Numerical stability analysis and compuation of Hopf bifurcation points for delay differential equations. *Journal of Computational and Applied Mathematics*, 72:379–392, 1996.

[78] M.J. Maron. *Numerical Analysis. A Practical Approach.* Macmillan Publishing Company, New York, second edition, 1987.

[79] R.M.M. Mattheij and G.W.M. Staarink. Boundpak: Numerical software for linear boundary value problems, part one. EUT Report 92-WSK-01, Eindhoven University of Technology, The Netherlands, 1992.

[80] R.M.M. Mattheij and G.W.M. Staarink. Boundpak: Numerical software for linear boundary value problems, part two. EUT Report 92-WSK-01, Eindhoven University of Technology, The Netherlands, 1992.

[81] K. Meerbergen and D. Roose. Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems. *IMA Journal of Numerical Analysis*, 16(3):297–346, 1996.

[82] C.B. Moler and G.W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10(2):241–256, 1973.

[83] G. Moore and A. Spence. The calculation of turning points of nonlinear equations. *SIAM J. Numer. Anal.*, 17(4):567–576, August 1980.

[84] J.J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical programming — the State of the Art*, pages 258–287, Bonn, 1982. Springer Verlag, Berlin, 1983.

[85] S. Nakamura. *Applied numerical methods with software.* Prentice-Hall International, Inc., London, 1991.

[86] R. Neubert. Predictor-corrector techniques for detecting Hopf bifurcation points. *International Journal of Bifurcation and Chaos*, 3(5):1311–1318, 1993.

[87] R. Neubert. *Über die Approximation van Lösungszweigen und die Entdeckung von Hopfschen Verzweigungspunkten in nichtlinearen Gleichungssystemen mit einem Parameter.* PhD thesis, Fakultät für Mathematik und Wirtschaftswissenschaften der Universität Ulm, 1993.

[88] K.-G. Nolte and I. L'Heureux. On the numerical detection of codimension-3 bifurcations of periodic solutions (with applications to optical bistability). *International Journal of Bifurcation and Chaos*, 4(6):1425–1446, 1994.

[89] R.D. Nussbaum. Periodic solutions of nonlinear autonomous functional differential equations. In H.-O. Peitgen and H.-O. Walther, editors, *Functional Differential Equations and Approximation of Fixed Points*, volume 730 of *Lecture Notes in Mathematics*, pages 283–325. Springer-Verlag, 1978.

[90] W.E. Olmstead, S.H. Davis, S. Rosenblat, and W.L. Kath. Bifurcations with memory. *SIAM J. Appl. Math.*, 40(2):171–188, 1986.

[91] V. Pereyra. PASVA3 — An adaptive finite-difference Fortran program for first-order nonlinear ordinary boundary problems. In Childs et al. [19], pages 67–88.

[92] G. Peters and J.H. Wilkinson. Eigenvectors of real and complex matrices by LR and QR triangularizations. *Numer. Math.*, 16:181–204, 1970.

[93] M.J.D. Powell. A hybrid method for nonlinear algebraic equations. In P. Rabinowitz, editor, *Numerical methods for nonlinear algebraic equations*, pages 87–114. Gordon and Breach, London, 1980.

[94] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical recipes.* Cambridge University Press, 1987.

[95] K. Radhakrishnan and A.C. Hindmarsh. Description and use of LSODE, the Livermore Solver for Ordinary Differential Equations. LLNL Report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1994. NASA Reference Publication 1327.

[96] W.C. Rheinboldt. *Numerical analysis of parametrized nonlinear equations*, volume 7 of *University of Arkansas Lecture Notes in the Mathematical Sciences.* Wiley-Interscience Publication, New York, 1986.

[97] W.C. Rheinboldt and J. Burkard. A locally parameterized continuation process. *ACM Trans. of math. software*, 9:215–235, 1983.

[98] D. Roose, K. Lust, A. Champneys, and A. Spence. A Newton–Picard shooting method for computing periodic solutions of large-scale dynamical systems. *Chaos, Solitons & Fractals*, 5(10):1913–1925, 1995.

[99] D. Roose and S. Vandewalle. Efficient parallel computation of periodic solutions of parabolic partial differential equations. In R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, editors, *Bifurcation and Chaos: Analysis, Algorithms, Applications*, volume 97 of *ISNM*, pages 307–317. Birkhäuser, Basel, 1991.

[100] Y. Saad. Analysis of some Krylov subpspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29:209–228, 1992.

[101] Y. Saad. *Numerical methods for large eigenvalue problems.* Algorithms and architectures for advanced scientific computing. Manchester University Press, Manchester, 1992.

[102] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[103] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1086.

[104] R. Seydel. Tutorial on continuation. *International Journal of Bifurcation and Chaos*, 1(1):3–11, 1991.

[105] R. Seydel. BIFPACK — a program package for calculating bifurcations. Buffalo 1983. Current version 3.0. Ulm, 1994.

[106] R. Seydel. *Practical Bifurcation and Stability Analysis. From Equilibrium to Chaos.* Springer-Verlag, New York, second edition, 1994.

[107] G. Shroff and H.B. Keller. Stabilization of unstable procedures: the recursive projection method. *SIAM J. Numer. Anal.*, 30(4):1099–1120, 1993.

[108] Inertial manifolds in scientific computing. *SIAM News*, 24(5):8-9, 1991.

[109] H.W. Stech. A numerical analysis of the structure of periodic orbits in autonomous functional differential equations. In S.-N. Chow and J.K. Hale, editors, *Dynamics of Infinite Dimensional Systems*, volume F37 of *NATO ASI Series*, pages 325–337. Springer-Verlag, 1987.

[110] G.W. Stewart. Simultaneous iteration for computing invariant subspaces of non-hermitian matrices. *Numer. Math*, 25:123–136, 1976.

[111] J. Stoer and R. Bulirsch. *Einführung in die Numerische Mathematik II.* Heidelberger Taschenbücher. Springer-Verlag, Berlin, 1973.

[112] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, 1980.

[113] M.A. Taylor and I.G. Kevrekidis. Interactive AUTO: A graphical interface for AUTO86. Technical report, Department of Chemical Engineering, Princeton University, 1989.

[114] S. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems.* B.G. Teubner Verlag, Stuttgart, 1993.

[115] H. Von Sosen. *Part I: Folds and bifurcations in the solutions of semi-explicit differential-algebraic equations. Part II: The Recursive Projection Method applied to differential-algebraic equations and incompressible fluid mechanics.* PhD thesis, California Institute of Technology, Pasadena, California, 1994.

[116] H.-O. Walther. A theorem on the amplitudes of periodic solutions of differential delay equations with application to bifurcation. *J. Diff. Eqns.*, 39:369–404, 1978.

[117] R. Weiss. The application of implicit Runge–Kutta and collocation methods to boundary value problems. *Math. Comp.*, 28:449–464, 1974.

[118] A. Williams and K. Burrage. A parallel implementation of a deflation algorithm for systems of linear equations. Research Report 94-12, ETHZ Seminar für Angewandte Mathematik, 1994.

[119] A. Williams and K. Burrage. Surface fitting using GCV smoothing splines on supercomputers. In *Proceedings of Supercomputing*, 1995.

[120] S.M. Zoldi and H.S. Greenside, 1997. Private communication.

[121] S.M. Zoldi and H.S. Greenside. Spatially localized unstable periodic orbits. Technical report, Department of Physics, Duke University, Durham, 1997.