

DISS. ETH NO. 15223

Preconditioned Arnoldi Methods for Systems of Nonlinear Equations

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
LEONHARD JASCHKE

Dipl. Informatik-Ing. ETH
born on March 24, 1971
citizen of Austria

accepted on the recommendation of
Prof. Dr. Walter Gander, examiner
Prof. Dr. Martin Gutknecht, co-examiner

2003

to Eva H. Tichy

Acknowledgement



Whenever a big project comes to an end that lasted for several years, one can be sure that it is not only the result of one person's work. My PhD thesis makes no difference to this observation. Many people have supported me. Without their help I would not have been able to finish my thesis. This section is dedicated to all these people.

In the first place I want to thank my thesis advisor *Prof. Dr. Walter Gander* for giving me the opportunity to work in the field of systems of nonlinear equations in his research group. He helped me from finding the right topic to proof reading. He invited me to a one week stay in Oxford, he always believed in me, and he supported me in many ways. I am also obliged to him for introducing me to all the important people in our area.

I am grateful to my co-advisor *Prof. Dr. Martin Gutknecht* for providing me with his excellent knowledge in all kinds of numerical methods. He read my thesis very carefully and pointed out some important improvements in both science and language. His scientific tips and his general technical hints (L^AT_EX, scientific programs, Unix, etc.) were very valuable.

My colleagues at the Institute of Scientific Computing also gave me very valuable hints. Special thanks go to *Rolf Strebel* and *Oscar Chinellato* for helping me understand the most complex mathematical problems. Whene-

ver I had a mathematical problem I could talk to them about it and they always tried to help in the most constructive way. Another special thanks goes to *Roman Geus*, my office mate, for providing me with his excellent knowledge on all kinds of topics in computer science. His contributions to my thesis include his \LaTeX thesis environment, helping me with PDF- \LaTeX , lending me a number of books several times. Among these were the \LaTeX -companion and Saad's book on Krylov space methods which I extensively needed. He also has been a competent person to consult when I wanted to buy some new hardware (eg. Palm, DVD/CD-Burner). Even more special thanks go to *Oliver Bröker*, most of all for encouraging me to attend a course in didactics. In several discussions and classes for lectures we developed together our knowledge in expressing complex thoughts understandably. If people are able to understand the thoughts issued in this thesis, it is in part Oliver's merit. He also helped me with English and technical questions (WWW- and \LaTeX -stuff). All my work mates at the institute have been good friends to me. We had a lot of fun also beyond our work, for example when going skiing together or sledging, when going to the movies or when going out for dinner or lunch. The friends I met at the ETH made this time an unforgettable and precious section of my life. Thank you all for your friendship.

I want to thank *Dr. Walter Egli* whose scientific problems at ABB inspired me to chose the topic. He helped me with all questions concerning the applications for my algorithms at ABB. He has not only been a fine colleague to work with but also a good friend.

Let me thank *Silvio Merazzi* from SMR Engineering & Development who provided me with the two programs ASTRID/B2000 and MEMCOM for free, so that I could test my algorithms on the real world example of the ABB company described in detail in Chapter 5.

Prof. Jörg Waldvogel always encouraged me to write the thesis. When I asked him to help me with a problem


described in Chapter 3 that had kept me busy for more than a month, he was able to find a solution within one week. The solution he found is a very nice contribution to my PhD thesis. I want to thank my parents *Erna* and *Otwin*, my sister *Verena*, my Grandmother *Erna* and the rest of my family who always believed in me. I could always count on them and they supported me in every way in what I was doing.

So far my thesis is the most important piece of work I have ever done, and to me *Eva Tichy* is the most important person I have ever met. Throughout the last four years — the time when I was writing the thesis — her vivid and passionate nature has been capturing most of my thoughts, my feelings and my dreams. This is why I dedicate my thesis to her.

DEDICATION

“Immer diese Rechnungen”
— Donald Duck[30]

Zusammenfassung

rylov-Raum-Methoden zum Lösen linearer Gleichungssysteme haben die Eigenschaft, dass sie implizit das minimale Polynom der Systemmatrix in bezug auf den ersten Residuumvektor bestimmen.

Vektorextrapolationsmethoden werden verwendet, um die Konvergenz von Vektorsequenzen zu beschleunigen. Sie benötigen keine Information, wie eine solche Sequenz erzeugt wurde. Wenn aber die Sequenz durch eine lineare Iteration erzeugt wurde, kann gezeigt werden, dass der extrapolierte Grenzwert einer Vektorextrapolationsmethode die Lösung eines linearen Gleichungssystems ist. Vektorextrapolationsmethoden approximieren explizit die Koeffizienten des minimalen Polynoms dieses Gleichungssystems ohne Zuhilfenahme der Systemmatrix und des Vektors auf der rechten Seite. Es ist bereits bekannt, dass diese Methoden und Krylov-Raum-Methoden in exakter Arithmetik dieselben Iterierten erzeugen, falls sie zum Lösen von linearen Gleichungssystemen verwendet werden.

Die explizite Bestimmung der Koeffizienten des minimalen Polynoms ist numerisch nicht stabil. Krylov-Raum-Methoden verwenden einen genaueren, impliziten Ansatz. Auf der anderen Seite aber sind Vektorextrapolationsmethoden im Stande, *nichtlineare Gleichungssysteme zu lösen*, die einen dominanten linearen Anteil in der Nähe des zu extrapolierenden Grenzwertes aufweisen.

Der erzeugende Prozess der Vektorsequenz einer Vektorextrapolationsmethode kann im Krylov-Raum-Kontext als Vorkonditionierer angesehen werden. Unter Verwendung der zentralen Idee von Vektorextrapolationsmethoden (Differenzvektoren der Sequenzvektoren) kann der Vorkonditionierer um eine rekursive Vorschrift erweitert werden, um die Matrix-Vektor-Multiplikation von wiedergestarteten Krylov-Raum-Methoden zu ersetzen. Auf diese Weise können Krylov-Raum-Methoden zum Lösen linearer Gleichungssysteme auf das Lösen nichtlinearer Gleichungssysteme adaptiert werden. Sie besitzen dann ähnliche Konvergenzeigenschaften wie Vektorextrapolationsmethoden.


er wissenschaftliche Hauptbeitrag dieser Doktorarbeit ist die Entwicklung und Umsetzung dieses Ansatzes für die allgemeinen Arnoldi-Methoden *FOM* und *GMRES*. Dadurch erhalten wir zwei neue Methoden zum Lösen nichtlinearer Gleichungssysteme — *vorkonditionierte Arnoldi-Methoden zum Lösen nichtlinearer Gleichungssysteme*, die für gewisse Problemstellungen eine bessere Leistung erbringen können als etablierte Methoden wie z.B. inexakte Newton-Methoden. Die neuen Algorithmen wurden anhand der Chandrasekhar H-Gleichung und einem Wärmestrahlungsproblem des Forschungszentrums der Firma ABB in Dättwil, Schweiz getestet.

Ein weiterer Beitrag dieser Doktorarbeit ist die Entwicklung einer allgemeinen Theorie für Vektorextrapolationsmethoden. Diese kann auch auf die neu entwickelten Algorithmen angewendet werden. Die Hauptaussage dieser Theorie ist: Vektorextrapolationsmethoden sind Implementationen der Methode von Henrici (eine Verallgemeinerung der Methode von Steffensen auf Vektorsequenzen). Neu in dieser Theorie ist die Formulierung eines Kontorovich-Theorems für die Methode von Henrici, das die Bedingungen für ihre Konvergenz festlegt. Aufgrund unserer geometrischen Untersuchungen waren wir ausserdem in der Lage, eine Klasse von

skalaren Iterationsvorschriften zu beschreiben, für welche die Schmidt-Shanks-Transformation *in nur einem Schritt* konvergiert.

In dieser Doktorarbeit werden die wichtigsten herkömmlichen Verfahren zum Lösen nichtlinearer Gleichungssysteme beschrieben. Überdies wird ein Überblick über die Theorie der linearen Krylov-Raum-Methoden gegeben. Beispiele und Graphiken illustrieren unsere Ausführungen.

Summary

 Krylov space methods for solving a nonsingular linear system of equations are known to have the property to determine implicitly the *minimal polynomial* of the system matrix with respect to the initial residual vector, i.e. a polynomial of minimum degree, which is annihilated by the system matrix of the linear system when multiplied with the first residual vector.

Vector extrapolation methods have the purpose to accelerate the convergence of a vector sequence. No knowledge is needed on how the vector sequence is generated. However, if the sequence is generated implicitly by a linear iteration, then the extrapolated limit of the sequence is the fixed-point of a linear system. Without making use of the system matrix and the right-hand side of this system these methods explicitly estimate the coefficients of the minimal polynomial of the system matrix with respect to the first residual vector. It is already a known fact that vector extrapolation methods and corresponding Krylov space methods generate the same iterates in exact arithmetic for linear systems of equations.

The explicit determination of the coefficient of the minimal polynomial is not a stable process. The implicit Krylov approach is better. On the other hand vector extrapolation methods are capable of solving systems of

nonlinear equations with a dominant linear part in the area of interest.

The generating process of the vector sequence of vector extrapolation methods can be regarded as a preconditioner in terms of Krylov space methods. By using the central ideas of vector extrapolation methods (taking differences of sequence elements) the preconditioner can be expanded into a recursive scheme to replace the matrix vector multiplication in a *restarted Krylov method*. This way common linear Arnoldi methods can be extended to nonlinear ones, which have the same convergence properties as vector extrapolation methods.

The main scientific contribution of this thesis is the development and implementation of this approach for the general Arnoldi type Krylov space methods *FOM* and *GMRES*. Thus we get two new methods for solving systems of nonlinear equations — *preconditioned Arnoldi methods for solving systems of nonlinear equations*. For particular problems they have a better performance than common methods, e.g. *inexact Newton methods*. We have tested the new algorithms on the Chandrasekhar H-equation and a heat transfer problem problem of the research center of the ABB company in Dättwil, Switzerland.

Another contribution of this thesis is the development of a general theory of vector extrapolation methods, which can be also applied to the new methods. The main statement of the theory is that vector extrapolation methods are implementations of *Henrici's method* (a generalization of Steffensen's method to vector sequences). New in this thesis is also a Kantorovich like theorem for Henrici's method stating the conditions for its convergence. Furthermore we examined the geometric properties of the Schmidt-Shanks transformation (another generalization of Steffensen's method) and were thus able to specify a class of scalar iterations for which this transformation converges within *only one* step.

Several common methods for solving systems of nonlin-

ear equations are reviewed in this thesis as well as the theory of Krylov space methods. Our considerations are illustrated by many examples and figures.

Contents

1	Introduction	25
1.1	Overview	32
1.2	Notation and Preliminaries	35
1.2.1	Terminology	35
1.2.2	Typesetting	36
1.2.3	Matrix Properties	37
1.2.4	Matrix Decompositions	38
1.2.5	Norms	39
1.2.6	Inverses	42
1.2.7	Sets	43
1.2.8	Derivatives	43
1.2.9	Mean Value Theorem	45
1.2.10	Iterative Methods	46
1.2.11	Limit and Anti-Limit	48
1.2.12	Types of Convergence	48
1.2.13	Spectral Radius	50
1.2.14	Contraction Mapping	51
1.2.15	Standard Assumptions	57
1.3	Existing Techniques	61
1.3.1	Newton's Method	61

1.3.2	Quasi-Newton methods	63
1.3.3	Inexact Newton methods	69
1.3.4	Classical Methods	75
1.4	A Sample Problem	84
1.5	Conclusion: Numerical Results	86
2	Linear Krylov Space Methods	91
2.1	Motivation	91
2.2	Introduction	91
2.3	Correction Space Methods	93
2.3.1	Projection Methods	96
2.4	Krylov Space Methods	116
2.4.1	Motivation	116
2.4.2	Arnoldi Process	120
2.4.3	Symmetric Krylov Space Generation	123
2.4.4	Lanczos Biorthogonalization	124
2.4.5	General Arnoldi Type Methods	127
2.4.6	General Lanczos Type Methods	133
3	Vector Extrapolation Methods	147
3.1	Introduction	147
3.2	General Theory	148
3.2.1	Mathematical Equivalence to Linear Krylov Space Methods	155
3.2.2	Choosing the Projection Space \mathcal{P}_j	157
3.2.3	$\Delta_j(x_0)$ Base Construction	158
3.3	Nonlinear Behavior	158
3.3.1	One Dimensional Motivation: Steffensen's Method	160
3.3.2	Multivariate Case	170
3.4	Methods Based on Polynomial Extrapolation	185

3.4.1	Minimal Polynomial Extrapolation	185
3.4.2	Reduced Rank Extrapolation	187
3.5	Epsilon Algorithms	192
3.5.1	Schmidt-Shanks Transform	193
3.5.2	Scalar Epsilon Algorithm	201
3.5.3	Generalized Schmidt-Shanks Transform	203
3.5.4	Topological Epsilon Algorithm	205
3.6	Conclusion	210
4	Nonlinear Krylov Space Methods	215
4.1	Introduction	215
4.2	Arnoldi Type Methods	217
4.2.1	Full Orthogonalization Method	224
4.2.2	General Minimum Residual Method	225
4.3	Lanczos Type Methods	225
4.4	Conclusion	227
5	Application	231
5.1	Introduction	231
5.2	ABB's Deglor Oven	232
5.2.1	Physical Model	233
5.2.2	Discretization	236
5.2.3	Simple Model: Exterior Region Only	240
5.2.4	Complex Model: Exterior and Interior Regions	242
5.3	Numerical Results	244
5.4	Conclusion	248


program PhDthesis;

begin

repeat ...

Chapter 1

Introduction

 In this thesis we adapt iterative methods for solving linear systems of equations such that they are able to find solutions of nonlinear systems of equations. This is not an unusual approach. In fact most of the iterative methods for solving nonlinear systems of equations known today use the idea that a given nonlinear operator behaves linearly in the vicinity of a solution. Even one of the oldest iterative methods is of that kind.

Example 1.1 Let us consider the following unconstrained nonlinear problem

EXAMPLE:
NEWTON'S
METHOD

$$\varphi(\mathbf{s}) = \mathbf{0}. \quad (1.1)$$

In this equation φ is a nonlinear operator $\mathbb{R}^n \rightarrow \mathbb{R}^n$, of which a zero \mathbf{s} has to be found. *Newton's method* does such a job by supposing that φ behaves linearly in the vicinity of every approximation \mathbf{s}_i for the solution. Given an approximate guess \mathbf{s}_0 for the solution of (1.1) Newton's method thus replaces the original nonlinear operator φ by a linear operator $\gamma(\mathbf{x}) := \varphi(\mathbf{s}_0) + \mathbf{D}\varphi(\mathbf{s}_0)(\mathbf{x} - \mathbf{s}_0)$, which is the best linear approximation of φ at \mathbf{s}_0 . If the linear system $\gamma(\mathbf{s}) = \mathbf{0}$ is nonsingular, its solution \mathbf{s}_1 is unique

and can be determined by a method for solving a linear system of equations. The pool of such methods is huge. However, due to the fact that γ is only an approximation to φ , s_1 is generally not the solution of (1.1). But in general it is a better approximation to s than s_0 was. Repeating the described procedure one can hope that in every step the method computes successively better approximations to s . *Newton's method* is not only known to do this job very well, but also its convergence rate is usually quadratic.

While Newton's approach is problem oriented in that the problem is linearized locally and then the local problem is solved by a linear solver, our approach is method oriented: By examining a large class of linear solvers — Krylov space methods — we adapted these solvers such that they are capable of solving nonlinear problems.

Such an approach has already been successfully applied to

- classical iterative methods for linear systems, like the *Jacobi method*, *Gauss Seidel iteration*, *successive overrelaxation (SOR)*,
- vector extrapolation methods, which have been developed originally to speedup the convergence of linearly generated vector sequences. However, people have already realized, that they are also useful for accelerating vector sequences generated by a nonlinear process.

Sporadically linear Krylov space methods have been employed to solve nonlinear systems of equations. One example for these approaches is the *conjugate gradient (CG) method*. We will shortly describe this method as a case study for adjusting a method solving a linear system for solving nonlinear systems of equations.

EXAMPLE:
CONJUGATE
GRADIENTS

Example 1.2 Originally the CG method has been described by Stiefel and Hestenes (18) in 1952 as an iterative approach to solve a positive definite linear system of equations in at most n steps — n denoting the dimension of the system matrix — if no

rounding errors are involved. Their point of view of the method is an optimization of the nonlinear functional

$$\begin{aligned}\varphi : \mathbb{R}^n &\rightarrow \mathbb{R}, \\ \mathbf{x} &\mapsto \mathbf{x}^H \mathbf{A} \mathbf{x} - 2\mathbf{b}^H \mathbf{x}.\end{aligned}$$

In this functional $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix and $\mathbf{b} \in \mathbb{R}^n$ is a vector. To find the minimum of φ with respect to \mathbf{x} one has to find the zero of its gradient. Fortunately, the gradient of this functional is

$$\nabla \varphi(\mathbf{x})^H = 2\mathbf{A}\mathbf{x} - 2\mathbf{b} =: -2\mathbf{r}(\mathbf{x}).$$

That is: To find a zero of the gradient the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ has to be solved. Or: *To find the solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ the minimum of φ has to be determined.*

Hestenes and Stiefel used $\mathbf{s}_k := \mathbf{s}_{k-1} + \alpha_{k-1} \mathbf{p}_{k-1}$ as a general ansatz to determine an approximate solution \mathbf{s}_k in each step of the method. The search direction \mathbf{p}_k , that updates the approximate solution in every step, was defined by

$$\mathbf{p}_k := \mathbf{r}_k + \beta_{k-1} \mathbf{p}_{k-1}, \quad (1.2)$$

with $\mathbf{p}_0 := \mathbf{r}_0$. By definition of \mathbf{s}_k the residual of the linear system to be solved satisfies

$$\mathbf{r}_k := \mathbf{r}_{k-1} - \alpha_{k-1} \mathbf{A} \mathbf{p}_{k-1}. \quad (1.3)$$

Minimizing φ in the search direction \mathbf{p}_{k-1} leads to the condition

$$\mathbf{r}_k^H \mathbf{p}_{k-1} = 0. \quad (1.4)$$

It turns out that much work can be saved if the search directions are \mathbf{A} -conjugate, which is mathematically

$$\mathbf{p}_i^H \mathbf{A} \mathbf{p}_j = 0, \quad \forall i > j. \quad (1.5)$$

Using (1.4) α_k can be determined. We have

$$\begin{aligned} 0 &\stackrel{!}{=} \mathbf{r}_{k+1}^H \mathbf{p}_k \\ &= \mathbf{r}_k^H \mathbf{p}_k - \alpha_k \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k \\ &\stackrel{(1.2)}{=} \mathbf{r}_k^H \mathbf{r}_k + \beta_{k-1} \underbrace{\mathbf{r}_k^H \mathbf{p}_{k-1}}_{=0} - \alpha_k \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k \end{aligned}$$

Hence $\alpha_k = (\mathbf{r}_k^H \mathbf{r}_k) / (\mathbf{p}_k^H \mathbf{A} \mathbf{p}_k)$. Using this result we also find that

$$\begin{aligned} \mathbf{r}_{k+1}^H \mathbf{r}_k &= \mathbf{r}_k^H \mathbf{r}_k - \alpha_k \mathbf{p}_k^H \mathbf{A} \mathbf{r}_k \\ &\stackrel{(1.2)}{=} \mathbf{r}_k^H \mathbf{r}_k - \alpha_k \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k + \alpha_k \beta_k \underbrace{\mathbf{p}_{k-1}^H \mathbf{A} \mathbf{p}_k}_{=0} \\ &= \mathbf{r}_{k+1}^H \mathbf{p}_k = 0. \end{aligned} \tag{1.6}$$

Roughly in the same manner β_k can be determined using (1.5).

$$\begin{aligned} 0 &\stackrel{!}{=} \mathbf{p}_{k+1}^H \mathbf{A} \mathbf{p}_k \\ &= \mathbf{r}_{k+1}^H \mathbf{A} \mathbf{p}_k + \beta_k \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k \\ &\stackrel{(1.3)}{=} \alpha_k^{-1} \underbrace{\mathbf{r}_{k+1}^H \mathbf{r}_k}_{=0} - \alpha_k^{-1} \mathbf{r}_{k+1}^H \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k \\ &= \left(-\frac{\mathbf{r}_{k+1}^H \mathbf{r}_{k+1}}{\mathbf{r}_k^H \mathbf{r}_k} + \beta_k \right) \mathbf{p}_k^H \mathbf{A} \mathbf{p}_k. \end{aligned}$$

Hence $\beta_k = (\mathbf{r}_{k+1}^H \mathbf{r}_{k+1}) / (\mathbf{r}_k^H \mathbf{r}_k)$, conversely, it has been proved by several authors that choosing these α_k and β_k implies (1.5).

The name of the method is somehow misleading, since the residuals are the actual gradients of the functional to be minimized. And, as (1.6) shows, they are rather orthogonal than conjugate. However, the directions \mathbf{p}_k are conjugate. Since by (1.2) \mathbf{p}_k is a linear combination of $\mathbf{r}_0, \dots, \mathbf{r}_k$, the vectors $\mathbf{p}_0, \dots, \mathbf{p}_k$ and $\mathbf{r}_0, \dots, \mathbf{r}_k$ span the same space. That is: the direction vectors are *conjugated* versions of the gradients. This caused Shewchuk to formulate the following statement: "The name 'Conjugate Gradients' is a bit of a misnomer, because

the gradients are not conjugate, and the conjugate directions are not all gradients. ‘Conjugated Gradients’ would be more accurate” (36, p. 29).

The *linear conjugate gradient* method shown in Algorithm 1.1 sums up all the considerations made so far. This functional expects as arguments the linear system specified by \mathbf{A} and \mathbf{b} , an initial guess for the solutions \mathbf{s}_0 and a desired tolerance ε that the relative error of the returned result has to meet.

Algorithm 1.1 Linear Conjugate Gradient Method.

```

function CG( $\mathbf{A}$ ,  $\mathbf{b}$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ):vector;
   $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$ ;
   $\mathbf{p}_0 := \mathbf{r}_0$ ;
   $i := 0$ ;
  while  $\|\mathbf{r}_i\| > \varepsilon \|\mathbf{b}\|$  do
     $\alpha_i := (\mathbf{r}_i^H \mathbf{r}_i) / (\mathbf{p}_i^H \mathbf{A} \mathbf{p}_i)$ ;
     $\mathbf{s}_{i+1} := \mathbf{s}_i + \alpha_i \mathbf{p}_i$ ;
     $\mathbf{r}_{i+1} := \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{p}_i$ ;

     $\beta_i := (\mathbf{r}_{i+1}^H \mathbf{r}_{i+1}) / (\mathbf{r}_i^H \mathbf{r}_i)$ ;
     $\mathbf{p}_{i+1} := \mathbf{r}_{i+1} + \beta_i \mathbf{p}_i$ ;
     $i := i + 1$ 
  end;
  CG :=  $\mathbf{s}_i$ 
end CG;

```

Applying this method to solve a nonlinear system of equations of type (1.1) the linear residual $\mathbf{r}(\mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{x}$ has to be replaced by the nonlinear residual $\varphi(\mathbf{x})$. Thus the computation (1.3) is replaced by

$$\mathbf{r}_k := \varphi(\mathbf{s}_{k-1} + \alpha_{k-1} \mathbf{p}_{k-1}).$$

However, because (1.3) implies $\mathbf{A} \mathbf{p}_{k-1} = -\alpha_{k-1}^{-1} (\mathbf{r}_k - \mathbf{r}_{k-1})$, we still use that equation to replace Condition (1.5) by

$$\mathbf{p}_i^H (\mathbf{r}_{j+1} - \mathbf{r}_j) = 0, \quad \forall i > j.$$

Since α_k cannot be determined by a linear equation, it has to be determined by the scalar nonlinear equation

$$\mathbf{p}_k^H \varphi(\mathbf{s}_k + \alpha_k \mathbf{p}_k) = 0. \quad (1.7)$$

Solving this equation is called a *line search*. The user of the nonlinear CG method has to provide a procedure that finds the solutions to this problem and decides which of these shall be chosen. Any algorithm that finds these solutions can be used.

Also the choice of β_k is not unique anymore. One can suppose that $\mathbf{r}_{k+1}^H \mathbf{r}_k = 0$ still holds (Fletcher-Reeves approach) or not (Polak-Ribière approach). In either case β_k becomes

$$\beta_k^{\text{FR}} := \frac{\mathbf{r}_{k+1}^H \mathbf{r}_{k+1}}{\mathbf{r}_k^H \mathbf{r}_k} \quad (\text{Fletcher-Reeves}), \text{ or}$$

$$\beta_k^{\text{PR}} := \frac{\mathbf{r}_{k+1}^H (\mathbf{r}_{k+1} - \mathbf{r}_k)}{\mathbf{r}_k^H \mathbf{r}_k} \quad (\text{Polak-Ribière}).$$

It is known that the Fletcher-Reeves approach converges if the starting point is sufficiently close to the solution. The Polak-Ribière approach converges often much more quickly. However, in rare cases, it can cycle infinitely without converging (36, p. 42). Fortunately, convergence of the Polak-Ribière approach can be guaranteed by choosing $\beta_k = \max \{ \beta_k^{\text{PR}}, 0 \}$. However, the best choice for β_k still remains a challenge in numerical research.

Two things are notable about the transformation from the linear case to the nonlinear case.

1. The nonlinear functional to be minimized is used implicitly. Its knowledge is not required to implement the method.
2. An important property of the linear CG method is that the system matrix of the linear system to be solved has not to be specified explicitly. It can be given implicitly as a function that returns a matrix vector product. Similarly the nonlinear version needs a function call that returns the value of $\varphi(\mathbf{x})$. However, additionally a line search procedure must be provided that solves the scalar problem (1.7). Nonetheless this is an improvement compared to Newton's method requiring the knowledge of the derivative of φ .

Algorithm 1.2 shows an implementation of the nonlinear conjugate gradient method.

Algorithm 1.2 Nonlinear Conjugate Gradient Method.

```

function nlCG( $\varphi()$ , linesearch(),  $s_0$ ,  $\varepsilon$ ):vector;
   $r_0 := \varphi(s_0)$ ;
   $p_0 := r_0$ ;
   $i := 0$ ;
  while  $\|r_i\| > \varepsilon \|\varphi(0)\|$  do
     $\alpha_i := \text{linesearch}(s_i, p_i)$ ;
     $s_{i+1} := s_i + \alpha_i p_i$ ;
     $r_{i+1} := \varphi(s_{i+1})$ ;
     $\beta_i := \begin{cases} (r_{k+1}^H r_{k+1}) / (r_k^H r_k); \\ \max\{0, (r_{k+1}^H (r_{k+1} - r_k)) / (r_k^H r_k)\} \end{cases}$ ;
     $p_{i+1} := r_{i+1} + \beta_i p_i$ ;
     $i := i + 1$ 
  end;
  nlCG :=  $s_{i-1}$ 
end nlCG;

```

The case study of the nonlinear CG method illustrates how a linear iterative method can be adjusted to solve nonlinear systems of equations.

Although CG is a Krylov space method (see Chapter 2), it is very special among these algorithms, since it is the only method of this class that can be interpreted in two ways in every step: (1) solving a linear system of equations, or (2) optimizing a scalar functional. This is why the approach for adjusting the CG method to solve nonlinear systems of equations cannot be generalized to the whole family of Krylov space methods.

In this sense our approach is different and new. We use the fact, that vector extrapolation methods (see Chapter 3) are algebraically equivalent to Krylov space methods. Vector extrapolation methods have already been proven to solve nonlinear systems of equations. However, their numerical performance is bad, while the numerical performance of Krylov space methods is very good. Considering this we came up with the idea to adjust existing Krylov space methods by the help of their mathematical equivalent vector extrapolation compan-

OUR
APPROACH

ions such that they are capable of solving nonlinear systems of equations.

Vector extrapolation methods accelerate the convergence of a sequence of vectors. This sequence of vectors has to be provided as an input to the particular vector extrapolation method. In our approach the sequence of vectors is generated by an iterative process that solves the given problem, but is rather slow or may even diverge. Since the original theory of vector extrapolation methods is based on linear problems, the iterative process generating the vector sequence should have a local linear convergence rate. Basically classical iterative methods like Jacobi, Gauss-Seidel or SOR were used in the linear case to serve as the generating process for linear vector extrapolation methods. Among others their nonlinear counterparts meet the requirements to generate the vector sequences that can be accelerated by nonlinear vector extrapolation methods.

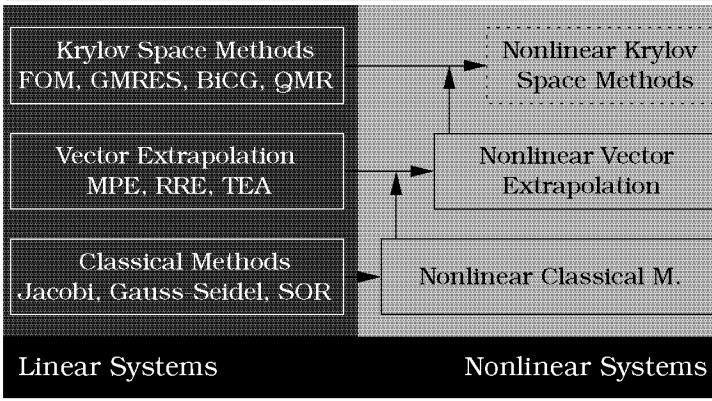
Our nonlinear class of Krylov space methods uses ideas from vector extrapolation to manage the transition from linear to nonlinear. The new methods still require an iterative process generating a vector sequence. However, in our point of view this process acts as a nonlinear preconditioner to the new methods. Figure 1.1 shows the theoretical layers of the approach discussed in this thesis.

1.1 Overview

CHAPTER 1

This thesis is structured as follows. In this chapter we first give an overview of our notation, definitions and some important lemmas and properties which we will use. Next we give an overview of methods solving nonlinear systems of equations. These are mainly variations of Newton's method, since we will see that our approach can be compared with these types of methods. The other class of nonlinear solvers described is

Figure 1.1 Our approach of developing nonlinear Krylov space methods involves nonlinear vector extrapolation methods, which rely on nonlinear classical methods. All of these method are linear methods originally.



the class of nonlinear classical methods since we need them as preconditioners. At the end of the chapter we describe a sample problem with which we will illustrate all multivariate nonlinear problems and present numerical results for the methods described.

In Chapter 2 we describe the theory of linear Krylov space methods. This large class of iterative linear solvers can be classified with respect to several parameters, of which we choose the three we consider the most important:

CHAPTER 2

Matrix Type Different Krylov space methods exist for *symmetric matrices*, *symmetric positive definite matrices* and *general matrices*.

Projection Method Krylov space methods use two different approaches for projecting the residual vector into the Krylov space: Either the residual vector is determined such that it is *orthogonal* to the considered Krylov space (OR approach) or the residual

vector is determined such that it is *minimized* over the Krylov space in consideration (MR approach).

Krylov Space Generation The generation of a Krylov space according to its definition is numerically unstable. Therefore special methods have been developed to generate a Krylov space in a numerically stable way. We distinguish between the *Arnoldi process* based on orthogonalization, and the *Lanczos process* based on biorthogonality.

Table 1.1 Classification of linear Krylov space methods with respect to three parameters: Matrix Type (rows), Projection Method (columns), Krylov Space Generation (colors): Grey is for symmetric Lanczos/Arnoldi methods, black is for general Arnoldi methods, white is for general Lanczos methods.

	Symmetric Matrix		General Matrix
	pos. def.	indefinite	
Minimization	CR	MINRES	GMRES
			QMR
Orthogonalization	CG	SYMMLQ	FOM
			BICG

In Table 1.1 an overview of linear Krylov space methods is shown with respect to these three parameters. For every category this figure shows only one representative, although there are much more methods around. Since nonlinear operators cannot be categorized with respect to the matrix type we have to choose the most general matrix type for our candidates to be nonlinearized. This is why we describe the four methods given in the last column of Table 1.1, of which GMRES and

FOM are Arnoldi-methods, BiCG and QMR are Lanczos-methods. The theory of Krylov space methods is developed with respect to the two remaining parameters, projection method and Krylov space generation, of which the first one is not restricted to Krylov space methods but belongs to the larger class of projection space methods.

In Chapter 3 the theory of vector extrapolation methods is reviewed with respect to both for the linear and the nonlinear case. Their connection with Steffensen's method in the scalar case and Henrici's extension of it in the multivariate case is revealed. The vector extrapolation methods MPE, RRE and TEA are described and their equivalence to the Krylov space methods FOM, GMRES and BiCG is proved.

CHAPTER 3

In Chapter 4 the approaches of Krylov space methods and vector extrapolation methods are combined to nonlinear Krylov space methods. For the Arnoldi process a method is given to perform the transition to the nonlinear case. These methods are illustrated by two new methods nlFOM, nlGMRES.

CHAPTER 4

In Chapter 5 we describe the application of an ABB DEGLOR oven. By this real world application we illustrate the performance of the new algorithms and we compare them with the existing ones that we described in Chapter 1. Numerical results are given and interpreted. Based on these results we conclude on our methods.

CHAPTER 5

1.2 Notation and Preliminaries

1.2.1 Terminology

We refer to any mapping that returns a scalar value as *functional*. Any mapping that returns a non scalar value (i.e. a vector or a matrix) is referred to as *operator*.

1.2.2 Typesetting

Throughout the thesis we use the typesetting conventions for mathematical variables shown in Table 1.2.

Table 1.2 Typesetting conventions for variables.

Mathematical Meaning	Typeset	Examples
Integer scalar values	small non bold roman letters	$i = 1$
Real scalar values	small non bold greek letters	$\alpha = 0.3$
Vectors	small bold letters (roman and greek)	$\mathbf{x}, \boldsymbol{\eta}_i$
Matrices	large bold letters (roman and greek)	$\mathbf{A}, \boldsymbol{\Psi}$
Sets	large calligraphic letters	\mathcal{M}

Operators and functionals reflect the type of their return value in their typesetting, i.e.

- the operator $\mathbf{D}\varphi$ returns a matrix,
- the operator φ returns a vector.

An exception are functionals that return a scalar value: Since we do not deal with functionals that return integer values, a scalar functional is always type set non bold. However, it can be represented by a roman or a greek letter.

We denote the transposition of a matrix by an H in the upper right corner of an object. We use a T instead of an H, if a transposition is used for “cosmetic” purposes only (e.g. to save space on paper).

1.2.3 Matrix Properties

Definition 1.3 A square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is called hermitian if HERMITIAN

$$\mathbf{A}^H = \mathbf{A}. \quad (1.8)$$

A real square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with property (1.8) is called symmetric. SYMMETRIC

Definition 1.4 We call a square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ skew hermitian if SKEW
HERMITIAN

$$\mathbf{A}^H = -\mathbf{A}. \quad (1.9)$$

A real square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with property (1.9) is called skew symmetric. SKEW
SYMMETRIC

Definition 1.5 A square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is called positive definite if the real part of the quadratic form is positive with respect to any nonzero vector $\mathbf{a} \in \mathbb{C}^n, \mathbf{a} \neq \mathbf{0}$, i.e. POSITIVE
DEFINITE

$$\operatorname{Re}(\mathbf{a}^H \mathbf{A} \mathbf{a}) > 0. \quad (1.10)$$

Note that condition (1.10) is equivalent to requesting that the eigenvalues (Def. 1.7) of the hermitian part (Def. 1.10) of \mathbf{A} are positive.

Definition 1.6 The range of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined by RANGE AND
NULL SPACE
OF A MATRIX

$$\mathcal{R}(\mathbf{A}) := \{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^n : \mathbf{y} = \mathbf{A}\mathbf{x}\}.$$

The null space of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined by

$$\mathcal{N}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{0}\}.$$

1.2.4 Matrix Decompositions

EIGENVALUE
DECOMPOSITION

Definition 1.7 If a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ can be decomposed into a diagonal matrix $\mathbf{\Lambda} \in \mathbb{C}^{n \times n}$ and a nonsingular matrix $\mathbf{V} \in \mathbb{C}^{n \times n}$ such that

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1},$$

\mathbf{A} is called diagonalizable. $\mathbf{\Lambda}$ is the diagonal matrix containing the eigenvalues of \mathbf{A} , \mathbf{V} is the matrix of eigenvectors of \mathbf{A} . This decomposition is called eigenvalue decomposition.

A hermitian matrix has real eigenvalues and orthonormal eigenvectors.

SCHUR DE-
COMPOSITION

Theorem 1.8 (cf. Theorem 7.1.3, [15]) Any square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ can be decomposed according to a unitary matrix $\mathbf{Q} \in \mathbb{C}^{n \times n}$ and an upper triangular matrix $\mathbf{U} \in \mathbb{C}^{n \times n}$ whose diagonal entries are the eigenvalues of \mathbf{A} . The decomposition of \mathbf{A} defined by

$$\mathbf{A} = \mathbf{Q}\mathbf{U}\mathbf{Q}^H$$

is called Schur Decomposition of \mathbf{A} . □

SINGULAR
VALUE DE-
COMPOSITION

Theorem 1.9 (cf. Theorem 2.5.2, [15]) Any matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$ can be decomposed according to

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = [\mathbf{U}_1 \mathbf{U}_2] \begin{bmatrix} \hat{\mathbf{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{bmatrix} \quad (1.11)$$

$$= \mathbf{U}_1 \hat{\mathbf{\Sigma}} \mathbf{V}_1^H. \quad (1.12)$$

$\hat{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ is a diagonal matrix with positive diagonal elements where r denotes the rank of \mathbf{A} , $\mathbf{U} \in \mathbb{C}^{n \times n}$ and $\mathbf{V} \in \mathbb{C}^{m \times m}$ are unitary matrices. They are partitioned into $\mathbf{U}_1 \in \mathbb{C}^{n \times r}$, $\mathbf{U}_2 \in \mathbb{C}^{n \times (n-r)}$, $\mathbf{V}_1 \in \mathbb{C}^{m \times r}$, and $\mathbf{V}_2 \in \mathbb{C}^{m \times (m-r)}$.

\mathbf{U}_1 is a orthonormal basis of $\mathcal{R}(\mathbf{A})$, \mathbf{U}_2 is an orthonormal basis of $\mathcal{R}(\mathbf{A})^\perp$, \mathbf{V}_1 is an orthonormal basis of $\mathcal{N}(\mathbf{A})^\perp$ and \mathbf{V}_2 is an orthonormal basis of $\mathcal{N}(\mathbf{A})$.

Decomposition (1.11) is called singular value decomposition (SVD) of \mathbf{A} , with $\hat{\Sigma}$ containing its singular values, \mathbf{U} containing its left singular vectors and \mathbf{V} containing its right singular vectors. The singular value decomposition of a matrix is unique up to the order and uniqueness of its singular values. Therefore the diagonal elements in $\hat{\Sigma}$ can be ordered from the largest one in the upper left to the smallest one in lower right corner of the matrix.

Decomposition (1.12) is called the economical version of the SVD of \mathbf{A} . \square

Definition 1.10 Any square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ can be decomposed into its hermitian part (Definition 1.3) $\mathbf{S} \in \mathbb{C}^{n \times n}$ and skew hermitian part (Definition 1.4) $\mathbf{T} \in \mathbb{C}^{n \times n}$ so that

HERMITIAN
PART AND
SKEW
HERMITIAN
PART

$$\mathbf{A} = \mathbf{S} + \mathbf{T}.$$

\mathbf{S} and \mathbf{T} are determined by

$$\begin{aligned} \mathbf{S} &:= \frac{1}{2} (\mathbf{A}^H + \mathbf{A}), \\ \mathbf{T} &:= \frac{1}{2} (\mathbf{A}^H - \mathbf{A}). \end{aligned}$$

1.2.5 Norms

We denote norms of vectors and matrices by the common notation $\|\cdot\|$. If nothing else is specified, any norm can be used.

OPERATOR
NORM

Definition 1.11 Let $\mathbf{A} \in \mathbb{C}^{m \times n}$ be a linear operator. We call

$$\|\mathbf{A}\| := \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$$

an operator norm or induced vector norm.

\mathbf{A} -NORM OF A
VECTOR

Definition 1.12 (cf. [15], p. 530) Given a vector $\mathbf{x} \in \mathbb{C}^n$ and a hermitian positive definite matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$. The \mathbf{A} -norm of a vector \mathbf{x} is defined by

$$\|\mathbf{x}\|_{\mathbf{A}} := \sqrt{\mathbf{x}^H \mathbf{A} \mathbf{x}}. \quad (1.13)$$

Please recall that the corresponding matrix \mathbf{A} must be positive definite to ensure that $\mathbf{x}^H \mathbf{A} \mathbf{x}$ is positive for any non zero vector \mathbf{x} — a necessary condition for (1.13) to be a norm. The following lemma taken from Anne Greenbaum [16] generalizes the definition above for any induced vector norm and extends it also to matrix norms if the matrix used is hermitian positive definite.

Lemma 1.13 (Theorem 1.3.1, [16]) If $\|\cdot\|$ is a matrix norm on $\mathbb{R}^{n \times n}$, and if $\mathbf{M} \in \mathbb{R}^{n \times n}$ is nonsingular, then

$$\|\mathbf{A}\|_{\mathbf{M}^H \mathbf{M}} := \|\mathbf{M} \mathbf{A} \mathbf{M}^{-1}\|$$

is a matrix norm. If $\|\cdot\|$ is induced by a vector norm then $\|\mathbf{A}\|_{\mathbf{M}^H \mathbf{M}}$ is induced by the vector norm

$$\|\mathbf{x}\|_{\mathbf{M}^H \mathbf{M}} := \|\mathbf{M}\mathbf{x}\|.$$

Proof. The matrix norm property is easily proved by verifying the axioms of matrix norms. We therefore concentrate on the induced vector norm. If $\|\mathbf{A}\| = \max_{\mathbf{x} \neq 0} \|\mathbf{A}\mathbf{x}\|/\|\mathbf{x}\|$, then

$$\begin{aligned}
\|\mathbf{A}\|_{\mathbf{M}^{\mathbf{H}}\mathbf{M}} &= \max_{\mathbf{x} \neq \mathbf{0}} \|\mathbf{M}\mathbf{A}\mathbf{M}^{-1}\mathbf{x}\|/\|\mathbf{x}\| \\
&= \max_{\mathbf{a} \neq \mathbf{0}} \|\mathbf{M}\mathbf{A}\mathbf{a}\|/\|\mathbf{M}\mathbf{a}\| \\
&= \max_{\mathbf{a} \neq \mathbf{0}} \|\mathbf{A}\mathbf{a}\|_{\mathbf{M}^{\mathbf{H}}\mathbf{M}}/\|\mathbf{a}\|_{\mathbf{M}^{\mathbf{H}}\mathbf{M}}.
\end{aligned}$$

□

Definition 1.14 Given an arbitrary matrix norm $\|\cdot\|$ and a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. The condition number $\kappa(\mathbf{A})$ is defined by

MATRIX
CONDITION
NUMBER

$$\kappa(\mathbf{A}) := \|\mathbf{A}^{-1}\| \|\mathbf{A}\|.$$

In [26, p. 168] we find the following Lemma that helps us estimate the norm of the inverse of an operator.

ESTIMATING
NORMS

Lemma 1.15 Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ and let $\|\cdot\|$ denote any operator norm. If $\|\mathbf{M}\| < 1$ then $\mathbf{1} + \mathbf{M}$ is nonsingular and

BANACH
LEMMA

$$\frac{1}{1 + \|\mathbf{M}\|} \leq \|(\mathbf{1} + \mathbf{M})^{-1}\| \leq \frac{1}{1 - \|\mathbf{M}\|}. \quad (1.14)$$

Proof. A matrix $\mathbf{1} + \mathbf{M}$ is nonsingular, if the linear system $(\mathbf{1} + \mathbf{M})\mathbf{x} = \mathbf{0}$ has only the trivial solution $\mathbf{x} = \mathbf{0}$. Suppose that $(\mathbf{1} + \mathbf{M})\mathbf{x} = \mathbf{0}$. Then $\mathbf{x} = -\mathbf{M}\mathbf{x}$ and $\|\mathbf{x}\| = \|\mathbf{M}\mathbf{x}\| \leq \|\mathbf{M}\|\|\mathbf{x}\|$. Since $\|\mathbf{M}\| < 1$, the only solution to this inequality is $\mathbf{x} = \mathbf{0}$. Therefore $\mathbf{A} := (\mathbf{1} + \mathbf{M})^{-1}$ exists.

From

$$1 = \|\mathbf{1}\| = \|\mathbf{A}(\mathbf{1} + \mathbf{M})\| \leq \|\mathbf{A}\| \|\mathbf{1} + \mathbf{M}\| \leq \|\mathbf{A}\|(1 + \|\mathbf{M}\|),$$

we conclude the left hand side of (1.14). Similarly

$$\|\mathbf{A}\| = \|\mathbf{1} - \mathbf{A}\mathbf{M}\| \leq 1 + \|\mathbf{A}\| \|\mathbf{M}\|$$

yields the right hand side of (1.14).

□

An important consequence of this lemma for estimating another matrix inverse is the following one.

PERTURBA-
TION
LEMMA

Lemma 1.16 (2.3.2, [28]) *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ and let $\|\cdot\|$ denote any operator norm. If $\|\mathbf{A}^{-1}\| \leq p$, $\|\mathbf{A} - \mathbf{B}\| \leq q$ and $pq < 1$, then \mathbf{B}^{-1} exists and*

$$\|\mathbf{B}^{-1}\| \leq \frac{p}{1 - pq}.$$

Proof. Define $\mathbf{M} := \mathbf{A}^{-1}(\mathbf{B} - \mathbf{A})$. We have $\|\mathbf{M}\| \leq \|\mathbf{A}^{-1}\| \|\mathbf{A} - \mathbf{B}\| \leq pq < 1$. By Lemma 1.15 $(\mathbf{1} + \mathbf{M})^{-1}$ exists and

$$\frac{1}{1 + pq} \leq \|(\mathbf{1} + \mathbf{M})^{-1}\| \leq \frac{1}{1 - pq}.$$

Since $\mathbf{A}(\mathbf{1} + \mathbf{M}) = \mathbf{B}$, also $(\mathbf{1} + \mathbf{M})^{-1}\mathbf{A}^{-1} = \mathbf{B}^{-1}$ exists.

Hence we have

$$\|\mathbf{B}^{-1}\| \leq \|(\mathbf{1} + \mathbf{M})^{-1}\| \|\mathbf{A}^{-1}\| \leq \frac{p}{1 - pq},$$

as claimed. □

1.2.6 Inverses

Definition 1.9 gives rise to the definition of a generalisation of the common matrix inverse.

MOORE-
PENROSE
PSEUDO
INVERSE

Definition 1.17 *Given the singular value decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ as*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \quad \text{with} \quad \mathbf{\Sigma} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n \times m}.$$

The pseudo inverse \mathbf{A}^+ of \mathbf{A} is defined by

$$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^H \quad \text{with} \quad \mathbf{\Sigma}^+ := \begin{bmatrix} \hat{\boldsymbol{\Sigma}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

1.2.7 Sets

Definition 1.18 Let $\mathbf{s}_0 \in \mathbb{R}^n$ be a given point and $\rho \in \mathbb{R}$ be a given scalar value.

We call the set

$$\mathbb{B}(\mathbf{s}_0, \rho) := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{s}_0\| < \rho\}$$

the open sphere around \mathbf{s}_0 with radius ρ .

Definition 1.19 Let $\mathbf{s}_0 \in \mathbb{R}^n$ be a given point and $\rho \in \mathbb{R}$ be a given scalar value.

We denote the closed sphere around \mathbf{s}_0 with radius ρ by

$$\overline{\mathbb{B}}(\mathbf{s}_0, \rho) := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{s}_0\| \leq \rho\}$$

1.2.8 Derivatives

We use the following definitions from Ortega and Rheinboldt [28] motivated by the mean value theorem 1.23 to define two kinds of derivatives for nonlinear operators.

Definition 1.20 (Def. 3.1.1, [28]) An operator $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ is Gateaux- (or G-) differentiable at an interior point of \mathcal{M} if there exists a linear operator $\mathbf{A} \in \mathbb{R}^{n \times m}$ such that for any $\mathbf{h} \in \mathbb{R}^n$,

$$\lim_{t \rightarrow 0} \frac{\|\varphi(\mathbf{x} + t\mathbf{h}) - \varphi(\mathbf{x}) - t\mathbf{A}\mathbf{h}\|}{t} = 0.$$

The linear operator \mathbf{A} is denoted by $\mathbf{D}\varphi(\mathbf{x})$ and is called the G-derivative of φ at \mathbf{x} .

Definition 1.21 (Def. 3.1.5, [28]) An operator $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ is Fréchet- (or F-) differentiable at $\mathbf{x} \in \text{int}(\mathcal{M})$ if there is an $\mathbf{A} \in \mathbb{R}^{n \times m}$ such that

$$\lim_{\mathbf{h} \rightarrow 0} \frac{\|\varphi(\mathbf{x} + \mathbf{h}) - \varphi(\mathbf{x}) - \mathbf{A}\mathbf{h}\|}{\|\mathbf{h}\|} = 0.$$

Again the linear operator \mathbf{A} is denoted by $\mathbf{D}\varphi(\mathbf{x})$ and is called the F-derivative of φ at \mathbf{x} .

Note that the F-derivative is a special case of a G-derivative. That is: Whenever an operator has an F-derivative at a point \mathbf{x} then it is also G-differentiable at that point.

The following definition of partial derivatives is also given in Ortega and Rheinboldt.

PARTIAL
DERIVATIVES

Definition 1.22 (Definition 5.2.2, [28]) Let \mathbb{R}^n denote the product space $\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_p}$, where $n_1 + \cdots + n_p = n$. We denote the elements of \mathbb{R}^n by $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$ with $\mathbf{x}_i \in \mathbb{R}^{n_i}$ for $i = 1, \dots, p$.

Let $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ and, for a given $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p) \in \mathcal{M}$, set

$$\mathcal{M}_i(\mathbf{X}) = \{\mathbf{y} \in \mathbb{R}^{n_i} \mid (\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{y}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_p) \in \mathcal{M}\}.$$

We define $\varphi_i(\mathbf{X}) : \mathcal{M}_i(\mathbf{X}) \rightarrow \mathbb{R}^m$ by

$$\varphi_i(\mathbf{X}, \mathbf{y}) := \varphi(\mathbf{x}_1, \dots, \mathbf{y}, \dots, \mathbf{x}_p), \quad \mathbf{y} \in \mathcal{M}_i(\mathbf{X}).$$

Then φ has a partial F-derivative $\partial_i \varphi(\mathbf{X}) := \mathbf{D}\varphi_i(\mathbf{X}, \mathbf{x}_i)$ at \mathbf{X} with respect to \mathbb{R}^{n_i} if \mathbf{x}_i is an element of the interior of \mathcal{M}_i and φ_i has an F-derivative at \mathbf{x}_i .

In addition, $\partial_i \varphi$ is strong at \mathbf{X} if, given $\epsilon > 0$, there is a $\delta > 0$ so that

$$\begin{aligned} \|\varphi(\mathbf{Y}, \mathbf{y}_i + \mathbf{h}_i) - \varphi(\mathbf{Y}, \mathbf{y}_i + \mathbf{k}_i) \\ - \partial_i \varphi(\mathbf{X})(\mathbf{h}_i - \mathbf{k}_i)\| \leq \epsilon \|\mathbf{h}_i - \mathbf{k}_i\| \end{aligned}$$

whenever $\|\mathbf{X} - \mathbf{Y}\| \leq \delta$, $\|\mathbf{h}_i\| \leq \delta$, and $\|\mathbf{k}_i\| \leq \delta$.

Both kinds of derivatives of φ — Fréchet and Gateaux — can be expressed by the *Jacobian* $J_\varphi(\mathbf{x})$ of the operator,

JACOBIAN

$$J_\varphi(\mathbf{x}) := \begin{bmatrix} \partial_1 \varphi_1(\mathbf{x}) & \cdots & \partial_n \varphi_1(\mathbf{x}) \\ \vdots & & \vdots \\ \partial_1 \varphi_m(\mathbf{x}) & \cdots & \partial_n \varphi_m(\mathbf{x}) \end{bmatrix},$$

if they exist. However, Ortega and Rheinboldt note that the existence of the Jacobian neither implies the G-differentiability nor the F-differentiability of the operator φ .

1.2.9 Mean Value Theorem

We cite a very important mean value theorem for non-linear operators from Ortega and Rheinboldt without proof. The proof can be found in their book [28, pp. 68 – 73].

Theorem 1.23 (3.2.12, [28]) *Let $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ be continuously (G- or F-)differentiable on a convex set $\mathcal{M}_\delta \subset \mathcal{M}$, and suppose that, for constants $\alpha \geq 0$ and $p \geq 0$, $\mathbf{D}\varphi$ satisfies*

$$\|\mathbf{D}\varphi(\mathbf{x}) - \mathbf{D}\varphi(\mathbf{y})\| \leq \alpha \|\mathbf{x} - \mathbf{y}\|^p, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{M}_\delta.$$

Then, for any $\mathbf{x}, \mathbf{y} \in \mathcal{M}_\delta$,

$$\|\varphi(\mathbf{y}) - \varphi(\mathbf{x}) - \mathbf{D}\varphi(\mathbf{x})(\mathbf{y} - \mathbf{x})\| \leq \frac{\alpha}{p+1} \|\mathbf{y} - \mathbf{x}\|^{p+1}.$$

□

1.2.10 Iterative Methods

ITERATIVE
METHOD

Definition 1.24 An iterative method is a computational rule for solving a system of (nonlinear) equations

$$\begin{aligned}\varphi : \mathbb{R}^n &\rightarrow \mathbb{R}^n, \mathbf{x} \mapsto \varphi(\mathbf{x}) \\ \varphi(\mathbf{s}) &= \mathbf{0}\end{aligned}\tag{1.15}$$

for \mathbf{s} starting with an initial approximation for the solution \mathbf{s}_0 and generating further approximations \mathbf{s}_i for $i = 1, 2, \dots$ by applying an iteration rule $\{\mathbf{g}_i\}_{i=0}^\infty$

$$\begin{aligned}\mathbf{g}_i : \mathbb{R}^{n \times (i+1)} &\rightarrow \mathbb{R}^n, (\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_i) \mapsto \mathbf{g}(\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_i) \\ \mathbf{s}_{i+1} &:= \mathbf{g}_i(\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_i),\end{aligned}\tag{1.16}$$

until convergence.

LINEAR
ITERATIVE
METHOD

Definition 1.25 A linear iterative method is an iterative method in which the functional φ , the zero of which has to be found, is defined by

$$\begin{aligned}\mathbf{A} &\in \mathbb{R}^{n \times n}, \mathbf{b} \in \mathbb{R}^n \\ \varphi : \mathbb{R}^n &\rightarrow \mathbb{R}^n, \mathbf{x} \mapsto \mathbf{b} - \mathbf{A}\mathbf{x}.\end{aligned}$$

STATIONARY
ITERATIVE
METHOD

Definition 1.26 If the operator of the iteration rule defined by (1.16) is linear, i.e. for some constant matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ and some constant vector $\mathbf{d} \in \mathbb{R}^n$

$$\mathbf{g}(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{d},$$

the iterative method is called stationary.

RESIDUAL

Definition 1.27 The operator

$$\mathbf{r} : (\mathbb{R}^n \rightarrow \mathbb{R}^n, \mathbb{R}^n) \rightarrow \mathbb{R}^n, (\varphi, \mathbf{s}_i) \mapsto \varphi(\mathbf{s}_i)$$

that maps the current approximation \mathbf{s}_i to the operator to be solved $\varphi(\cdot)$ is called residual.

For linear iterative methods the residual is of the form

$$\mathbf{r} : (\mathbb{R}^{n \times n}, \mathbb{R}^n, \mathbb{R}^n) \rightarrow \mathbb{R}^n, (\mathbf{A}, \mathbf{b}, \mathbf{s}_i) \mapsto \mathbf{b} - \mathbf{A}\mathbf{s}_i.$$

In sections where \mathbf{A} , \mathbf{b} and \mathbf{s}_i are obvious from the context, an abbreviation for the residual is used

$$\mathbf{r}_i := \mathbf{r}(\mathbf{A}, \mathbf{b}, \mathbf{s}_i),$$

\mathbf{r}_i is called the residual vector.

Definition 1.28 Given the exact solution \mathbf{s} of φ .

ERROR

For all iterative methods we define the error operator of the current approximation \mathbf{s}_i to be

$$\boldsymbol{\eta} : \mathbb{R}^n \rightarrow \mathbb{R}^n, \mathbf{s}_i \mapsto \mathbf{s} - \mathbf{s}_i.$$

In sections where \mathbf{s} and the current approximation \mathbf{s}_i are obvious from the context, an abbreviation for the error operator is used

$$\boldsymbol{\eta}_i := \boldsymbol{\eta}(\mathbf{s}_i),$$

$\boldsymbol{\eta}_i$ is called the error vector.

Lemma 1.29 Given an iterative method that converges to a zero \mathbf{s} of an F -differentiable operator φ , satisfying the conditions of Lemma 1.23.

In step i the error operator $\boldsymbol{\eta}$ and the residual operator \mathbf{r} are connected by

$$\mathbf{r}(\varphi, \mathbf{s}_i) = -\mathbf{D}\varphi(\mathbf{s})\boldsymbol{\eta}(\mathbf{s}_i) + \mathcal{O}(\|\boldsymbol{\eta}(\mathbf{s}_i)\|^2).$$

Proof. By Definition 1.28 of the error operator we have $\mathbf{s}_i = \mathbf{s} - \boldsymbol{\eta}(\mathbf{s}_i)$. Using the Taylor approximation of $\mathbf{r}(\boldsymbol{\varphi}, \mathbf{s} - \boldsymbol{\eta}(\mathbf{s}_i))$ around \mathbf{s} we get

$$\begin{aligned} \mathbf{r}(\boldsymbol{\varphi}, \mathbf{s}_i) &= \underbrace{\mathbf{r}(\boldsymbol{\varphi}, \mathbf{s})}_0 - \mathbf{D}\mathbf{r}(\boldsymbol{\varphi}, \mathbf{s})\boldsymbol{\eta}(\mathbf{s}_i) + O(\|\boldsymbol{\eta}(\mathbf{s}_i)\|^2) \\ &= -\mathbf{D}\boldsymbol{\varphi}(\mathbf{s})\boldsymbol{\eta}(\mathbf{s}_i) + O(\|\boldsymbol{\eta}(\mathbf{s}_i)\|^2) \end{aligned}$$

as claimed. \square

Note that Lemma 1.29 reads for linear systems

$$\mathbf{r}(\mathbf{A}, \mathbf{b}, \mathbf{s}_i) = \mathbf{A}\boldsymbol{\eta}(\mathbf{s}_i).$$

1.2.11 Limit and Anti-Limit

Definition 1.30 Given an infinite sequence of vectors (or values) $\{\mathbf{x}_i\}$, $i = 0, 1, 2, \dots$, generated by a (possibly unknown) operator

$$\begin{aligned} \mathbf{g} : \mathbb{R}^n &\rightarrow \mathbb{R}^n, \mathbf{x} \rightarrow \mathbf{g}(\mathbf{x}) \\ \mathbf{x}_{i+1} &= \mathbf{g}(\mathbf{x}_i) \quad \text{for } i \geq 0. \end{aligned}$$

We call

$$\mathbf{s} = \mathbf{g}(\mathbf{s})$$

a limit of the given sequence if for every $\varepsilon > 0$ there exists an $i \geq 0$ such that $\|\mathbf{x}_j - \mathbf{s}\| < \varepsilon$ for all $j \geq i$.

If \mathbf{s} is not a limit, it is called anti-limit of the sequence $\langle \mathbf{x}_i \rangle$.

1.2.12 Types of Convergence

Iterative methods can be classified by the *rate of convergence*. We distinguish between two classes. The first

class, the *quotient convergence class* is motivated by the fact that estimates of the form

$$\|\boldsymbol{\eta}(\mathbf{x}_{k+1})\| \leq \gamma \|\boldsymbol{\eta}(\mathbf{x}_k)\|^p, \forall k \geq k_0 \quad (1.17)$$

often arise naturally in the study of iterative processes. We recite a definition by Kelley [25].

Definition 1.31 (Definition 4.1.1, [25]) *Let $\{\mathbf{x}_i\} \subset \mathbb{R}^n$ be a given sequence and $\mathbf{s} \in \mathbb{R}^n$. Then*

- *the sequence converges to \mathbf{s} \mathcal{Q} -linearly with \mathcal{Q} -factor $Q_1 \in (0, 1)$ if for k sufficiently large*

$$\|\boldsymbol{\eta}(\mathbf{x}_{k+1})\| \leq Q_1 \|\boldsymbol{\eta}(\mathbf{x}_k)\|.$$

- *the sequence converges to \mathbf{s} \mathcal{Q} -superlinearly if*

$$\lim_{i \rightarrow \infty} \frac{\|\boldsymbol{\eta}(\mathbf{x}_{i+1})\|}{\|\boldsymbol{\eta}(\mathbf{x}_i)\|} = 0.$$

- *the sequence converges to \mathbf{s} \mathcal{Q} -quadratically if the sequence converges to \mathbf{s} and there is a $Q_2 > 0$ such that for k sufficiently large*

$$\|\boldsymbol{\eta}(\mathbf{x}_{k+1})\| \leq Q_2 \|\boldsymbol{\eta}(\mathbf{x}_k)\|^2.$$

- *the sequence converges to \mathbf{s} with \mathcal{Q} -order $p > 1$ if the sequence converges to \mathbf{s} and there is a $Q_p > 0$ such that for k sufficiently large*

$$\|\boldsymbol{\eta}(\mathbf{x}_{k+1})\| \leq Q_p \|\boldsymbol{\eta}(\mathbf{x}_k)\|^p.$$

The optimal factors Q_1 , Q_2 and Q_p are called quotient convergence factors or \mathcal{Q} -factors, for short.

The second class, the *root convergence class*, is motivated in Ortega and Rheinboldt [28] by (1.17). For $p = 1$ and $k_0 = 0$ this equation reduces to

$$\|\boldsymbol{\eta}(\mathbf{x}_k)\| \leq \gamma \|\boldsymbol{\eta}(\mathbf{x}_{k-1})\| \leq \cdots \leq \gamma^k \|\boldsymbol{\eta}(\mathbf{x}_0)\|. \quad (1.18)$$

“If $\gamma < 1$ then (1.18) shows that the norms of the error vectors $\eta(x_k)$ are decreasing as rapidly as a geometric progression with ratio γ . Hence, in analogy with the *root test* for the convergence of series, we are led to consider geometric averages of the $\eta(x_k)$ ” [28, p. 288]. Thus the name *root convergence*.

Because it is much simpler to understand than the definitions given by Ortega and Rheinboldt (although they are more precise) we use Kelley’s [25] definition of R-convergence.

Definition 1.32 (Definition 4.1.3, [25]) *Let $\{x_i\} \subset \mathbb{R}^n$ be a given sequence and $s \in \mathbb{R}^n$. Then $\{x_i\}$ converges to s R-(quadratically, superlinearly, linearly) if there is a sequence $\{\xi_i\} \subset \mathbb{R}$ converging Q-(quadratically, superlinearly, linearly) to zero such that*

$$\|x_i - s\| \leq \xi_i.$$

We say that $\{x_i\}$ converges R-superlinearly with R-order $p > 1$ if $\{\xi_i\}$ converges to zero Q-superlinearly with Q-order p .

1.2.13 Spectral Radius

Definition 1.33 (Def. 1.3.3, [16]) *The spectral radius $\rho(\mathbf{A})$ of a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is*

$$\rho(\mathbf{A}) := \max\{|\lambda| : \lambda \text{ is an eigenvalue of } \mathbf{A}\}.$$

Lemma 1.34 (Theorem 1.3.2, [16]) *If $\|\cdot\|$ is any operator norm and $\mathbf{A} \in \mathbb{R}^{n \times n}$, then $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$.*

Proof. Let λ be an eigenvalue of \mathbf{A} with $|\lambda| = \rho(\mathbf{A})$, and let \mathbf{v} be a corresponding eigenvector. Let \mathbf{V} be the matrix in $\mathbb{R}^{n \times n}$ each of whose columns is \mathbf{v} . Then $\mathbf{A}\mathbf{V} = \lambda\mathbf{V}$, and

$$\rho(\mathbf{A})\|\mathbf{V}\| = |\lambda|\|\mathbf{V}\| = \|\lambda\mathbf{V}\| = \|\mathbf{A}\mathbf{V}\| \leq \|\mathbf{A}\|\|\mathbf{V}\|.$$

Since $\|\mathbf{V}\| \geq 0$, it follows that $\rho(\mathbf{A}) \leq \|\mathbf{V}\|$. \square

Lemma 1.35 (Theorem 1.3.3, [16]) *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be given. For any $\epsilon > 0$ there is a matrix norm $\|\cdot\|$ induced by a certain vector norm such that*

$$\rho(\mathbf{A}) \leq \|\mathbf{A}\| \leq \rho(\mathbf{A}) + \epsilon.$$

Proof. The left hand side of the inequality is established by Lemma 1.34. Lets concentrate on the right hand side. By the help of the Schur decomposition (Definition 1.8) we construct a norm to a given ϵ . Let $\mathbf{A} = \mathbf{Q}\mathbf{U}\mathbf{Q}^H$ be the Schur decomposition of \mathbf{A} and $\mathbf{D}_t := \text{Diag}(t, t^2, \dots, t^n)$. Then we have

$$\mathbf{D}_t \mathbf{U} \mathbf{D}_t^{-1} = \begin{bmatrix} \lambda_1 & t^{-1}u_{1,2} & t^{-2}u_{1,3} & \dots & t^{-n+1}u_{1,n} \\ & \lambda_2 & t^{-2}u_{2,3} & \dots & t^{-n+1}u_{2,n} \\ & & \ddots & & \vdots \\ & & & \lambda_{n-1} & t^{-n+1}u_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}.$$

For t sufficiently large the sum of all off diagonal elements is less than ϵ and $\|\mathbf{D}_t \mathbf{U} \mathbf{D}_t^{-1}\|_1 \leq \rho(\mathbf{A}) + \epsilon$. Therefore let us define a norm

$$\|\mathbf{A}\|_\epsilon := \|\mathbf{D}_t \mathbf{Q}^H \mathbf{A} \mathbf{Q} \mathbf{D}_t^{-1}\|_1$$

According to Lemma 1.13 this is a matrix norm induced by the vector norm $\|\cdot\|_{\mathbf{M}^H \mathbf{M}}$ with $\mathbf{M} = \mathbf{D}_t \mathbf{Q}^H$. \square

1.2.14 Contraction Mapping

The following theorem is well known and very important. It tells under what circumstances the fixed point of a contraction mapping is unique, and it gives an upper bound for the convergence rate. We cite theorem and proof from [25].

CONTRACTION
MAPPING
THEOREM

Theorem 1.36 (Theorem 4.2.1, [25]) *Let \mathcal{M} be a closed subset of \mathbb{R}^n and let \mathbf{g} be Lipschitz continuous on \mathcal{M} with Lipschitz constant $0 \leq \sigma < 1$ such that $\mathbf{g}(\mathbf{x}) \in \mathcal{M}$ for all $\mathbf{x} \in \mathcal{M}$. Then there is a unique fixed point of \mathbf{g} , $\mathbf{s} \in \mathcal{M}$, and the iteration defined by*

$$\mathbf{s}_{i+1} := \mathbf{g}(\mathbf{s}_i)$$

converges Q -linearly to \mathbf{s} with Q -factor σ for all initial iterates $\mathbf{s}_0 \in \mathcal{M}$.

Proof. Let $\mathbf{s}_0 \in \mathcal{M}$. Since \mathbf{g} is closed on \mathcal{M} the sequence $\{\mathbf{s}_i\}$ stays in \mathcal{M} . The sequence $\{\mathbf{s}_i\}$ remains bounded, since for all $j > 1$

$$\begin{aligned} \|\mathbf{s}_{j+1} - \mathbf{s}_j\| &= \|\mathbf{g}(\mathbf{s}_j) - \mathbf{g}(\mathbf{s}_{j-1})\| \leq \sigma \|\mathbf{s}_j - \mathbf{s}_{j-1}\| \leq \dots \\ &\leq \sigma^j \|\mathbf{s}_1 - \mathbf{s}_0\|. \end{aligned}$$

Therefore

$$\begin{aligned} \|\mathbf{s}_i - \mathbf{s}_0\| &= \left\| \sum_{k=0}^{i-1} \mathbf{s}_{k+1} - \mathbf{s}_k \right\| \\ &\leq \sum_{k=0}^{i-1} \|\mathbf{s}_{k+1} - \mathbf{s}_k\| \\ &\leq \|\mathbf{s}_1 - \mathbf{s}_0\| \sum_{k=0}^{i-1} \sigma^k \\ &\leq \|\mathbf{s}_1 - \mathbf{s}_0\| / (1 - \sigma). \end{aligned}$$

Now, for all $i, j \geq 0$,

$$\begin{aligned} \|\mathbf{s}_{i+j} - \mathbf{s}_i\| &= \|\mathbf{g}(\mathbf{s}_{i+j-1}) - \mathbf{g}(\mathbf{s}_{i-1})\| \\ &\leq \sigma \|\mathbf{s}_{i+j-1} - \mathbf{s}_{i-1}\| \leq \dots \\ &\leq \sigma^i \|\mathbf{s}_j - \mathbf{s}_0\| \\ &\leq \frac{\sigma^i}{1 - \sigma} \|\mathbf{s}_1 - \mathbf{s}_0\|. \end{aligned}$$

Hence for all j , $\lim_{i \rightarrow \infty} \|\mathbf{s}_{i+j} - \mathbf{s}_i\| = 0$ and therefore the sequence $\{\mathbf{s}_i\}$ is a Cauchy sequence and has a limit \mathbf{s} .

If \mathbf{g} had two fixed points \mathbf{s} and $\hat{\mathbf{s}}$ in \mathcal{M} , then the following inequality would hold,

$$\|\mathbf{s} - \hat{\mathbf{s}}\| = \|\mathbf{g}(\mathbf{s}) - \mathbf{g}(\hat{\mathbf{s}})\| \leq \sigma \|\mathbf{s} - \hat{\mathbf{s}}\|.$$

Since $\sigma < 1$ this can only be true if $\|\mathbf{s} - \hat{\mathbf{s}}\| = 0$, i.e. $\mathbf{s} = \hat{\mathbf{s}}$. Hence the fixed point is unique.

Finally we note that

$$\|\mathbf{s}_{i+1} - \mathbf{s}\| = \|\mathbf{g}(\mathbf{s}_i) - \mathbf{g}(\mathbf{s})\| \leq \sigma \|\mathbf{s}_i - \mathbf{s}\|,$$

which shows that the iteration has a \mathcal{Q} -linear convergence rate with \mathcal{Q} -factor σ to \mathbf{s} . \square

We like to remark that if \mathbf{g} is linear, i.e. $\mathbf{g}(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{d}$ with $\mathbf{G} \in \mathbb{R}^{n \times n}$ being a matrix and $\mathbf{d} \in \mathbb{R}^n$ being a vector, then by Lemma 1.35 σ equals the spectral radius $\rho(\mathbf{G})$.

Thus Theorem 1.36 reads for a linear iterative method:

Corollary 1.37 *Let $\mathbf{G} \in \mathbb{R}^{n \times n}$ and $\mathbf{d} \in \mathbb{R}^n$, let $\mathbf{g}(\mathbf{x}) := \mathbf{G}\mathbf{x} + \mathbf{d}$. If $\rho(\mathbf{G}) < 1$, then \mathbf{g} has a unique fixed point \mathbf{s} and the iteration defined by*

$$\mathbf{s}_{i+1} := \mathbf{g}(\mathbf{s}_i)$$

converges \mathcal{Q} -linearly to \mathbf{s} with \mathcal{Q} -factor $\rho(\mathbf{G})$ for all initial iterates $\mathbf{s}_0 \in \mathbb{R}^n$. \square

The following lemma specifies conditions such that a nonlinear operator \mathbf{f} with a dominant linear part has a unique solution $\mathbf{f}(\mathbf{x}) = \mathbf{y}$. We cite this lemma as a shortened version of Lemma 5.1.6 by Ortega and Rheinboldt [28].

Lemma 1.38 (cf. 5.1.6, [28]) *Suppose that $\mathbf{A} \in \mathbb{R}^{n \times n}$ is nonsingular and that $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ is Lipschitz continuous on a closed sphere $\mathcal{B}_0 = \overline{\mathcal{B}}(\mathbf{x}_0, \epsilon) \subset \mathcal{M}$ with constant σ where*

$$0 < \sigma < \beta^{-1}, \quad \beta = \|\mathbf{A}^{-1}\|. \quad (1.19)$$

Consider the operator $\mathbf{f} : \mathcal{B}_0 \rightarrow \mathbb{R}^n$ by $\mathbf{f}(\mathbf{x}) := \mathbf{A}\mathbf{x} - \varphi(\mathbf{x})$.

Then for any $\mathbf{y} \in \mathcal{B}_1 = \overline{\mathcal{B}}(\mathbf{f}(\mathbf{x}_0), \rho)$, where $\rho := (\beta^{-1} - \sigma)\epsilon$, the equation $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ has a unique solution in \mathcal{B}_0 . Hence, in particular, $\mathcal{B}_1 \subset \mathbf{f}(\mathcal{B}_0)$.

Proof. For fixed $\mathbf{y} \in \mathcal{B}_1$, define the operator $\mathbf{g} : \mathcal{B}_0 \rightarrow \mathbb{R}^n$ by $\mathbf{g}(\mathbf{x}) := \mathbf{A}^{-1}(\varphi(\mathbf{x}) + \mathbf{y}) = \mathbf{x} - \mathbf{A}^{-1}(\mathbf{f}(\mathbf{x}) - \mathbf{y})$. Clearly, $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ has a unique solution in \mathcal{B}_0 if and only if \mathbf{g} has a unique fixed point. But, for any $\mathbf{x}, \mathbf{z} \in \mathcal{B}_0$

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{z})\| = \|\mathbf{A}^{-1}(\varphi(\mathbf{x}) - \varphi(\mathbf{z}))\| \leq \beta\sigma\|\mathbf{x} - \mathbf{z}\|,$$

so that, by (1.19), \mathbf{g} is contractive on \mathcal{B}_0 . Moreover, for any $\mathbf{x} \in \mathcal{B}_0$,

$$\begin{aligned} \|\mathbf{g}(\mathbf{x}) - \mathbf{x}_0\| &\leq \|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}_0)\| + \|\mathbf{g}(\mathbf{x}_0) - \mathbf{x}_0\| \\ &\leq \beta\sigma\|\mathbf{x} - \mathbf{x}_0\| + \beta\|\mathbf{f}(\mathbf{x}_0) - \mathbf{y}\| \\ &\leq \beta\sigma\epsilon + \beta\rho = \epsilon \end{aligned}$$

by definition of ρ . Hence, \mathbf{g} maps \mathcal{B}_0 into \mathcal{B}_0 , and by the contraction mapping theorem 1.36, \mathbf{g} has a unique fixed point \mathbf{s} in \mathcal{B}_0 . \square

The implicit function theorem is well known and important. It shows that under some circumstances the solution of a multivariate operator equation $\varphi(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ is defined by an implicit continuous operator $\mathbf{x} = \mathbf{g}(\mathbf{y})$. It also gives conditions for the existence of the F-derivative of the implicit operator at the solution. Theorem 1.39 and its proof are cited from Ortega and Rheinboldt [28].

Theorem 1.39 (Theorem 5.2.4, [28]) *Suppose that $\varphi : \mathcal{M} \subset \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ is continuous on an open neighborhood $\mathcal{M}_0 \subset \mathcal{M}$ of a point $(\mathbf{x}_0, \mathbf{y}_0)$ for which $\varphi(\mathbf{x}_0, \mathbf{y}_0) = \mathbf{0}$. Assume that $\partial_1 \varphi$ exists and is strong at $(\mathbf{x}_0, \mathbf{y}_0)$, and that $\partial_1 \varphi(\mathbf{x}_0, \mathbf{y}_0)$ is nonsingular.*

Then there exist open neighborhoods $\mathcal{B}_1 \subset \mathbb{R}^n$ and $\mathcal{B}_2 \subset \mathbb{R}^p$ of \mathbf{x}_0 and \mathbf{y}_0 , respectively, such that, for any $\mathbf{y} \in \overline{\mathcal{B}}_2$, the equation $\varphi(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ has a unique solution $\mathbf{x} = \mathbf{g}(\mathbf{y}) \in \overline{\mathcal{B}}_1$ and the operator $\mathbf{g} : \mathcal{B}_2 \rightarrow \mathbb{R}^n$ is continuous.

Moreover, if $\partial_2 \varphi$ exists at $(\mathbf{x}_0, \mathbf{y}_0)$, then \mathbf{g} is F -differentiable at \mathbf{x}_0 and

$$D\mathbf{g}(\mathbf{y}_0) = -(\partial_1 \varphi(\mathbf{x}_0, \mathbf{y}_0))^{-1} \partial_2 \varphi(\mathbf{x}_0, \mathbf{y}_0). \quad (1.20)$$

Proof. In a first step we show that the solution $(\mathbf{x}_0, \mathbf{y}_0)$ of the equation $\varphi(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ is unique in $\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_2$. Let us therefore define $\mathbf{A} := \partial_1 \varphi(\mathbf{x}_0, \mathbf{y}_0)$, $\beta := \|\mathbf{A}^{-1}\|$, and let $0 < \sigma < \beta^{-1}$.

Since $\partial_1 \varphi$ is strong at $(\mathbf{x}_0, \mathbf{y}_0)$, we may choose $\delta_1, \delta_2 > 0$ so that

$$\|\varphi(\mathbf{x}, \mathbf{y}) - \varphi(\mathbf{z}, \mathbf{y}) - \mathbf{A}(\mathbf{x} - \mathbf{z})\| \leq \sigma \|\mathbf{x} - \mathbf{z}\| \quad (1.21)$$

for all $\mathbf{x}, \mathbf{z} \in \overline{\mathcal{B}}_1 := \overline{\mathcal{B}}(\mathbf{x}_0, \delta_1)$ and $\mathbf{y} \in \overline{\mathcal{B}}_2 := \overline{\mathcal{B}}(\mathbf{y}_0, \delta_2)$, and that $\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_2 \subset \mathcal{M}_0$.

For a fixed $\mathbf{y} \in \overline{\mathcal{B}}_2$, we define the operator $\mathbf{f}_{\mathbf{y}} : \overline{\mathcal{B}}_1 \rightarrow \mathbb{R}^n$ by

$$\mathbf{f}_{\mathbf{y}}(\mathbf{x}) := \mathbf{A}\mathbf{x} - \varphi(\mathbf{x}, \mathbf{y}) - \varphi(\mathbf{x}_0, \mathbf{y}), \quad \forall \mathbf{x} \in \overline{\mathcal{B}}_1.$$

Now (1.21) implies that

$$\|\mathbf{f}_{\mathbf{y}}(\mathbf{x}) - \mathbf{f}_{\mathbf{y}}(\mathbf{z})\| \leq \sigma \|\mathbf{x} - \mathbf{z}\|, \quad \forall \mathbf{x}, \mathbf{z} \in \overline{\mathcal{B}}_1.$$

Since φ is continuous at $(\mathbf{x}_0, \mathbf{y}_0)$ we may also assume that δ_2 has been chosen sufficiently small that

$$\|\varphi(\mathbf{x}_0, \mathbf{y})\| = \|\varphi(\mathbf{x}_0, \mathbf{y}) - \varphi(\mathbf{x}_0, \mathbf{y}_0)\| < \rho := (\beta^{-1} - \sigma)\delta_1.$$

Now Lemma 1.38 can be applied. Thus we know that the equation $\mathbf{A}\mathbf{x} - \mathbf{f}_{\mathbf{y}}(\mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y})$ has a unique solution in \mathcal{B}_1 . This implies that for any $\mathbf{y} \in \mathcal{B}_2$ the equation $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ has a unique solution in \mathcal{B}_1 .

Let us denote this solution by $\mathbf{g}(\mathbf{y})$. In a second step we show that the operator $\mathbf{g} : \mathcal{B}_2 \rightarrow \mathbb{R}^n$ is continuous. Let therefore $\mathbf{y}, \mathbf{z} \in \mathcal{B}_2$. By definition of \mathbf{g} , we have $\boldsymbol{\varphi}(\mathbf{g}(\mathbf{y}), \mathbf{y}) = \boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{z}) = \mathbf{0}$. With the help of (1.21) we construct

$$\begin{aligned} \|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{z})\| & \\ & \leq \|\mathbf{A}^{-1}(\boldsymbol{\varphi}(\mathbf{g}(\mathbf{y}), \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{y}) - \mathbf{A}(\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{z})))\| \\ & \quad + \|\mathbf{A}^{-1}(\boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{z}))\| \\ & \leq \beta\sigma\|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{z})\| + \beta\|\boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{z})\|. \end{aligned}$$

Since $\beta\sigma < 1$, by solving this inequality for $\|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{z})\|$ we obtain

$$\|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{z})\| \leq \frac{\beta}{1 - \beta\sigma} \|\boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{g}(\mathbf{z}), \mathbf{z})\|. \quad (1.22)$$

Thus the continuity of $\boldsymbol{\varphi}$ implies the continuity of \mathbf{g} .

In the final step we show the existence of the derivative of \mathbf{g} at \mathbf{y}_0 . Assume that $\partial_2\boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}_0)$ exists. Then by adding and subtracting $\partial_2\boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}_0)$ in the second step (1.22) becomes

$$\begin{aligned} \|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0)\| & \\ & \leq \frac{\beta}{1 - \beta\sigma} \|\boldsymbol{\varphi}(\mathbf{g}(\mathbf{y}_0), \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{g}(\mathbf{y}_0), \mathbf{y}_0)\| \\ & \leq \frac{\beta}{1 - \beta\sigma} (\|\boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}) - \boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}_0) - \partial_2\boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}_0)(\mathbf{y} - \mathbf{y}_0)\| \\ & \quad + \|\partial_2\boldsymbol{\varphi}(\mathbf{x}_0, \mathbf{y}_0)(\mathbf{y} - \mathbf{y}_0)\|) \end{aligned}$$

For a given $\epsilon > 0$ we find $\delta > 0$ so that for all $\mathbf{y} \in \mathcal{B}(\mathbf{y}_0, \delta)$ the last inequality becomes

$$\begin{aligned} \|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0)\| &\leq \lambda \|\mathbf{y} - \mathbf{y}_0\|, \\ \lambda &:= \beta / (1 - \beta\sigma)(\|\partial_2\varphi(\mathbf{x}_0, \mathbf{y}_0)\| + \epsilon). \end{aligned}$$

Set $\mathbf{A} := \partial_1\varphi(\mathbf{x}_0, \mathbf{y}_0)$ and $\mathbf{B} := \partial_2\varphi(\mathbf{x}_0, \mathbf{y}_0)$. We are able to construct now

$$\begin{aligned} &\|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0) + \mathbf{A}^{-1}\mathbf{B}(\mathbf{y} - \mathbf{y}_0)\| \\ &\leq \beta\|\mathbf{A}(\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0)) + \mathbf{B}(\mathbf{y} - \mathbf{y}_0)\| \\ &\leq \beta\|\varphi(\mathbf{g}(\mathbf{y}), \mathbf{y}) - \varphi(\mathbf{g}(\mathbf{y}_0), \mathbf{y}) - \mathbf{A}(\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0))\| \\ &\quad + \beta\|\varphi(\mathbf{g}(\mathbf{y}_0), \mathbf{y}) - \varphi(\mathbf{x}_0, \mathbf{y}_0) - \mathbf{B}(\mathbf{y} - \mathbf{y}_0)\| \\ &\leq \beta\sigma\|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y}_0)\| + \beta\epsilon\|\mathbf{y} - \mathbf{y}_0\| \leq \beta(\sigma\lambda + \epsilon)\|\mathbf{y} - \mathbf{y}_0\|. \end{aligned}$$

This result implies that the F-derivative of \mathbf{g} at \mathbf{y}_0 exists and that (1.20) holds. \square

1.2.15 Standard Assumptions

In order to determine the type and rate of convergence of an iterative method, Kelly [25] specifies three assumptions on a nonlinear operator φ that are often required to prove the convergence of a particular method.

Definition 1.40 (Assumption 4.3.1, [25]) *Given a nonlinear system as defined by (1.15). We refer to the following assumption on φ as the standard assumptions on φ .*

1. *The nonlinear system $\varphi(\mathbf{x}) = \mathbf{0}$ has a solution \mathbf{s} .*
2. *$\mathbf{D}\varphi : \mathcal{M} \rightarrow \mathbb{R}^{n \times n}$ is Lipschitz continuous with Lipschitz constant σ .*
3. *$\mathbf{D}\varphi(\mathbf{s})$ is nonsingular.*

The following lemma guarantees some upper and lower bounds on $\mathbf{D}\varphi$ and φ if the standard assumptions hold.

Lemma 1.41 (Lemma 4.3.1, [25]) *Let us suppose that the standard assumptions hold.*

Then there is $\delta > 0$ so that for all $\mathbf{x} \in \mathcal{B}(\mathbf{s}, \delta)$ and any operator norm $\|\cdot\|$

$$\|\mathbf{D}\varphi(\mathbf{x})\| \leq 2\|\mathbf{D}\varphi(\mathbf{s})\|, \quad (1.23)$$

$$\|\mathbf{D}\varphi(\mathbf{x})^{-1}\| \leq 2\|\mathbf{D}\varphi(\mathbf{s})^{-1}\|, \quad (1.24)$$

and

$$\frac{\|\boldsymbol{\eta}\|}{2\|\mathbf{D}\varphi(\mathbf{s})^{-1}\|} \leq \|\varphi(\mathbf{x})\| \leq 2\|\mathbf{D}\varphi(\mathbf{s})\|\|\boldsymbol{\eta}\|. \quad (1.25)$$

Proof. Let δ be small enough such that $\mathcal{B}(\mathbf{s}, \delta) \subset \mathcal{M}$. From the Lipschitz condition on $\mathbf{D}\varphi$ it follows that for all $\mathbf{x} \in \mathcal{B}(\mathbf{s}, \delta)$

$$\begin{aligned} \|\mathbf{D}\varphi(\mathbf{x})\| &\leq \|\mathbf{D}\varphi(\mathbf{x}) - \mathbf{D}\varphi(\mathbf{s})\| + \|\mathbf{D}\varphi(\mathbf{s})\| \\ &\leq \sigma\|\boldsymbol{\eta}\| + \|\mathbf{D}\varphi(\mathbf{s})\|. \end{aligned}$$

Hence (1.23) holds if $\sigma\delta < \|\mathbf{D}\varphi(\mathbf{s})\|$.

For the second equation let us set $\mathbf{A} := \mathbf{D}\varphi(\mathbf{x})$ and $\mathbf{B} := \mathbf{D}\varphi(\mathbf{s})^{-1}$. We have

$$\|\mathbf{A}^{-1}\| = \|\mathbf{A}^{-1}\mathbf{B}^{-1}\mathbf{B}\| \leq \|(\mathbf{B}\mathbf{A})^{-1}\|\|\mathbf{B}\|.$$

We can now apply the Banach Lemma 1.15 to $\|(\mathbf{B}\mathbf{A})^{-1}\|$ and obtain

$$\|\mathbf{A}^{-1}\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{1} - \mathbf{B}\mathbf{A}\|}.$$

Requiring $\|1 - \mathbf{BA}\| < 1/2$ makes sure that the Banach Lemma can be applied. This inequality becomes

$$\begin{aligned} \|1 - \mathbf{D}\varphi(\mathbf{s})^{-1}\mathbf{D}\varphi(\mathbf{x})\| &= \|\mathbf{D}\varphi(\mathbf{s})^{-1}(\mathbf{D}\varphi(\mathbf{s}) - \mathbf{D}\varphi(\mathbf{x}))\| \\ &\leq \|\mathbf{D}\varphi(\mathbf{s})^{-1}\| \|\mathbf{D}\varphi(\mathbf{s}) - \mathbf{D}\varphi(\mathbf{y})\| \\ &\leq \sigma \|\mathbf{D}\varphi(\mathbf{s})^{-1}\| \|\boldsymbol{\eta}\| \quad (1.26) \\ &\leq \sigma\delta \|\mathbf{D}\varphi(\mathbf{s})^{-1}\| \stackrel{!}{<} 1/2. \end{aligned}$$

This yields $2\sigma\delta < \|\mathbf{D}\varphi(\mathbf{s})^{-1}\|$ and the inequality (1.24).

To prove the final inequality (1.25) we have to recall the fundamental theorem of calculus which ensures

$$\varphi(\mathbf{x}) - \varphi(\mathbf{y}) = \int_0^1 \mathbf{D}\varphi(\mathbf{y} + t(\mathbf{x} - \mathbf{y}))(\mathbf{x} - \mathbf{y}) dt.$$

Applying this theorem to our case and using (1.23) we get

$$\begin{aligned} \|\varphi(\mathbf{x})\| &= \left\| \int_0^1 \mathbf{D}\varphi(\mathbf{s} + t\boldsymbol{\eta})\boldsymbol{\eta} dt \right\| \leq \|\boldsymbol{\eta}\| \int_0^1 \|\mathbf{D}\varphi(\mathbf{x})\| dt \\ &\leq 2\|\mathbf{D}\varphi(\mathbf{s})\| \|\boldsymbol{\eta}\|. \end{aligned}$$

This is the right hand side of inequality (1.25).

To prove the left hand side of the inequality we use

$$\mathbf{D}\varphi(\mathbf{s})^{-1}\varphi(\mathbf{x}) = \mathbf{D}\varphi(\mathbf{s})^{-1} \int_0^1 \mathbf{D}\varphi(\mathbf{s} + t\boldsymbol{\eta})\boldsymbol{\eta} dt,$$

which we transform into

$$\boldsymbol{\eta} = \mathbf{D}\varphi(\mathbf{s})^{-1}\varphi(\mathbf{x}) + \int_0^1 (1 - \mathbf{D}\varphi(\mathbf{s})^{-1}\mathbf{D}\varphi(\mathbf{x})) \boldsymbol{\eta} dt.$$

By (1.26) we are now able to estimate

$$\begin{aligned} \|\mathbf{D}\varphi(\mathbf{s})^{-1}\varphi(\mathbf{x})\| &\geq \|\boldsymbol{\eta}\| \left(1 - \left\| \int_0^1 (1 - \mathbf{D}\varphi(\mathbf{s})^{-1}\mathbf{D}\varphi(\mathbf{x})) dt \right\| \right) \\ &\geq \|\boldsymbol{\eta}\|/2. \end{aligned}$$

Therefore

$$\|\boldsymbol{\eta}\|/2 \leq \|\mathbf{D}\varphi(\mathbf{s})^{-1}\varphi(\mathbf{x})\| \leq \|\mathbf{D}\varphi(\mathbf{s})^{-1}\| \|\varphi(\mathbf{x})\|,$$

which completes the proof. \square

By the help of this lemma we are now able to prove a lemma that provides a stopping criteria for nonlinear iterative methods by the use of the relative nonlinear residual norm $\|\varphi(\mathbf{x})\|/\|\varphi(\mathbf{s}_0)\|$.

Lemma 1.42 (Lemma 5.2.1, [25]) *Let the standard assumptions hold and let $\delta > 0$ be small enough so that the conclusions of Lemma 1.41 hold for $\mathbf{x} \in \mathcal{B}(\mathbf{s}, \delta)$.*

Then for all $\mathbf{x}, \mathbf{s}_0 \in \mathcal{B}(\mathbf{s}, \delta)$

$$\frac{\|\boldsymbol{\eta}\|}{4\|\boldsymbol{\eta}_0\|\kappa(\mathbf{D}\varphi(\mathbf{s}))} \leq \frac{\|\varphi(\mathbf{x})\|}{\|\varphi(\mathbf{s}_0)\|} \leq \frac{4\kappa(\mathbf{D}\varphi(\mathbf{s}))\|\boldsymbol{\eta}\|}{\|\boldsymbol{\eta}_0\|}.$$

Proof. By Lemma 1.41 we have

$$\begin{aligned} \frac{\|\boldsymbol{\eta}\|}{2\|\mathbf{D}\varphi(\mathbf{s})^{-1}\|} &\leq \|\varphi(\mathbf{x})\| \leq 2\|\mathbf{D}\varphi(\mathbf{s})\|\|\boldsymbol{\eta}\|, \quad \text{and} \\ \frac{1}{2\|\mathbf{D}\varphi(\mathbf{s})\|\|\boldsymbol{\eta}_0\|} &\leq \frac{1}{\|\varphi(\mathbf{s}_0)\|} \leq \frac{2\|\mathbf{D}\varphi(\mathbf{s})^{-1}\|}{\|\boldsymbol{\eta}_0\|} \end{aligned}$$

Combining these two inequalities yields the stated inequality. \square

1.3 Existing Techniques

1.3.1 Newton's Method

The best known method to find a zero of a system of nonlinear equations is Newton's method. It can be derived very easily by cutting the Taylor expansion of a nonlinear operator after the first order term. Let $\mathbf{D}\varphi(\mathbf{x})$ denote the Fréchetderivative of φ at \mathbf{x} . In the vicinity of the zero \mathbf{s} of φ the following expansion holds,

$$\mathbf{0} = \varphi(\mathbf{s}) = \varphi(\mathbf{x}) + \mathbf{D}\varphi(\mathbf{x})\boldsymbol{\eta}(\mathbf{x}) + O(\|\boldsymbol{\eta}(\mathbf{x})\|^2).$$

Solving for the error vector we get

$$\boldsymbol{\eta}(\mathbf{x}) \approx -\mathbf{D}\varphi(\mathbf{x})^{-1}\varphi(\mathbf{x}).$$

Now we use $\mathbf{s} = \mathbf{x} + \boldsymbol{\eta}(\mathbf{x})$ giving rise to the definition of an iteration $\mathbf{s}_{i+1} := \mathbf{s}_i + \boldsymbol{\eta}(\mathbf{s}_i)$ which yields Newton's well known method

$$\mathbf{s}_{i+1} := \mathbf{s}_i - \mathbf{D}\varphi(\mathbf{s}_i)^{-1}\varphi(\mathbf{s}_i).$$

An algorithm for Newton's method is presented in Algorithm 1.3.

It is well known that Newton's method converges \mathcal{Q} -quadratically to a single zero of φ if the starting vector \mathbf{s}_0 is sufficiently close to the solution and the standard assumptions hold (see for example [25, pp. 71]). If $\mathbf{D}\varphi$ is not Lipschitz continuous, Newton's method converges \mathcal{Q} -superlinearly to a single zero of φ . However, if \mathbf{s} is a multiple zero of φ the convergence of Newton's method with respect to this solution is only linear.

Another important result about Newton's method is the *Kantorovich Theorem*. This theorem guarantees the convergence of Newton's method for initial guesses with certain properties. We cite the theorem without proof

CONVER-
GENCE OF
NEWTON'S
METHOD

Algorithm 1.3 Basic Newton Algorithm.

```

1  function Newton( $\varphi()$ ,  $\mathbf{D}\varphi()$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ):vector;
2     $\mathbf{r}_0 := \varphi(\mathbf{s}_0)$ ;                                {initial residual}
3     $i := 0$ ;
4    while  $\|\mathbf{r}_i\| > \varepsilon\|\mathbf{r}_0\|$  do
5       $\mathbf{J}_i := \mathbf{D}\varphi(\mathbf{s}_i)$ ;                        {derivative}
6       $\mathbf{h}_i := -\mathbf{J}_i^{-1}\mathbf{r}_i$ ;                       {factorize and solve}
7       $\mathbf{s}_{i+1} := \mathbf{s}_i + \mathbf{h}_i$ ;                   {new solution}
8       $\mathbf{r}_{i+1} := \varphi(\mathbf{s}_{i+1})$ ;               {new residual}
9       $i := i + 1$ 
10   end;
11   Newton :=  $\mathbf{s}_i$ 
12 end Newton;

```

from Stoer [38, pp. 326]. For a proof we refer to Ortega and Rheinboldt [28] or Rheinboldt and Antosiewicz [2].

Theorem 1.43 (Newton-Kantorovich) *Given an operator $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ and the convex set $\mathcal{M}_0 \subset \mathcal{M}$. Let φ be continuously differentiable on \mathcal{M}_0 . Moreover let φ meet the following conditions for an initial guess $\mathbf{s}_0 \in \mathcal{M}_0$:*

1. $\|\mathbf{D}\varphi(\mathbf{x}) - \mathbf{D}\varphi(\mathbf{y})\| \leq \gamma\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{M}_0$,
2. $\|\mathbf{D}\varphi(\mathbf{s}_0)^{-1}\| \leq \beta$,
3. $\|\mathbf{D}\varphi(\mathbf{s}_0)^{-1}\varphi(\mathbf{s}_0)\| \leq \alpha$.

We define constants $h := \alpha\beta\gamma$ and

$$\rho_{+,-} := \frac{1 \pm \sqrt{1 - 2h}}{h}\alpha.$$

If $h \leq 1/2$ and $\overline{\mathcal{B}}(\mathbf{s}_0, \rho_-) \subset \mathcal{M}_0$ the following statement holds: The sequence $\{\mathbf{s}_k\}$ defined by

$$\mathbf{s}_{k+1} := \mathbf{s}_k - \mathbf{D}\varphi(\mathbf{s}_k)^{-1}\varphi(\mathbf{s}_k) \quad \text{for } k = 0, 1, \dots$$

stays in $\mathcal{B}(\mathbf{s}_0, \rho_-)$ and converges to the unique root of φ in $\mathcal{M}_0 \cap \mathcal{B}(\mathbf{s}_0, \rho_+)$. \square

Although Newton's method converges very fast and uses few steps in general, one step could be very expensive. The two main tasks in every step are

DISADVANTAGES OF
NEWTON'S
METHOD

- (a) computing the derivative (line 5),
- (b) solving the linear system (line 6).

In the dense case Kelley [25] estimates that the cost for computing $\mathbf{D}\varphi$ by finite differences is n times the cost of an evaluation of φ , the amount of work for solving the linear system is of order n^3 .

Several suggestions have been made to reduce the costs per step of Newton's method. However, every attempt to solve the problem has resulted in a convergence order less than 2. Among the proposed solutions are *Quasi Newton methods* that address task (a) by approximating the first derivative of φ (or its inverses) in some way, and *inexact Newton methods* that address task (b) by solving the linear system of Newton's method approximately.

1.3.2 Quasi-Newton methods

Definition 1.44 A quasi-Newton method is a Newton method in which the derivative of $\varphi(\mathbf{s}_i)$ is replaced by an approximation of $\mathbf{D}\varphi(\mathbf{s}_i)$.

Among the approaches to reduce the costs per step of Newton's method the following can be categorized as quasi-Newton methods:

- the chord method and chord type methods,
- high-order methods,
- difference approximation methods,
- Broyden's method.

The Chord Method. This is a Newton method that replaces the derivative of $\mathbf{D}\varphi$ by the constant matrix $\mathbf{J}_0 \equiv \mathbf{D}\varphi(\mathbf{s}_0)$ throughout the algorithm. Thus \mathbf{J}_0 has to be computed only once at the beginning of the algorithm. Also the factorization can take place before the main loop. This way the amount of work for solving the linear system in every step is reduced to $O(n^2)$, which is about the same as for a matrix vector product. An algorithm is given in Algorithm 1.4.

Algorithm 1.4 The chord method.

```

function Chord( $\varphi()$ ,  $\mathbf{D}\varphi()$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ):vector;
   $\mathbf{r}_0 := \varphi(\mathbf{s}_0)$ ;                                {initial residual}
   $\mathbf{J}_0 := \mathbf{D}\varphi(\mathbf{s}_0)$ ;                       {initial derivative}
   $i := 0$ ;
  while  $\|\mathbf{r}_i\| > \varepsilon\|\mathbf{r}_0\|$  do
     $\mathbf{h}_i := -\mathbf{J}_0^{-1}\mathbf{r}_i$ ;                          {just solve}
     $\mathbf{s}_{i+1} := \mathbf{s}_i + \mathbf{h}_i$ ;                    {new solution}
     $\mathbf{r}_{i+1} := \varphi(\mathbf{s}_{i+1})$ ;                {new residual}
     $i := i + 1$ 
  end;
  Chord :=  $\mathbf{s}_i$ 
end Chord;

```

CONVER-
GENCE

In general the convergence order of the chord method is 1 provided that approximation is useful and the standard assumptions hold.

VARIATIONS
OF THE CHORD
METHOD

A slight modification of the chord method is using any approximation \mathbf{J} for $\mathbf{D}\varphi(\mathbf{s})$ instead of $\mathbf{D}\varphi(\mathbf{s}_0)$.

Another way of minimizing work is the direct approximation of the inverse of the derivative of φ . In this case $\mathbf{D}\varphi(\mathbf{x})^{-1}$ is replaced by $\mathbf{M}(\mathbf{x})$ using that the matrix vector multiplication of $\mathbf{M}(\mathbf{x})$ with $\varphi(\mathbf{s}_i)$ is less expensive than factorizing the matrix. In doing so one hopes that estimating the approximate inverse in each step and multiplying with it, is more accurate than using forward and backward substitution of a constant matrix.

A variant of this method consists in leaving M constant throughout the iteration. The convergence order of this method is also 1 provided that the approximation is useful and the standard assumptions hold.

High-Order Methods: the Shamanskii Method. The *Shamanskii method* combines the approach of the chord method with Newton's method by introducing a new parameter m in the following scheme.

$$\begin{aligned} \mathbf{s}_i^{(0)} &:= \mathbf{s}_i, \\ \mathbf{s}_i^{(j+1)} &:= \mathbf{s}_i^{(j)} - \mathbf{D}\varphi(\mathbf{s}_i)^{-1}\varphi(\mathbf{s}_i^{(j)}) \quad \text{for } 0 \leq j < m, \\ \mathbf{s}_{i+1} &:= \mathbf{s}_i^{(m)}. \end{aligned}$$

Thus for $m = 0$ Shamanskii's method becomes Newton's method, for $m = \infty$ it becomes the chord method.

In [25] Kelley proves that the rate of convergence of the Shamanskii method with respect to \mathbf{s}_i and \mathbf{s}_{i+1} is \mathcal{Q} -superlinear provided that the standard assumptions hold.

CONVER-
GENCE

The algorithm for the Shamanskii method is an extension of the first two Newton-like algorithms. Another loop is introduced into the first while loop to perform the inner iteration with respect to j . A simple version of the Shamanskii method with constant m is presented in Algorithm 1.5. We like to note that Brent [5] has considered a similar algorithm that varies m in each outer loop with respect to some optimality condition.

ALGORITHM

Difference approximation to $\mathbf{D}\varphi$. Motivated by the definition of the G -derivative one might approximate the

Algorithm 1.5 The Shamanskii method.

function Shamanskii($\varphi()$, $D\varphi()$, s_0 , ε , m):**vector**;
 $r_0 := \varphi(s_0)$; {initial residual}
 $i := 0$;
while $\|r_i\| > \varepsilon \|r_0\|$ **do**
 $J_i := D\varphi(s_i^{(0)})$; {derivative}
 $h_i^{(0)} := -J_i^{-1} r_i$; {factorize and solve}
 $s_i^{(1)} := s_i + h_i^{(0)}$; {initial intermediate solution}
 $r_i^{(1)} := \varphi(s_i^{(0)})$; {initial intermediate residual}
for $j := 1$ **to** $m - 1$ **do**
 $h_i^{(j)} := -J_i^{-1} r_i^{(j)}$; {just solve}
 $s_i^{(j+1)} := s_i^{(j)} + h_i^{(j)}$; {intermediate solution}
 $r_i^{(j+1)} := \varphi(s_i^{(j+1)})$; {intermediate residual}
end;
 $s_{i+1} := s_i^{(m)}$; {new solution}
 $r_{i+1} := r_i^{(m)}$; {new residual}
 $i := i + 1$
end;
Shamanskii := s_i
end Shamanskii;

matrix vector product of the derivative of a nonlinear operator φ with a vector p by

$$D\varphi(x)p \approx \frac{\varphi(x + hp) - \varphi(x)}{h}.$$

The following analysis is roughly copied from Kelley [25]. Suppose that an error is made in the evaluation of $\varphi(x)$ of size $\eta(x)$ with $\|\eta(x)\| \leq \varepsilon$ for all x . Then this approximation becomes

$$\begin{aligned} D\varphi(x)p &\approx \frac{\varphi(x + hp) - \varphi(x) + \eta(x + hp) - \eta(x)}{h} \\ &= D\varphi(x)p + O(h + \varepsilon/h). \end{aligned}$$

The quantity inside the O -term is minimized by setting $h = \sqrt{\epsilon}$. However, Kelley wishes \mathbf{x} and \mathbf{p} to be of the same size. If this is not the case, Kelley suggests h to be scaled to reflect his demand. The choice

$$h = \sqrt{\epsilon} \frac{\|\mathbf{x}\|}{\|\mathbf{p}\|}$$

meets all these conditions.

Concluding the considerations made so far we define the difference approximation to the directional derivative $\mathbf{D}\varphi(\mathbf{x})\mathbf{p}$ to be

$$\mathbf{d}_\varphi^h(\mathbf{x}, \mathbf{p}) := \begin{cases} 0, & \mathbf{p} = \mathbf{0}, \\ \frac{\varphi(\mathbf{x} + h\|\mathbf{x}\|/\|\mathbf{p}\|\mathbf{p}) - \varphi(\mathbf{x})}{h\|\mathbf{x}\|/\|\mathbf{p}\|}, & \mathbf{p}, \mathbf{x} \neq \mathbf{0}, \\ \frac{\varphi(h\|\mathbf{p}\|/\|\mathbf{p}\|\mathbf{p}) - \varphi(\mathbf{x})}{h/\|\mathbf{p}\|}, & \mathbf{p} \neq \mathbf{0}, \mathbf{x} = \mathbf{0}. \end{cases}$$

If \mathbf{p} is chosen to be a canonical unit vector \mathbf{e}_j , one gets the column vectors of the Jacobian. This leads to a simpler difference approximation of the j^{th} column vector of $\mathbf{D}\varphi(\mathbf{x})$ by

$$\mathbf{d}_\varphi^h(\mathbf{x})_j := \begin{cases} \frac{\varphi(\mathbf{x} + h\|\mathbf{x}\|\mathbf{e}_j) - \varphi(\mathbf{x})}{h\|\mathbf{x}\|}, & \mathbf{x} \neq \mathbf{0}, \\ \frac{\varphi(h\mathbf{e}_j) - \varphi(\mathbf{x})}{h}, & \mathbf{x} = \mathbf{0}. \end{cases}$$

We denote the full approximation of the derivative of φ containing all column vectors generated by the above rule by $\mathbf{D}_\varphi^h(\mathbf{x})$.

Kelley points out that in general $\mathbf{D}_\varphi^h(\mathbf{x})\mathbf{p} \neq \mathbf{d}_\varphi^h(\mathbf{x}, \mathbf{p})$ since the approximation of the directional derivative of φ is usually not a linear operator of \mathbf{p} .

Broyden's Method. Kelley [25] defines a quasi-Newton method as a Newton method that provides a separate iteration for updating the approximation of the Jacobian in each step. Thus the Newton iteration becomes

$$\mathbf{s}_{i+1} := \mathbf{s}_i - \mathbf{J}_i^{-1}\varphi(\mathbf{s}_i), \quad \text{with } \mathbf{J}_{i+1} := \Psi(\mathbf{J}_i).$$

The construction of J_{i+1} determines the quasi-Newton method.

Broyden's method computes J_{i+1} by a rank-1 update of J_i

$$J_{i+1} := J_i + \frac{\varphi(\mathbf{s}_{i+1})\mathbf{h}_i^H}{\mathbf{h}_i^H\mathbf{h}_i} \quad \text{with } \mathbf{h}_i = \mathbf{s}_{i+1} - \mathbf{s}_i. \quad (1.27)$$

Thus the updated approximation of $D\varphi$ satisfies the *secant equation*

$$J_{i+1}\mathbf{h}_i = \varphi(\mathbf{s}_{i+1}) - \varphi(\mathbf{s}_i).$$

Broyden's method has the following advantages over Newton's method.

- A dense rank-1 updated Jacobian can be inverted recursively by the *Sherman-Morrison* formula

$$(\mathbf{M} + \mathbf{x}\mathbf{y}^H)^{-1} = \left(\mathbf{1} - \frac{\mathbf{M}^{-1}\mathbf{x}\mathbf{y}^H}{1 + \mathbf{y}^H\mathbf{M}^{-1}\mathbf{x}} \right) \mathbf{M}^{-1}.$$

In the context of Broyden's method this formula gets applied to (1.27) with $\mathbf{M} = J_i$, $\mathbf{x} = \varphi(\mathbf{s}_i)/\|\mathbf{h}_i\|$ and $\mathbf{y} = \mathbf{h}_i/\|\mathbf{h}_i\|$.

A few transformations and the assumption $J_0 = \mathbf{1}$ are needed (see [25, p.125 – 126]) to derive the following recursive formulas for \mathbf{h}_{i+1} and J_{i+1} ,

$$\mathbf{h}_{i+1} := -\frac{J_i^{-1}\varphi(\mathbf{s}_{i+1})}{1 + \mathbf{h}_i^H J_i^{-1}\varphi(\mathbf{s}_{i+1})/(\mathbf{h}_i^H\mathbf{h}_i)},$$

$$J_{i+1}^{-1} := \prod_{j=0}^i \left(\mathbf{1} + \frac{\mathbf{h}_{j+1}\mathbf{h}_j^H}{\mathbf{h}_j^H\mathbf{h}_j} \right).$$

Only the vectors \mathbf{h}_i have to be stored to perform Broyden's iteration.

- Broyden's method needs only one function evaluation per step.

Kelley has proven local \mathcal{Q} -superlinear convergence for Broyden's method if the standard assumptions hold.

An algorithm for Broyden's method is presented in Algorithm 1.6.

Algorithm 1.6 Broyden's method.

```

function Broyden( $\varphi()$ ,  $s_0$ ,  $\varepsilon$ ):vector;
   $r_0 := \varphi(s_0)$ ;                                {initial residual}
  if  $\|r_0\| > \varepsilon\|r_0\|$  then
     $h_0 := -r_0$ ;                                {initial step}
     $s_1 := s_0 + h_0$ ;                            {first improvement}
     $r_1 := \varphi(s_1)$ ;                          {residual}
     $i := 1$ ;
    while  $\|r_i\| > \varepsilon\|r_0\|$  do
       $n_i := -\varphi(s_i)$ ;                        {nominator of the new step}
      for  $j := 1$  to  $i - 1$  do
         $n_i := n_i + h_{j+1}(h_j^H n_i)/(h_j^H h_j)$ ;
      end;
       $h_i := n_i/(1 - h_i^H n_i/(h_i^H h_i))$ ;      {new step}
       $s_{i+1} := s_i + h_i$ ;                    {new solution}
       $r_{i+1} := \varphi(s_{i+1})$ ;                {new residual}
       $i := i + 1$ ;
    end
  end;
  Broyden :=  $s_i$ ;
end Broyden;

```

1.3.3 Inexact Newton methods

Inexact Newton methods deal with the approximative solution of the linear system, that has to be solved in every step of the Newton iteration. Their definition is due to Dembo, Eisenstat and Steihaug [8].

Definition 1.45 A Newton method is called *inexact Newton method* if the linear system $\mathbf{D}\varphi(\mathbf{s}_i)\mathbf{h}_i = -\varphi(\mathbf{s}_i)$ for \mathbf{h}_i is solved *approximately*, such that the following condition holds

$$\|\mathbf{D}\varphi(\mathbf{s}_i)\mathbf{h}_i + \varphi(\mathbf{s}_i)\| \leq \theta_i \|\varphi(\mathbf{s}_i)\|. \quad (1.28)$$

The term θ_i is called *forcing term*.

In an inexact Newton method the tolerance θ_i determines how accurate the linear system has to be solved in step i . This is not a new idea: Actually in every Newton iteration the linear system is solved approximately due to round off errors (if the method is implemented on a computer with finite arithmetic). Though in general Newton's method converges \mathcal{Q} -quadratically on such systems. In an inexact Newton method this observation is used as a principle, combined with the question: How accurate has the linear system to be solved in step i to achieve a certain convergence rate? The sequence of forcing terms is used to control the convergence behavior of the inexact Newton method.

Another point of view is that it is not justified to determine an approximation \mathbf{s}_i to machine precision if \mathbf{s}_i is far away from the solution and \mathbf{h}_i does not significantly reduce the error. Computing \mathbf{s}_i up to an accuracy beyond the required tolerance is called *oversolving* and can be very expensive. On the other hand for iterates very close to the solution it is very important to determine \mathbf{s}_i very precisely to achieve the desired tolerance quickly. From this point of view the question is: How accurate has the linear system to be solved in order to minimize the amount of work?

The answer to these questions is hidden in the appropriate choice of the sequence of forcing terms. Variants of choices have been considered by Eisenstat and Walker in [11].

The questions of inexact Newton methods become very important if the linear system of the Newton method is solved by an iterative method, because condition (1.28) is a very common stopping criterion for linear iterative methods. The combination of Newton with linear iterative solvers forms a large class of inexact Newton methods often referred to as *Newton-iterative* methods.

Kelley [25] points out that inexact methods are sometimes referred to as *truncated Newton methods* in the context of optimization. A basic algorithm for an inexact Newton method is given in Algorithm 1.7.

Algorithm 1.7 An inexact Newton method.

```

function iNewton( $\varphi(\cdot)$ ,  $\mathbf{D}\varphi(\cdot)$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ,  $\theta$ ):vector;
   $\mathbf{r}_0 := \varphi(\mathbf{s}_0)$ ;                                {initial residual}
   $i := 0$ ;
  while  $\|\mathbf{r}_i\| > \varepsilon\|\mathbf{r}_0\|$  do
     $\mathbf{J}_i := \mathbf{D}\varphi(\mathbf{s}_i)$ ;                    {derivative}
    Find  $\mathbf{h}_i$  with
       $\|\varphi(\mathbf{s}_i) + \mathbf{J}_i\mathbf{h}_i\| \leq \theta_i\|\varphi(\mathbf{s}_i)\|$ ;
     $\mathbf{s}_{i+1} := \mathbf{s}_i + \mathbf{h}_i$ ;                {new solution}
     $i := i + 1$ 
  end;
  iNewton :=  $\mathbf{s}_i$ 
end iNewton;

```

Dembo, Eisenstat and Steihaug proved some theorems on the convergence of inexact methods. The first one, which we cite here, states the local convergence of an inexact Newton method, which is linear in the weighted norm $\|\cdot\|_* := \|\mathbf{D}\varphi(\mathbf{s})\cdot\|$. We give the theorems without proof.

CONVER-
GENCE

Theorem 1.46 (Theorem 2.3, [8]) Assume that $\theta_i \leq \theta_{\max} < t < 1$. There exists $\varepsilon > 0$ such that, if $\|\mathbf{s}_0 - \mathbf{s}\| \leq \varepsilon$, then the sequence of inexact Newton iterates $\{\mathbf{s}_i\}$ con-

verges to \mathbf{s} . Moreover the convergence is linear in the sense that

$$\|\mathbf{s}_{i+1} - \mathbf{s}\|_* \leq t \|\mathbf{s}_i - \mathbf{s}\|_*,$$

where $\|\cdot\|_* := \|\mathbf{D}\varphi(\mathbf{s})\cdot\|$. □

The sequence of forcing terms can be used to control the convergence rate. Dembo, Eisenstat and Steihaug also proved a theorem that summarizes the consequences of the choice of a certain sequence. To understand their theorem, the following definition is needed.

Definition 1.47 ([8]) $\mathbf{D}\varphi$ is Hölder continuous with exponent p , ($0 < p \leq 1$) at \mathbf{s} if there exists $\sigma > 0$ such that

$$\|\mathbf{D}\varphi(\mathbf{x}) - \mathbf{D}\varphi(\mathbf{s})\| \leq \sigma \|\mathbf{x} - \mathbf{s}\|^p$$

for $\|\mathbf{y} - \mathbf{x}\|$ sufficiently small.

With this definition, the following theorem can be formulated.

Theorem 1.48 (Corollary 3.5, [8]) Assume that the inexact Newton iterates $\{\mathbf{s}_i\}$ converge to \mathbf{s} .

Then:

- $\mathbf{s}_i \rightarrow \mathbf{s}$ superlinearly if $\lim_{i \rightarrow \infty} \theta_i = 0$;
- $\mathbf{s}_i \rightarrow \mathbf{s}$ with Q -order at least $1 + p$ if $\mathbf{D}\varphi$ is Hölder continuous with exponent p at \mathbf{s} and

$$\theta_i = O(\|\varphi(\mathbf{s}_i)\|^p) \quad \text{as } i \rightarrow \infty;$$

- $\mathbf{s}_i \rightarrow \mathbf{s}$ with R -order at least $1 + p$ if $\mathbf{D}\varphi$ is Hölder continuous with exponent p at \mathbf{s} and $\{\theta_i\}$ converges to 0 with R -order at least $1 + p$. □

Now that we know how the choice of the sequence of forcing terms influences the convergence rate, we are interested in particular choices of $\{\theta_i\}$. Eisenstat and Walker [11] have investigated this topic.

CHOOSING
THE FORCING
TERMS

They assume the following extension of the standard assumptions. Let $\mu := \max\{\|\mathbf{D}\varphi(\mathbf{s})\|, \|\mathbf{D}\varphi(\mathbf{s})^{-1}\|\}$ and $\mathcal{M} := \mathcal{B}(\mathbf{s}, \delta)$. Suppose that there is a $\delta > 0$ such that the following conditions on φ and \mathbf{s} hold

1. φ is continuously differentiable and $\mathbf{D}\varphi$ is nonsingular on \mathcal{M} ,
2. $\|\mathbf{D}\varphi(\mathbf{x})^{-1}\| \leq 2\mu$ for $\mathbf{x} \in \mathcal{M}$,
3. $\mathbf{D}\varphi$ is Lipschitz continuous with Lipschitz constant σ on \mathcal{M} ,
4. $\delta < 2/(\sigma\mu)$.

They give two choices for θ_i .

Choice 1 Given $\theta_0 \in [0, 1]$, for $i = 1, 2, \dots$ choose

$$\theta_i = \frac{\|\varphi(\mathbf{s}_i) - \varphi(\mathbf{s}_{i-1}) - \mathbf{D}\varphi(\mathbf{s}_{i-1})\mathbf{h}_{i-1}\|}{\|\mathbf{s}_{i-1}\|}$$

or

$$\theta_i = \frac{|\|\varphi(\mathbf{s}_i)\| - \|\varphi(\mathbf{s}_{i-1}) + \mathbf{D}\varphi(\mathbf{s}_{i-1})\mathbf{h}_{i-1}\||}{\|\mathbf{s}_{i-1}\|}.$$

Choice 2 Given $\gamma \in [0, 1]$, $\alpha \in (1, 2]$, and $\theta_0 \in [0, 1]$, choose

$$\theta_i = \gamma \left(\frac{\|\varphi(\mathbf{s}_i)\|}{\|\varphi(\mathbf{s}_{i-1})\|} \right)^\alpha, \quad i = 1, 2, \dots$$

Eisenstat and Walker note that θ_i according to Choice 1 directly reflects the agreement between φ and its linear local model at step $i - 1$. The second variant of Choice 1 may be more convenient to evaluate than the first one. They proved the following theorem.

Theorem 1.49 (Theorem 2.2, [11]) *Under the assumptions (1) – (4) given above on $\mathbf{D}\varphi$ and \mathbf{s} , if \mathbf{s}_0 is sufficiently near \mathbf{s} , then $\{\mathbf{s}_i\}$ produced by Algorithm 1.7 with $\{\theta_i\}$ given by Choice 1 remains in $\mathcal{B}(\mathbf{s}, \delta)$ and converges to \mathbf{s} with*

$$\|\mathbf{s}_{i+1} - \mathbf{s}\| \leq c \|\mathbf{s}_i - \mathbf{s}\| \|\mathbf{s}_{i-1} - \mathbf{s}\|, \quad i = 1, 2, \dots,$$

for a constant c independent of i . □

This theorem tells us that the convergence of Choice 1 is \mathcal{Q} -superlinear. It is even \mathcal{Q} -quadratic within two steps. As Eisenstat and Walker note, according to a similar analysis as in the case of the classical secant method, it also follows that the convergence is of R-order $(1 + \sqrt{5})/2$.

Choice 2 is the result of trying to obtain faster local convergence while retaining the potential advantages of Choice 1. The following theorem by Eisenstat and Walker shows that it offers attractive local convergence.

Theorem 1.50 (Theorem 2.3, [11]) *If \mathbf{s}_0 is sufficiently near \mathbf{s} , then under the assumptions (1) – (4) given above on $\mathbf{D}\varphi$ and \mathbf{s} , $\{\mathbf{s}_i\}$ produced by Algorithm 1.7 with $\{\theta_i\}$ given by Choice 2 remains in $\mathcal{B}(\mathbf{s}, \delta)$ and converges to \mathbf{s} .*

If $\gamma < 1$, then the convergence is of \mathcal{Q} -order α .

If $\gamma = 1$, then the convergence is of R-order α and of \mathcal{Q} -order p for every $p \in [1, \alpha)$. □

SAFEGUARD-
ING

In some cases θ_i can become small for one or more iterations while \mathbf{s}_i is still far from the solution. Therefore a method of *safeguarding* was suggested by Eisenstat and Walker to avoid a volatile decreasing in θ_i . The idea is that if θ_{i-1} is large at some point we do not allow subsequent θ -values to become much smaller until this has been justified over several iterations. For each choice they demand θ_i to be no less than a minimum value, but only if that minimum value is above a threshold. The minimum value is determined by raising θ_{i-1} to a power associated with the rate of convergence of the

choice (neglecting the safeguard). The threshold they chose was 0.1, which they note is somewhat arbitrary but turned out to be effective in both their experiments and in the experiments of Kelley.

In order to apply this safeguarding mechanism, the forcing terms have to be adjusted accordingly:

for Choice 1

$$\theta_i^s := \max \left\{ \theta_i, \theta_{i-1}^{\frac{1+\sqrt{5}}{2}} \right\}, \quad \text{whenever } \theta_{i-1}^{\frac{1+\sqrt{5}}{2}} > 0.1,$$

for Choice 2

$$\theta_i^s := \max \left\{ \theta_i, \gamma \theta_{i-1}^\alpha \right\}, \quad \text{whenever } \gamma \theta_{i-1}^\alpha > 0.1.$$

Eisenstat and Walker note that “it may be possible for each of the proposed choices to be greater than one”. Accordingly they demand, that another safeguard has to make sure that $\theta_i \in [0, 1)$ for each i . This is done by minimizing the sequence with a value $\theta_{\max} < 1$.

Algorithm 1.8 shows an implementation of an inexact Newton method with Choice 2 that uses a linear iterative solver. The implementation shown in this algorithm is focused on the choice of the sequence of forcing terms.

1.3.4 Classical Methods

In this section we review iterative methods that have been invented by well known classical mathematicians (Jacobi, Gauss) to solve their linear problems. They are in such a sense classical that their invention marks the beginning of iterative methods for linear systems. For more than 100 years until the rise of the Krylov space methods in the early 1970s, Jacobi, Gauss-Seidel and SOR have been the main choice for solving linear

Algorithm 1.8 A Newton iterative method with Choice 2 using the iterative method *ItMeth* to solve the linear system $\mathbf{J}_i \mathbf{h}_i = -\boldsymbol{\varphi}(\mathbf{s}_i)$.

```

1  function NewtonIt( $\boldsymbol{\varphi}()$ ,  $\mathbf{D}\boldsymbol{\varphi}()$ ,  $\mathbf{s}_0, \varepsilon, \theta_{\max}, \gamma, \alpha$ ):vector;
2     $\mathbf{r}_0 := \boldsymbol{\varphi}(\mathbf{s}_0)$ ;                                {initial residual}
3     $\theta_0 := \theta_{\max}$ ;
4     $i := 0$ ;
5    while  $\|\mathbf{r}_i\| > \varepsilon \|\mathbf{r}_0\|$  do
6       $\mathbf{J}_i := \mathbf{D}\boldsymbol{\varphi}(\mathbf{s}_i)$ ;                            {derivative}
7       $\mathbf{h}_i := \text{ItMeth}(\mathbf{J}_i, -\boldsymbol{\varphi}(\mathbf{r}_i), \theta_i)$ ;        {new step}
8       $\mathbf{s}_{i+1} := \mathbf{s}_i + \mathbf{h}_i$ ;                            {new solution}
9       $\mathbf{r}_{i+1} := \boldsymbol{\varphi}(\mathbf{s}_{i+1})$ ;                    {new residual}
10
11     {Forcing terms according to Choice 2}
12     if  $\gamma \theta_i^\alpha > 0.1$  then                            {first safeguard}
13        $\theta_{i+1} := \max\{\gamma(\|\mathbf{r}_{i+1}\|/\|\mathbf{r}_i\|)^\alpha, \gamma \theta_i^\alpha\}$ 
14     else
15        $\theta_{i+1} := \gamma(\|\mathbf{r}_{i+1}\|/\|\mathbf{r}_i\|)^\alpha$ 
16     end;
17      $\theta_{i+1} := \min\{\theta_{\max}, \theta_{i+1}\}$ ;        {second safeguard}
18      $i := i + 1$ 
19   end;
20   NewtonIt :=  $\mathbf{s}_i$ 
21 end NewtonIt;

```

systems whenever the application of an iterative solver was indicated. Although nowadays Krylov space based methods are the state of the art iterative solvers for linear systems of equations, classical methods have still some importance because

- they are very easy to implement,
- their behavior is well understood,
- they can be adjusted in a natural way to solve non-linear systems of equations,

- some of them have natural parallelization properties.

The last two properties motivate the ongoing research with respect to developing new methods based on the ideas of the classical methods (*multisplitting* and *overrelaxation*).

Moreover, if the performance of a state of the art linear solver stays below the expectations, a classical method often serves as a good preconditioner.

Our interest in these methods is their capability of solving nonlinear systems of equations. Nevertheless we shortly review the linear methods. Then we show their nonlinear interpretations and present some properties.

Let the linear system

$$\mathbf{A}\mathbf{s} = \mathbf{b} \quad (1.29)$$

REVIEW OF
LINEAR
SOLVERS

be given. We define a splitting of the system matrix \mathbf{A} by

$$\mathbf{A} =: \mathbf{L} + \mathbf{D} + \mathbf{U},$$

where \mathbf{L} is the strict lower triangular part of \mathbf{A} , \mathbf{D} its diagonal part and \mathbf{U} its strict upper triangular part. Rewriting (1.29) with respect to the definition of the splitting we get

$$(\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{s} = \mathbf{b}.$$

The aim of splitting the matrix \mathbf{A} into several parts is to find a part of \mathbf{A} that is easier to invert. This is the idea of the so called *splitting methods*. The given splitting is a very natural one. Two ways of developing an iterative method based on the given splitting are

JACOBI AND
GAUSS-
SEIDEL

$$\begin{aligned} \mathbf{D}\mathbf{s}_{i+1}^J &:= \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{s}_i^J, \\ (\mathbf{L} + \mathbf{D})\mathbf{s}_{i+1}^{\text{GS}} &:= \mathbf{b} - \mathbf{U}\mathbf{s}_i^{\text{GS}}. \end{aligned} \quad (1.30)$$

In these implicit rules J denotes the *Jacobi method* and GS denotes the *Gauss-Seidel method*. Of course one could also choose $\mathbf{D} + \mathbf{U}$ as the matrix to be inverted. This method is called *backward Gauss-Seidel method*.

Rewriting these recursive instructions with respect to the residual of the linear system $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{s}_i$ we get a formulation suitable for implementation

$$\begin{aligned}\mathbf{s}_{i+1}^J &:= \mathbf{s}_i^J + \mathbf{D}^{-1}\mathbf{r}_i^J, \\ \mathbf{s}_{i+1}^{GS} &:= \mathbf{s}_i^{GS} + (\mathbf{L} + \mathbf{D})^{-1}\mathbf{r}_i^{GS}.\end{aligned}$$

According to Definition 1.26, both methods are stationary with $\mathbf{G} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ and $\mathbf{d} = \mathbf{D}^{-1}\mathbf{b}$ for the Jacobi method and $\mathbf{G} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}$ and $\mathbf{d} = (\mathbf{L} + \mathbf{D})^{-1}\mathbf{b}$ for Gauss-Seidel.

CONVER-
GENCE

Several authors have proven that the Jacobi method converges for linear systems, whose system matrix is an H-matrix [3], and the Gauss-Seidel method converges for systems, the matrix of which is symmetric and positive definite [15].

As the contraction mapping theorem for linear systems, Corollary 1.37, shows, the methods converge linearly with an asymptotic convergence rate of $\rho(\mathbf{G})$. In general the Gauss-Seidel method converges faster than the Jacobi method. However, its application is restricted to a smaller class of matrices.

RELAXATION

The methods can be extended such that the new iterate is a weighted sum of the old iterate and the new value. Doing this on the vector level we obtain the *Jacobi overrelaxation method* and the *Gauss-Seidel overrelaxation method*, which we do not focus on any further.

Proceeding so on the component level, we get the same method for the Jacobi case. However, taking a weighted sum of the components of the old iterate and the new Gauss-Seidel value, we obtain the popular *successive overrelaxation method (SOR)*.

As (1.30) shows, the Gauss-Seidel method determines \mathbf{s}_{i+1} implicitly by

$$\mathbf{s}_{i+1}^{\text{GS}} := \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{s}_{i+1}^{\text{GS}} - \mathbf{U}\mathbf{s}_i^{\text{GS}}).$$

We determine the new iterate of SOR by introducing the weighted sum $\mathbf{s}_{i+1}^{\text{SOR}} := (1 - \omega)\mathbf{s}_i^{\text{SOR}} + \omega\mathbf{s}_{i+1}^{\text{GS}}$. The weight ω is called the *relaxation parameter*. We obtain

$$\mathbf{s}_{i+1}^{\text{SOR}} := (1 - \omega)\mathbf{s}_i^{\text{SOR}} + \omega\mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{s}_{i+1}^{\text{SOR}} - \mathbf{U}\mathbf{s}_i^{\text{SOR}}).$$

With a few transformations this implicit formulation of SOR can be turned into the explicit iteration rule

$$\mathbf{s}_{i+1}^{\text{SOR}} := \mathbf{s}_i^{\text{SOR}} + (\mathbf{L} + \omega^{-1}\mathbf{D})^{-1}\mathbf{r}_i^{\text{SOR}}.$$

SOR fits into the framework of *matrix splitting methods* because it corresponds to the matrix splitting $\mathbf{A} = \mathbf{L} + 1/\omega\mathbf{D} + (\omega - 1)/\omega\mathbf{D} + \mathbf{U}$. Its iteration matrix is $\mathbf{G} := -(\omega\mathbf{L} + \mathbf{D})^{-1}((\omega - 1)\mathbf{D} + \omega\mathbf{U})$.

The term *overrelaxation* is used if $\omega > 1$, which is the usual case. If $\omega < 1$ then the appropriate term is *underrelaxation*. In the case $\omega = 1$ the Gauss-Seidel method is retained.

Several authors have proved (see for example [3, p. 173]) that the useful interval for values of ω is $(0, 2)$. For some matrix types the optimum value of ω is known. However, in general it can be very difficult to find such an optimum. Given the optimum relaxation parameter for SOR, this can speedup the convergence significantly over the Gauss-Seidel method. However, since the method is still a stationary method, it cannot compete with Krylov space methods or polynomial acceleration.

In a natural way the Jacobi method, the Gauss-Seidel method and the SOR method can be adjusted to solve nonlinear systems of equations. However, to do this the methods have to be examined in terms of components

rather than vectors. Let us therefore examine the non-linear operator φ , a zero of which has to be found, in terms of its component functionals,

$$\varphi(\mathbf{s}) = [\varphi_1(\mathbf{s}), \dots, \varphi_n(\mathbf{s})]^\top.$$

In the same manner let us define the implicit iteration operator \mathbf{f} for which we would like to find the solution of $\mathbf{f}(\mathbf{s}_{i+1}, \mathbf{s}_i) = \mathbf{0}$ with $\mathbf{f}(\mathbf{x}, \mathbf{y}) = [f_1(\mathbf{x}, \mathbf{y}), \dots, f_n(\mathbf{x}, \mathbf{y})]^\top$. Then we can specify the nonlinear Jacobi and the nonlinear Gauss-Seidel method implicitly by

$$\mathbf{f}_i^J(\mathbf{x}, \mathbf{y}) := \varphi_i(y_1, \dots, y_{i-1}, x_i, y_{i+1}, \dots, y_n), \quad (1.31)$$

$$\mathbf{f}_i^{\text{GS}}(\mathbf{x}, \mathbf{y}) := \varphi_i(x_1, \dots, x_i, y_{i+1}, \dots, y_n). \quad (1.32)$$

The nonlinear SOR method can be specified by $s_{i+1,j}^{\text{SOR}} := (1-\omega)s_{i,j}^{\text{SOR}} + \omega s_{i+1,j}^{\text{GS}}$. Since this formulation involves the Gauss-Seidel method, we solve for this value and obtain

$$s_{i+1,j}^{\text{GS}} = s_{i,j}^{\text{SOR}} + \frac{1}{\omega} (s_{i+1,j}^{\text{SOR}} - s_{i,j}^{\text{SOR}}).$$

Thus the nonlinear Gauss-Seidel specification (1.32) can be used to find an analogous specification for nonlinear SOR

$$\mathbf{f}_i^{\text{SOR}}(\mathbf{x}, \mathbf{y}) := \varphi_i(x_1, \dots, x_{i-1}, y_i + \frac{1}{\omega}(x_i - y_i), y_{i+1}, \dots, y_n). \quad (1.33)$$

These methods are natural extensions of the linear methods, since they turn into the classical linear methods if φ is linear. A generic algorithm for nonlinear classical methods is given in Algorithm 1.9.

Algorithm 1.9 Nonlinear classical method.

```

function nlClassic( $\varphi()$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ,  $[\omega]$ ):vector;
   $\mathbf{r}_0 := \varphi(\mathbf{s}_0)$ ;                                {initial residual}
   $i := 0$ ;
  while  $\|\mathbf{r}_i\| > \varepsilon \|\mathbf{r}_0\|$  do
    for  $j := 1$  to  $n$  do
      perform one Newton step on
      Jacobi:
         $\mathbf{t}_j := \text{solve } \varphi_j(\mathbf{s}_{i[\dots j]} + \mathbf{e}_j \mathbf{x} + \mathbf{s}_{i[j \dots]})$  for  $\mathbf{x}$ ;
      Gauss–Seidel:
         $\mathbf{t}_j := \text{solve } \varphi_j(\mathbf{s}_{i+1[\dots j]} + \mathbf{e}_j \mathbf{x} + \mathbf{s}_{i[j \dots]})$  for  $\mathbf{x}$ ;
         $s_{i+1,j} := \mathbf{t}_j$ ;
      SOR:
         $\mathbf{t}_j := \text{solve } \varphi_j(\mathbf{s}_{i+1[\dots j]} + \mathbf{e}_j \mathbf{x} + \mathbf{s}_{i[j \dots]})$  for  $\mathbf{x}$ ;
         $s_{i+1,j} := (1 - \omega)s_{i,j} + \omega \mathbf{t}_j$ 
    end;
     $\mathbf{s}_{i+1} := \mathbf{t}$ ;                                {new solution}
     $\mathbf{r}_{i+1} := \varphi(\mathbf{s}_{i+1})$ ;                    {new residual}
     $i := i + 1$ 
  end;
  nlClassic :=  $\mathbf{s}_i$ 
end nlClassic;

```

An implication of the close relationship to the linear methods is, that the nonlinear Jacobi method converges locally if $\mathbf{D}\varphi(s)$ exists and is an H-matrix, the Gauss–Seidel method converges if $\mathbf{D}\varphi(s)$ exists and is symmetric positive definite. The following corollary (taken in part from Ortega and Rheinboldt [28]) is mainly an implication of the implicit function theorem 1.39. It contains the guarantee that the nonlinear classical methods are well-defined. It also gives their local Q-convergence rate.

LOCAL
CONVERGENCE

Corollary 1.51 (cf. 10.3.4, [28]) Suppose that $f : (\mathcal{M} \times \mathcal{M}) \subset (\mathbb{R}^n \times \mathbb{R}^n) \rightarrow \mathbb{R}^n$ has continuous partial derivatives $\partial_1 f$ and $\partial_2 f$ on an open neighborhood $\mathcal{B}_0 \subset \mathcal{M}$ of a point

$\mathbf{s} \in \mathcal{M}$ at which $\mathbf{f}(\mathbf{s}, \mathbf{s}) = \mathbf{0}$. Assume further that $\partial_1 \mathbf{f}(\mathbf{s}, \mathbf{s})$ is nonsingular and that

$$\sigma := \rho(-\partial_1 \mathbf{f}(\mathbf{s}, \mathbf{s})^{-1} \partial_2 \mathbf{f}(\mathbf{s}, \mathbf{s})) < 1.$$

Then there is an open sphere $\mathcal{B} := \mathcal{B}(\mathbf{s}, \delta) \subset \mathcal{B}_0$ such that, for any $\mathbf{y} \in \mathcal{B}$, the equation $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ has a unique solution $\mathbf{x} = \mathbf{g}(\mathbf{y})$ in \mathcal{B} . Therefore, the sequence

$$\mathbf{s}_{i+1} := \mathbf{g}(\mathbf{s}_i), \quad i = 0, 1, \dots, \quad (1.34)$$

is well-defined for any $\mathbf{s}_0 \in \mathcal{B}$ and satisfies

$$\mathbf{f}(\mathbf{s}_{i+1}, \mathbf{s}_i) = \mathbf{0}, \quad i = 0, 1, \dots$$

Moreover, \mathbf{s} is a point of attraction of the iteration defined by (1.34). The Q -convergence is of order 1 and the Q -factor is σ .

Proof. From the implicit function theorem 1.39 it follows that there are open neighborhoods \mathcal{B}_1 and \mathcal{B}_2 of \mathbf{s} such that, for any $\mathbf{y} \in \mathcal{B}_2$, the equation $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ has a unique solution $\mathbf{x} = \mathbf{g}(\mathbf{y})$ in \mathcal{B}_1 ; of course in particular $\mathbf{s} = \mathbf{g}(\mathbf{s})$. The theorem also shows that the F-derivative of the mapping $\mathbf{g} : \mathcal{B}_2 \rightarrow \mathcal{B}_1$ at \mathbf{s} exists and that

$$\mathbf{Dg}(\mathbf{s}) = -\partial_1 \mathbf{f}(\mathbf{s}, \mathbf{s})^{-1} \partial_2 \mathbf{f}(\mathbf{s}, \mathbf{s}).$$

Now for arbitrary $\epsilon > 0$ Lemma 1.35 ensures the existence of a norm on \mathbb{R}^n such that

$$\|\mathbf{Dg}(\mathbf{s})\| \leq \sigma + \epsilon.$$

Moreover, the F-differentiability of \mathbf{g} at \mathbf{s} implies that there is a δ so that $\mathcal{B} = \mathcal{B}(\mathbf{s}, \delta) \subset \mathcal{B}_1 \cap \mathcal{B}_2$ and

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{s}) - \mathbf{Dg}(\mathbf{s})(\mathbf{x} - \mathbf{s})\| \leq \epsilon \|\mathbf{x} - \mathbf{s}\|, \quad \forall \mathbf{x} \in \mathcal{B}.$$

Therefore,

$$\begin{aligned}\|\mathbf{g}(\mathbf{x}) - \mathbf{s}\| &\leq \|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{s}) - \mathbf{D}\mathbf{g}(\mathbf{s})(\mathbf{x} - \mathbf{s})\| + \|\mathbf{D}\mathbf{g}(\mathbf{s})(\mathbf{x} - \mathbf{s})\| \\ &\leq (\sigma + 2\epsilon)\|\mathbf{x} - \mathbf{s}\|, \quad \forall \mathbf{x} \in \mathcal{B}.\end{aligned}$$

Hence, $\mathbf{g}(\mathcal{B}) \subset \mathcal{B}$ and the \mathcal{Q} -convergence is of order 1. By construction, for any $\mathbf{x}_0 \in \mathcal{B}$ the sequence (1.34) satisfies $\mathbf{g}(\mathbf{s}_{i+1}, \mathbf{s}_i) = \mathbf{0}$, $i = 0, 1, \dots$. Since ϵ was arbitrary, σ is the \mathcal{Q} -factor. \square

With this corollary we are now able to reformulate a theorem by Ortega and Rheinboldt for the local convergence of the nonlinear Jacobi, Gauss-Seidel and SOR method.

Theorem 1.52 (10.3.5, [28]) *Let $\varphi : \mathcal{M} \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ be continuously differentiable in an open neighborhood $\mathcal{B}_0 \subset \mathcal{M}$ of a point $\mathbf{s} \in \mathcal{M}$ for which $\varphi(\mathbf{s}) = \mathbf{0}$. Consider the decomposition*

$$\mathbf{D}\varphi(\mathbf{s}) = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

of $\mathbf{D}\varphi(\mathbf{s})$ into its diagonal, strictly lower-, and strictly upper-triangular parts, and suppose that \mathbf{D} is nonsingular and $\rho(\mathbf{G}(\mathbf{s})) < 1$, where $\mathbf{G}(\mathbf{s})$ is defined by

$$\begin{aligned}\mathbf{G}^J(\mathbf{s}) &= -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}), \\ \mathbf{G}^{GS}(\mathbf{s}) &= -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}, \\ \mathbf{G}^{SOR}(\mathbf{s}) &= -(\omega\mathbf{L} + \mathbf{D})^{-1}((\omega - 1)\mathbf{D} + \omega\mathbf{U}), \quad \omega > 0.\end{aligned}$$

Then there exists an open sphere $\mathcal{B} = \mathcal{B}(\mathbf{s}, \delta)$ in \mathcal{B}_0 such that, for any $\mathbf{s}_0 \in \mathcal{B}$, there is a unique sequence $\{\mathbf{s}_i\} \in \mathcal{B}$ which satisfies the nonlinear Jacobi, Gauss-Seidel and SOR description. Moreover $\lim_{i \rightarrow \infty} \mathbf{s}_i = \mathbf{s}$ and the \mathcal{Q} -order of the iteration is 1 with \mathcal{Q} -factor $\rho(\mathbf{G}(\mathbf{s}))$.

Proof. Because φ is continuously F-differentiable on \mathcal{B}_0 , the implicit iteration operator \mathbf{f} for Jacobi (1.31),

Gauss-Seidel (1.32) and SOR (1.33) has continuous partial derivatives on some set $\mathcal{B}' \times \mathcal{B}'$, where \mathcal{B}' is an open subset of \mathcal{B}_0 of the form

$$\mathcal{B}' := \{\mathbf{x} \in \mathcal{B}_0 \mid |x_i - s_i| < \delta', i = 1, \dots, n\}.$$

Moreover a straightforward computation shows that

$$\begin{aligned} \partial_1 \mathbf{f}^J(\mathbf{s}, \mathbf{s}) &= \mathbf{D}, & \partial_2 \mathbf{f}^J(\mathbf{s}, \mathbf{s}) &= \mathbf{L} + \mathbf{U}, \\ \partial_1 \mathbf{f}^{\text{GS}}(\mathbf{s}, \mathbf{s}) &= \mathbf{L} + \mathbf{D}, & \partial_2 \mathbf{f}^{\text{GS}}(\mathbf{s}, \mathbf{s}) &= \mathbf{U}, \\ \partial_1 \mathbf{f}^{\text{SOR}}(\mathbf{s}, \mathbf{s}) &= \mathbf{L} + \frac{1}{\omega} \mathbf{D}, & \partial_2 \mathbf{f}^{\text{SOR}}(\mathbf{s}, \mathbf{s}) &= \frac{\omega - 1}{\omega} \mathbf{D} + \mathbf{U}. \end{aligned}$$

Since \mathbf{D} is nonsingular $\partial_1 \mathbf{f}(\mathbf{s}, \mathbf{s})$ is also nonsingular. Hence

$$\mathbf{G}(\mathbf{s}) = -\partial_1 \mathbf{f}(\mathbf{s}, \mathbf{s})^{-1} \partial_2 \mathbf{f}(\mathbf{s}, \mathbf{s}),$$

and the result follows directly from Corollary 1.51. \square

Ortega and Rheinboldt [28, pp. 320 – 331] also proved for SOR that the R-factor is precisely $\rho(\mathbf{G}(\mathbf{s}))$ (and this also applies to Jacobi and Gauss-Seidel). Moreover they proved that more than one step of the inner scalar Newton iteration does not improve the R-factor of the iteration. Even if the inner Newton iteration is executed until convergence the R-factor still remains $\rho(\mathbf{G}(\mathbf{s}))$. We therefore rephrase this result according to some famous words of Kahan: *Once is enough*.

1.4 A Sample Problem

To illustrate the performance and properties of the presented algorithms for solving nonlinear systems

of equations we use in all our examples the Chandrasekhar H-equation taken from Kelley [25]. The Chandrasekhar H-functional describes the exit distribution in radiative transfer. It is implicitly defined by

$$c \in (0, 1),$$

$$H : [0, 1] \rightarrow \mathbb{R}, \mu \mapsto \left(1 - \frac{c}{2} \int_0^1 \frac{\mu H(\nu)}{\mu + \nu} d\nu \right)^{-1}.$$

To find the values of H for the argument $\mu \in [0, 1]$ we define a nonlinear functional φ

$$\mu \in [0, 1], c \in (0, 1),$$

$$\varphi : ([0, 1] \rightarrow \mathbb{R}) \rightarrow \mathbb{R}, H(\cdot) \mapsto H(\mu) - \left(1 - \frac{c}{2} \int_0^1 \frac{\mu H(\nu)}{\mu + \nu} d\nu \right)^{-1}.$$

In order to find values of H for the argument μ the equation $\varphi(H) = 0$ has to be solved. Therefore the integral is discretized by the composite midpoint rule. Integrals on $[0, 1]$ are approximated by

$$\int_0^1 f(x) dx \approx \frac{1}{n} \sum_{j=1}^n f(x_j) \quad \text{with } x_j = (j - \frac{1}{2})/n \text{ for } 1 \leq j \leq n.$$

Setting $y_i = H(x_i)$ the discretized problem turns into finding the zero of a nonlinear operator φ , the i^{th} component of which is

$$\varphi_i(\mathbf{y}) = y_i - \left(1 - \frac{c}{2n} \sum_{j=1}^n \frac{x_i y_j}{x_i + x_j} \right)^{-1}.$$

Kelley points out that the discretized problem has its own physical meaning. Moreover because H has a singularity at $x = 0$, the solution of the discrete problem is

not even a first-order accurate approximation to that of the continuous problem.

Let us define the matrix

$$\mathbf{M} := \begin{bmatrix} \frac{x_1}{x_1+x_1} & \frac{x_1}{x_1+x_2} & \cdots & \frac{x_1}{x_1+x_n} \\ \frac{x_2}{x_2+x_1} & \frac{x_2}{x_2+x_2} & \cdots & \frac{x_2}{x_2+x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_n}{x_n+x_1} & \frac{x_n}{x_n+x_2} & \cdots & \frac{x_n}{x_n+x_n} \end{bmatrix}.$$

Then the derivative of φ is

$$\mathbf{D}\varphi(\mathbf{y}) = 1 - \frac{c}{2n} \text{Diag}(\mathbf{x} - \mathbf{y})^2 \mathbf{M}.$$

1.5 Conclusion: Numerical Results

We summarize our overview of existing techniques for solving nonlinear systems of equations by presenting numerical results for the Chandrasekhar H-equation. Parameters chosen for this example are $n := 100$ and $c := 0.9999$ — a case for which Kelley (25) notes, that the Jacobian at the solution is not very well conditioned. Every method had to achieve a relative residual norm of 10^{-7} to stop. Due to the condition number of the derivative of φ at \mathbf{s} , $\kappa(\mathbf{D}\varphi(\mathbf{s})) \sim 10^2$ this stopping criteria could not take all of the six tested methods below a relative error of 10^{-7} . Two of them — the chord method and the inexact Newton method — stopped before a relative error of 10^{-7} could be achieved. The good news is that all methods achieved a relative error of less than 10^{-6} .

The methods — the Newton method, the Chord method, the Shamanskii method, Broyden's method, nonlinear SOR and an inexact Newton method — have been implemented in Matlab 5.3. This version of Matlab has been chosen because it still offers a flop count command, and we like to compare the algorithms in terms of flops.

The following implementation details have to be noted:

Shamanskii method The parameter m that specifies, how often the derivative of φ should be reused, was set to $m := 2$ — a choice suggested by Kelley (25) for this problem.

SOR Empirically we found that setting the relaxation parameter $\omega := 1.39647$ turned out to be a very good choice. According to what has been said at the end of Section 1.3.4 the number of steps for the inner scalar Newton was limited to 1.

Inexact Newton For the inexact Newton method the linear iterative method GMRES was used, the function of which we describe in detail in Chapter 2. We used the built in implementation of Matlab 5.3. We implemented Choice 2 for generating the sequence of forcing terms. The following parameters were used: $\theta_{\max} := 0.9999$, $\alpha := 2$, $\gamma := 0.9$ as suggested by Kelley (25). For the threshold we chose 0.1 as suggested by Eisenstadt and Walker (11).

Table 1.3 Flops and execution times of existing methods for solving the Chandrasekhar H-equation. Flops were counted with the `flops` command of Matlab 5.3. Times were measured in Matlab on a Sun SPARC 336 MHz with 3 GB main memory.

Method	Iterations	Mflops	Time(s)
Broyden	13	2.2	0.23
Newton-GMRES	8	2.8	0.23
SOR	17	3.2	(3.33)
Shamanskii	6	6.1	0.32
Newton's method	8	7.2	0.32
Chord method	262	27.8	3.52

Table 1.3 lists the number of iterations, floating point operations and times taken on a SPARC 366 Mhz architecture with 3 GB main memory in Matlab 5.3. The time of nonlinear SOR is given in parentheses. The reason for this is that all other algorithms can be implemented in a high level matrix-vector notation, letting Matlab choose highly optimized routines for solving triangular linear systems, matrix vector multiplication etc. These routines are at least BLAS level 2 routines that are able to exploit the benefits of modern microprocessors such as pipelining architectures, parallel ALUs and big caches. SOR however, had to be implemented on the component level. No matrix operations were used, just some scalar products, which makes it BLAS level 1 only. Since Matlab is an interpreted language, no optimizations (not even optimal cache usage) could be applied

to SOR. This is why SOR achieves the lousy flop rate of 96 Kflop/s compared to the Newton method, which achieves about 2.3 Mflops/s. Even the chord method, which is very slow for this example, achieved a flop rate, that is one magnitude higher than SOR: 789.7 Kflops/s. Of course the bad behavior of SOR can be improved by using a compiled language instead of an interpreted one, giving the compiler a chance to do some optimization. Another improvement would be a blocked variant of SOR.

Figure 1.2 Convergence of different methods on the Chandrasekhar H-equation for $n = 100$ and $c = 0.9999$.

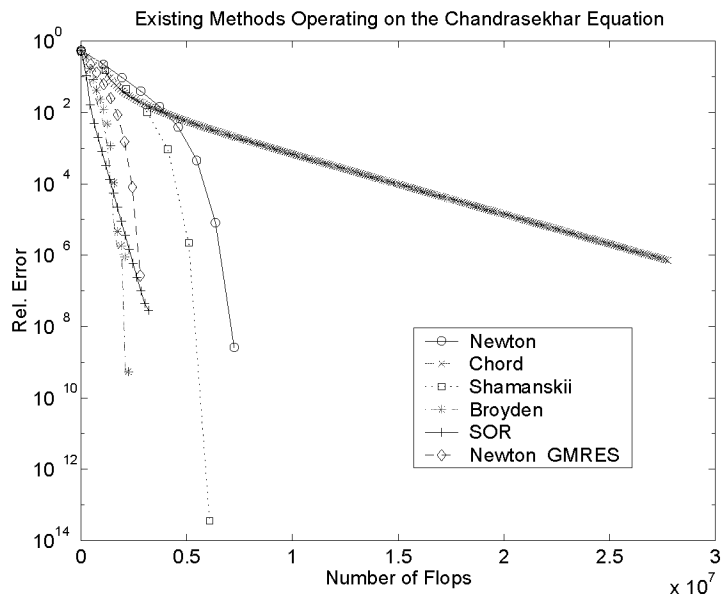


Figure 1.2 shows the convergence curves of the algorithms. In this figure the relative error is plotted against the number of flops needed to achieve this accuracy. The figure reveals the local linear convergence behavior of the chord method and SOR. We will use this linear behavior in later chapters to develop nonlinear Krylov space methods. However, the linear convergence behavior does not take place until a few steps have been carried out. For SOR this happens after about 3 steps,

the chord method needs about 22 steps to establish the linear convergence rate. For both methods it is interesting to see, that the convergence rate is better, before the linear behavior takes place.

The other methods show superlinear convergence as has been proved before. Nearly all of them went far below the required tolerance of 10^{-7} . This is a typical property of superlinear convergence: Near the solution one step results in a huge improvement.


The algorithms can be ranked in different ways: With respect to time (the less the better), flops needed to find the solution (the less the better) or flop rate (the higher the better). For the Chandrasekhar H-equation the overall winner with respect to time and flops is Broyden's method. The second place is also very clear: it is taken by Newton-GMRES. However, the other candidates show not such a clear behavior: Although the chord method behaves really bad in this example, there are other problems for which the chord method is able to beat the Newton method (see for example (25, ch. 5.6)). SOR is the worst algorithm with respect to the flop rate. However, implemented in an environment which allows for better optimization it could achieve a better time, competitive to the others. Finally Newton's method, although beaten by many algorithms with respect to flops achieves the highest flop rate — a mark for good parallelization properties.

RANKINGS OF
THE
ALGORITHMS

Chapter 2

Linear Krylov Space Methods

2.1 Motivation

efore we are able to understand nonlinear Krylov space methods, we have to analyze the well known linear Krylov space methods. Furthermore we have to know their properties and how they depend on each other. In order to build up a network of dependencies of properties we embed linear Krylov space methods into *correction space methods* and *projection methods*, followed by a more detailed discussion of the four Krylov space methods FOM, GMRES, BiCG and QMR operating on general coefficient matrices. FOM and GMRES will be adopted to nonlinear problems in the next chapter.

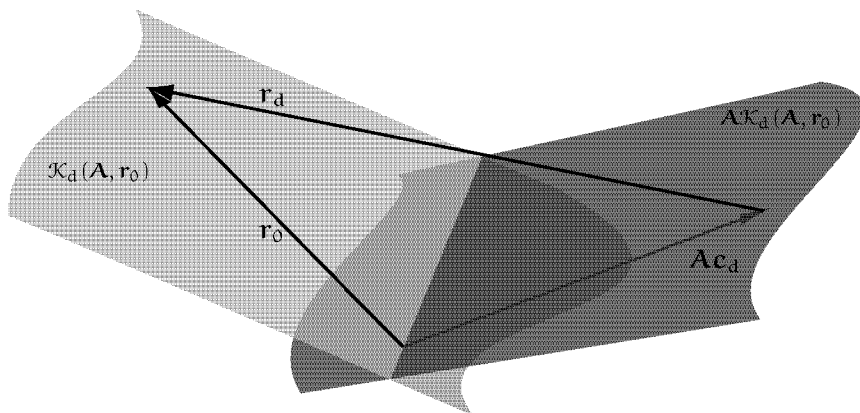
2.2 Introduction

In 1952 the conjugate gradient method was invented to solve linear systems with a hermitian positive def-

inite coefficient matrix [27]. Until today many people have been working on generalizations of this method [31, 12, 13, 16]. The conjugate gradient method has been adopted to linear systems with symmetric indefinite and nonhermitian coefficient matrices. Several theories around these so-called *Krylov space methods* have been developed.

In contrast to vector extrapolation methods, Krylov space methods can be explained geometrically. They even have a lot of geometric properties. Therefore, the basic mechanism of Krylov space methods for solving linear systems can be explained within a few sentences:

Figure 2.1 A basic step of a linear Krylov space method.



Given a nonsingular matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^n$. Suppose we want to solve the following linear system for $\mathbf{s} \in \mathbb{R}^n$:

$$\mathbf{A}\mathbf{s} = \mathbf{b}. \quad (2.1)$$

Krylov space methods solve this system by building subsequent Krylov spaces $\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$ with $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$. An important property of this sequence is: they are em-

bedded. That means the Krylov space of dimension d contains all Krylov spaces of smaller dimension.

Let \mathbf{c}_d be an element of $\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$. In step d some Krylov space methods solve the linear least squares problem

$$\min_{\mathbf{z}_d} \|\mathbf{r}_0 - \mathbf{A}\mathbf{c}_d\|. \quad (2.2)$$

$\mathbf{A}\mathbf{c}_d$ is an element of $\mathbf{A}\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$, whereas \mathbf{r}_0 is an element of $\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$. If $\mathbf{A}\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$ and $\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$ are equivalent (i.e. $\mathcal{K}_d(\mathbf{A}, \mathbf{r}_0)$ is an *invariant subspace* with respect to \mathbf{A}) — and it is guaranteed that they become equivalent after at most n steps —, the least squares problem becomes a linear system that is exactly solvable.

- The hope of all Krylov space methods is that the sequence of residuals $\mathbf{r}_d := \mathbf{r}_0 - \mathbf{A}\mathbf{c}_d$ is (strongly) monotonously decreasing and the required precision of the result is obtained long before the n^{th} step. As we will see this hope cannot always be fulfilled.
- The aim of these methods is that in every step the least squares problem (2.2) can be solved by using as much information as possible from the step before and therefore the computational effort can be minimized.

2.3 Correction Space Methods

Definition 2.1 *Given the linear system of equation (2.1). We want to solve this system by an iterative method. If the iterates \mathbf{s}_i of the solution of the linear system can be expressed as*

$$\mathbf{s}_i = \mathbf{s}_0 + \mathbf{c}_i, \quad (2.3)$$

a sum of a starting vector s_0 and a correction vector $c_i \in \mathcal{C}_i$ with $\mathcal{C}_i \subset \mathbb{R}^n$ being a vector space, the iterative method for solving the linear system is called a correction space method. The space \mathcal{C}_i is called the correction space.

Algorithm 2.1 Basic algorithm template for a correction space method. Input parameters are the matrix A , the right hand side b of (2.1), an initial guess for the solution s_0 and a relative tolerance ε for the stopping criteria. The function value is the approximate solution of the system according to the given relative tolerance ε .

```

function CORSPM( $A$ ,  $b$ ,  $s_0$ ,  $\varepsilon$ ): vector;
   $i := 0$ ;
   $r_0 := b - As_0$ ;
  while  $\|r_i\| > \varepsilon \|b\|$  do

    Find  $c_{i+1}$  in  $\mathcal{C}_{i+1}$ ;

     $s_{i+1} := s_0 + c_{i+1}$ ;
     $r_{i+1} := r_0 - Ac_{i+1}$ ;
     $i := i + 1$ ;
  end;
  CORSPM :=  $s_i$ ;
end CORSPM;

```

In the i^{th} step of a correction space method a correction vector $c_i \in \mathcal{C}_i$ has to be found. An approximate solution s_i is determined by the combination of an initial guess s_0 plus the correction vector. This procedure is repeated until convergence. A basic template of an algorithm for a correction space method is shown in Algorithm 2.1.

In correction space methods residual vectors play an important role. With the ansatz of s_i in equation (2.3) we get the following expression for the residual vector in step i

$$r_i = b - As_i = r_0 - Ac_i. \quad (2.4)$$

A correction space method is a very general concept. In fact all practical iterative methods for solving linear systems are correction space methods.

Example 2.2 (Stationary Methods) The class of the stationary methods, that among others contains the classical methods described in Chapter 1, uses the following iteration rule to create approximations $s_i, i = 1, 2, \dots$ of the solution of the linear system of (2.1):

$$s_{i+1} := Gs_i + d, \quad (2.5)$$

with G and d being a matrix and a vector accordingly chosen such that $s := s_\infty$ solves (2.1). In other words $(I - G) = M^{-1}A$ and $d = M^{-1}b$ for M being a nonsingular matrix.

With the residual vector of the given system (2.1) $r_i = b - As_i$ we can reformulate (2.5) as

$$s_{i+1} = s_i + M^{-1}r_i = s_0 + M^{-1} \sum_{j=0}^i r_j. \quad (2.6)$$

This equation tells us that every stationary method is a simple correction space method using

$$C_i = [M^{-1}r_0, M^{-1}r_1, \dots, M^{-1}r_i]$$

as the basis for its correction space. Using this correction space we always choose the vector $z_i = [1, \dots, 1]^T$, $c_i = C_i z_i$ for the correction. The constant vector z_i is one reason why the method is called *stationary*.

As we have seen in Example 2.2, stationary methods choose a constant correction vector $c_i \in \mathcal{C}_i$ in every step. When the maximum dimension of the space is reached, they do not stop. In this case they start to generate correction vectors combined by more than one basis vector, thus creating the fractions of composites of the basis vectors necessary in general. Trying to find a solution quickly, we see that these methods obviously choose neither an optimum space nor an optimum combination of the basis vectors.

More elaborated methods try to make use of the correction space: In every step they generate a different correction vector with respect to some given optimization criteria, e.g. minimizing some norm of the error vector or the residual vector.

For finding an optimal method the following questions arise:

1. How do we minimize the norm of the error vector (residual vector) over the correction space?
2. How to choose the correction space?

The following section answers the first question. The second question will be answered in Section 2.4.

2.3.1 Projection Methods

Definition 2.3 *A projection method is a correction space method that determines $c_i \in \mathcal{C}_i$ in every step i such that $r_i = r_0 - A c_i$ is orthogonal to a given vector space $\mathcal{P}_i \subset \mathbb{R}^n$. We call \mathcal{P}_i the projection space.*

Let P_i be the basis of \mathcal{P}_i in step i of a projection method. According to definition 2.3 a correction vector c_i is determined in every step i of a projection method such that

$$P_i^H r_0 = P_i^H A c_i \quad \text{with} \quad c_i \in \mathcal{C}_i. \quad (2.7)$$

Let C_i be the basis of \mathcal{C}_i and let $c_i := C_i z_i$. Solving (2.7) for z_i we get $z_i = (P_i^H A C_i)^{-1} P_i^H r_0$. Therefore, in step i s_i and r_i become

$$\begin{aligned} s_i &= s_0 + C_i (P_i^H A C_i)^{-1} P_i^H r_0, \\ r_i &= r_0 - A C_i (P_i^H A C_i)^{-1} P_i^H r_0. \end{aligned}$$

A template algorithm for a projection method is shown in Algorithm 2.2.

Algorithm 2.2 Basic template algorithm for a projection method. Input parameters are the matrix \mathbf{A} , the right hand side \mathbf{b} of (2.1), an initial guess for the solution \mathbf{s}_0 and a relative tolerance ε for the stopping criteria. The function returns the approximate solution of the system according to the given relative tolerance ε .

```

1  function PROM( $\mathbf{A}$ ,  $\mathbf{b}$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ): vector;
2       $i := 0$ ;
3       $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$ ;
4      while  $\|\mathbf{r}_i\| > \varepsilon\|\mathbf{b}\|$  do
5          choose  $\mathcal{C}_{i+1}$ ;
6          choose  $\mathcal{P}_{i+1}$ ;
7           $\mathbf{C}_{i+1} :=$  basis of  $\mathcal{C}_{i+1}$ ;
8           $\mathbf{P}_{i+1} :=$  basis of  $\mathcal{P}_{i+1}$ ;
9           $\mathbf{Q}_{i+1} := \mathbf{A}\mathbf{C}_{i+1}$ ;
10
11          $\mathbf{z}_{i+1} := (\mathbf{P}_{i+1}^H \mathbf{Q}_{i+1})^{-1} \mathbf{P}_{i+1}^H \mathbf{r}_0$ ;
12          $\mathbf{q}_{i+1} := \mathbf{Q}\mathbf{z}_{i+1}$ ;
13          $\mathbf{r}_{i+1} := \mathbf{r}_0 - \mathbf{q}_{i+1}$ ;
14          $i := i + 1$ ;
15     end;
16      $\mathbf{c}_i := \mathbf{C}_i \mathbf{z}_i$ ;
17      $\mathbf{s}_i := \mathbf{s}_0 + \mathbf{c}_i$ ;
18     PROM :=  $\mathbf{s}_i$ ;
19 end PROM;
```

In the following we will concentrate mostly on an analysis of the residual vectors \mathbf{r}_i . For simplicity reasons let us define $\mathcal{Q}_i := \mathbf{A}\mathcal{C}_i := \{\mathbf{A}\mathbf{a} \mid \mathbf{a} \in \mathcal{C}_i\}$ to be the space containing all elements in \mathcal{C}_i multiplied by \mathbf{A} . With this definition a projection method has to determine a $\mathbf{q}_i \in \mathcal{Q}_i$ such that $\mathbf{r}_i = \mathbf{r}_0 - \mathbf{q}_i \perp \mathcal{P}_i$. Furthermore we will use the $\mathbf{Q}_i := \mathbf{A}\mathbf{C}_i$ as the basis of \mathcal{Q}_i .

EXISTENCE
AND
UNIQUENESS
OF \mathbf{c}_i

The following lemma from Eiermann and Ernst [10] gives conditions for the existence and uniqueness of a $\mathbf{q}_i \in \mathcal{Q}_i$ in a projection method.

Lemma 2.4 *Given two subspaces \mathcal{Q} and \mathcal{P} of \mathbb{R}^n .*

The following statements hold.

1. *For an arbitrary $\mathbf{r}_0 \in \mathbb{R}^n$ there exists a $\mathbf{q} \in \mathcal{Q}$ such that $\mathbf{r} = \mathbf{r}_0 - \mathbf{q} \perp \mathcal{P}$ if and only if $\mathcal{Q} + \mathcal{P}^\perp = \mathbb{R}^n$.*
2. *Such a \mathbf{q} is unique if and only if*

$$\mathcal{Q} \cap \mathcal{P}^\perp = \{\mathbf{0}\}. \quad (2.8)$$

Proof. Let \mathbf{P}^\perp and \mathbf{Q} be bases of \mathcal{P}^\perp and \mathcal{Q} respectively.

1. If $\mathbf{q} \in \mathcal{Q}$ exists, then $\mathbf{r}_0 = \mathbf{r} + \mathbf{q}$ with $\mathbf{r} \in \mathcal{P}^\perp$ and $\mathbf{q} \in \mathcal{Q}$. It is obvious that \mathbf{r}_0 must be in $\mathcal{Q} + \mathcal{P}^\perp$. Since \mathbf{r}_0 was arbitrarily chosen, $\mathcal{Q} + \mathcal{P}^\perp$ must be equal to \mathbb{R}^n .

On the other hand let $\mathcal{Q} + \mathcal{P}^\perp = \mathbb{R}^n$. Then we can find (not necessarily unique) vectors \mathbf{z}_1 and \mathbf{z}_2 such that $\mathbf{r}_0 = \mathbf{P}^\perp \mathbf{z}_1 + \mathbf{Q} \mathbf{z}_2$. Thus $\mathbf{r} = \mathbf{P}^\perp \mathbf{z}_1$ and $\mathbf{q} = \mathbf{Q} \mathbf{z}_2 \in \mathcal{Q}$ exists.

2. If $\mathcal{Q} \cap \mathcal{P}^\perp = \{\mathbf{0}\}$ then there exist unique vectors \mathbf{z}_1 and \mathbf{z}_2 such that $\mathbf{r}_0 = \mathbf{P}^\perp \mathbf{z}_1 + \mathbf{Q} \mathbf{z}_2$. Using $\mathbf{r} = \mathbf{P}^\perp \mathbf{x}_1$ and $\mathbf{q} = \mathbf{Q} \mathbf{x}_2$ we see that $\mathbf{x}_1 = \mathbf{z}_1$ and $\mathbf{x}_2 = \mathbf{z}_2$ is the only solution for $\mathbf{r} = \mathbf{r}_0 - \mathbf{q} \in \mathcal{P}^\perp$ and therefore $\mathbf{q} = \mathbf{z}_2$ is unique.

On the other hand if $\mathcal{Q} \cap \mathcal{P}^\perp \neq \{\mathbf{0}\}$ there exist several pairs \mathbf{z}_1 and \mathbf{z}_2 such that $\mathbf{r}_0 = \mathbf{P}^\perp \mathbf{z}_1 + \mathbf{Q} \mathbf{z}_2$. Therefore, \mathbf{q} cannot be unique.

□

If \mathbf{q}_i is unique then \mathbf{c}_i is also unique, since \mathbf{A} is nonsingular.

An immediate consequence of this lemma is the following corollary:

Corollary 2.5 *Given two spaces \mathcal{P} and \mathcal{Q} of \mathbb{R}^n with their bases \mathbf{P} and \mathbf{Q} respectively. Let $\mathbf{r}_0 \in \mathbb{R}^n$ be arbitrarily chosen.*

A unique $\mathbf{q} \in \mathcal{Q}$ for the problem $\mathbf{r} = \mathbf{r}_0 - \mathbf{q} \perp \mathcal{P}$ exists if and only if $\dim(\mathcal{P}) = \dim(\mathcal{Q})$ and $\mathbf{P}^H \mathbf{Q}$ is nonsingular.

Proof. Let us denote the dimensions of \mathcal{P} and \mathcal{Q} as d and e respectively. If \mathbf{q} exists, Lemma 2.4 tells us that $\mathcal{Q} + \mathcal{P}^\perp = \mathbb{R}^n$, and since \mathbf{q} is unique $\mathcal{Q} \cap \mathcal{P}^\perp = \{0\}$. Therefore,

$$\begin{aligned} \dim(\mathcal{Q} + \mathcal{P}^\perp) &= \dim(\mathcal{Q}) + \dim(\mathcal{P}^\perp) \\ &= e + (n - d) \stackrel{!}{=} n = \dim(\mathbb{R}^n) \end{aligned}$$

This equation implies that $e = d$. $\mathcal{A} \mathcal{C} \cap \mathcal{P}^\perp = \{0\}$ directly implies that $\mathbf{P}^H \mathbf{Q}$ is a nonsingular matrix.

On the other hand let $e = d$. The nonsingular property of $\mathbf{P}^H \mathbf{Q}$ implies directly $\mathcal{Q} \cap \mathcal{P}^\perp = \{0\}$. Then we have

$$\dim(\mathcal{Q} + \mathcal{P}^\perp) = \dim(\mathcal{Q}) + \dim(\mathcal{P}^\perp) = e + (n - e) = n.$$

Therefore, $\mathcal{Q} + \mathcal{P}^\perp = \mathbb{R}^n$, which by Lemma 2.4 implies that \mathbf{q} is unique. \square

Given the definition of projection methods let \mathbf{P}_i , \mathbf{Q}_i be bases of \mathcal{P}_i , \mathcal{Q}_i respectively. Now we can rewrite $\mathbf{q}_i = \mathbf{Q}_i \mathbf{z}_i$ and with the orthogonality condition we have

WHY ARE
THEY CALLED
"PROJECTION
METHODS"?

$$\mathbf{P}_i^H \mathbf{r}_i = \mathbf{P}_i^H \mathbf{r}_0 - \mathbf{P}_i^H \mathbf{Q}_i \mathbf{z}_i.$$

From Corollary 2.5 we know, that $\mathbf{P}_i^H \mathbf{Q}_i$ is an invertible matrix if \mathbf{q}_i is unique. Therefore, we obtain a unique

$z_i = (\mathbf{P}_i^H \mathbf{Q}_i)^{-1} \mathbf{P}_i^H \mathbf{r}_0$. Using this result for \mathbf{r}_i (2.6) yields in

$$\mathbf{r}_i = \mathbf{r}_0 - \mathbf{Q}_i (\mathbf{P}_i^H \mathbf{Q}_i)^{-1} \mathbf{P}_i^H \mathbf{r}_0 = (1 - \mathbf{Q}_i (\mathbf{P}_i^H \mathbf{Q}_i)^{-1} \mathbf{P}_i^H) \mathbf{r}_0.$$

Lemma 2.6 *Let \mathcal{P} and \mathcal{Q} be two subspaces of \mathbb{R}^n equal in dimension. Let \mathbf{P} and \mathbf{Q} be their bases.*

$\Psi := \mathbf{Q}(\mathbf{P}^H \mathbf{Q})^{-1} \mathbf{P}^H$ is an oblique projector from $\mathbb{R}^n = \mathcal{Q} + \mathcal{P}^\perp$ to \mathcal{Q} orthogonal to \mathcal{P} (or, equivalently, along \mathcal{P}^\perp).

Proof. First we show that Ψ is idempotent.

$$\Psi^2 = \mathbf{Q}(\mathbf{P}^H \mathbf{Q})^{-1} (\mathbf{P}^H \mathbf{Q}) (\mathbf{P}^H \mathbf{Q})^{-1} \mathbf{P}^H = \mathbf{Q}(\mathbf{P}^H \mathbf{Q})^{-1} \mathbf{P}^H = \Psi$$

This is why Ψ is a projector. Left to be shown is that $\mathcal{R}(\Psi) = \mathcal{Q}$ and $\mathcal{N}(\Psi) = \mathcal{P}^\perp$. We do that within two steps.

- We show that $\mathcal{Q} = \mathcal{R}(\Psi)$ and $\mathcal{Q}^\perp = \mathcal{N}(\Psi^H)$. The range of a projector has the property that all of its elements stay invariant on application by the projector. Let \mathbf{Q}^\perp denote the basis of \mathcal{Q}^\perp .

$$\begin{aligned} \Psi \mathbf{Q} &= \mathbf{Q}(\mathbf{P}^H \mathbf{Q})^{-1} \mathbf{P}^H \mathbf{Q} = \mathbf{Q} \\ \Psi^H \mathbf{Q}^\perp &= \mathbf{P}(\mathbf{Q}^H \mathbf{P})^{-1} \mathbf{Q}^H \mathbf{Q}^\perp = \mathbf{0} \end{aligned}$$

Since $\mathcal{Q} + \mathcal{Q}^\perp = \mathbb{R}^n$, $\mathcal{Q} = \mathcal{R}(\Psi)$ and $\mathcal{Q}^\perp = \mathcal{N}(\Psi^H)$.

- Left to be shown is that $\mathcal{P}^\perp = \mathcal{N}(\Psi)$ and $\mathcal{P} = \mathcal{R}(\Psi^H)$. Let \mathbf{P}^\perp be the basis of \mathcal{P}^\perp .

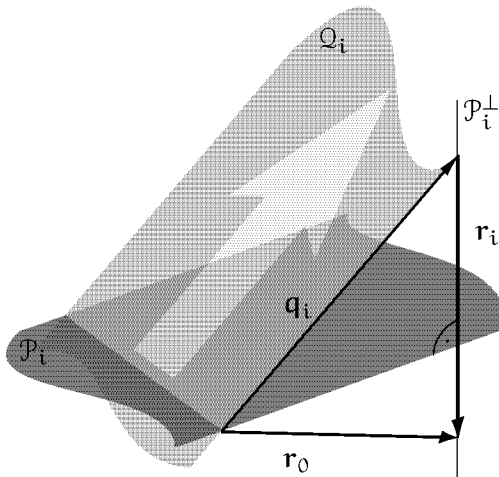
$$\begin{aligned} \Psi \mathbf{P}^\perp &= \mathbf{Q}(\mathbf{P}^H \mathbf{Q})^{-1} \mathbf{P}^H \mathbf{P}^\perp = \mathbf{0} \\ \Psi^H \mathbf{P} &= \mathbf{P}(\mathbf{Q}^H \mathbf{P})^{-1} \mathbf{Q}^H \mathbf{P} = \mathbf{P} \end{aligned}$$

Since $\mathcal{P}^\perp + \mathcal{P} = \mathbb{R}^n$, $\mathcal{P}^\perp = \mathcal{N}(\Psi)$ and $\mathcal{P} = \mathcal{R}(\Psi^H)$.

Therefore, $\mathcal{N}(\Psi) = \mathcal{P}^\perp$ and $\mathcal{R}(\Psi) = \mathcal{Q}$ as is required for Ψ to be the proclaimed projector. \square

Like $\Psi_i := Q_i(P_i^H Q_i)^{-1} P_i^H$ performs an *oblique projection* of r_0 onto Q_i along \mathcal{P}_i^\perp , $1 - \Psi_i$ performs an oblique projection of r_0 onto \mathcal{P}_i^\perp along of Q_i . The effect of $1 - \Psi_i$ on r_0 is shown in Figure 2.2.

Figure 2.2 Oblique projection: In the i^{th} step r_i is obtained by an oblique projection of r_0 onto \mathcal{P}_i^\perp by the direction of Q_i .



With the definition of Ψ_i the following computations are performed in every step i of a projection method

$$\begin{aligned} \mathbf{q}_i &= \Psi_i \mathbf{r}_0 \\ \mathbf{r}_i &= (1 - \Psi_i) \mathbf{r}_0. \end{aligned}$$

Lemma 2.7 Given step i of a projection method with correction space \mathcal{C}_i , projection space \mathcal{P}_i together with their bases \mathbf{C}_i and \mathbf{P}_i respectively solving the linear system (2.1). Let us define $\Pi_{\mathcal{P}_i} := \mathbf{P}_i \mathbf{P}_i^+$ to be the orthogonal projector onto \mathcal{P}_i .

Solving (2.7) for $\mathbf{c}_i \in \mathcal{C}_i$ is equivalent to solving

$$\Pi_{\mathcal{P}_i} \mathbf{r}_i = 0. \quad (2.9)$$

Proof. If we want to solve (2.9), we have to find a $\mathbf{z}_i \in \mathbb{R}^i$ such that

$$\Pi_{\mathcal{P}_i} (\mathbf{b} - \mathbf{A}\mathbf{C}_i \mathbf{z}_i) = 0$$

That is: We solve (2.1) in \mathcal{P}_i and therefore we have to solve

$$\mathbf{P}_i^+ \mathbf{A}\mathbf{C}_i \mathbf{z}_i = \mathbf{P}_i^+ \mathbf{b},$$

the solution of which is $\mathbf{z}_i = (\mathbf{P}_i^+ \mathbf{A}\mathbf{C}_i)^{-1} \mathbf{P}_i^+ \mathbf{b}$. Note that $\mathbf{P}_i^+ \mathbf{A}\mathbf{C}_i$ is invertible because of condition (2.8).

With this solution the residual vector becomes

$$\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{C}_i \mathbf{z}_i = \underbrace{(\mathbf{1} - \mathbf{A}\mathbf{C}_i (\mathbf{P}_i^+ \mathbf{A}\mathbf{C}_i)^{-1} \mathbf{P}_i^+)}_{=: \Phi} \mathbf{b}.$$

Φ is an oblique projector with $\mathcal{R}(\Phi) = \mathcal{A}\mathcal{C}_i$ and $\mathcal{N}(\Phi) = \mathcal{P}_i^\perp$. Then $\mathbf{r}_i = \Phi \mathbf{b}$ implies $\mathbf{P}_i^\top \mathbf{r}_i = 0$.

On the other hand let $\mathbf{P}_i = \mathbf{U}_1 \hat{\Sigma} \mathbf{V}_1^\top$ be the economical version of the singular value decomposition of \mathbf{P}_i (Definition 1.9). With this decomposition we have $\Pi_{\mathcal{P}_i} = \mathbf{P}_i \mathbf{P}_i^+ = \mathbf{U}_1 \mathbf{U}_1^\mathbf{H}$. \mathbf{U}_1 is an orthonormal basis of \mathcal{P}_i and therefore, if $\mathbf{P}_i^\mathbf{H} \mathbf{r}_i = 0$, (2.9) must hold as well. \square

WHAT IF

$\mathcal{P}_i^\perp \cap \mathcal{Q}_i \neq \{0\}$?

Remark. It is easy in practice to make sure that both subspaces \mathcal{Q}_i and \mathcal{P}_i have the same dimension. However, it might be difficult to guarantee that $\mathbf{P}_i^\mathbf{H} \mathbf{Q}_i$ is not singular. Such a situation is referred to as a *Galerkin breakdown*.

Assume that $i := \dim \mathcal{Q}_i = \dim \mathcal{P}_i$ and $\mathbf{P}_i^\mathbf{H} \mathbf{Q}_i$ is singular. Then Ψ_i does not exist. However, we can use the Moore-Penrose inverse to define

$$\hat{\Psi}_i = \mathbf{Q}_i (\mathbf{P}_i^\mathbf{H} \mathbf{Q}_i)^+ \mathbf{P}_i^\mathbf{H}.$$

$\hat{\Psi}_i$ is still a projector, which can be shown by the help of one of the Moore Penrose equations:

$$\hat{\Psi}_i^2 = \mathbf{Q}_i(\mathbf{P}_i^H \mathbf{Q}_i)^+(\mathbf{P}_i^H \mathbf{Q}_i)(\mathbf{P}_i^H \mathbf{Q}_i)^+ \mathbf{P}_i^H = \mathbf{Q}_i(\mathbf{P}_i^H \mathbf{Q}_i)^+ \mathbf{P}_i^H = \hat{\Psi}_i.$$

Let us consider the singular value decomposition (Definition 1.9) of $\mathbf{P}^H \mathbf{Q}_i$:

$$\mathbf{P}^H \mathbf{Q}_i = \mathbf{U} \Sigma \mathbf{V}^H = [\mathbf{U}_1 \quad \mathbf{U}_2] \begin{bmatrix} \hat{\Sigma} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{bmatrix} = \mathbf{U}_1 \hat{\Sigma} \mathbf{V}_1^H.$$

With the SVD we can rewrite $\hat{\Psi}_i$ as

$$\hat{\Psi}_i = \mathbf{Q}_i \mathbf{V} \Sigma^+ \mathbf{U}^H \mathbf{P}^H = \mathbf{Q}_i \mathbf{V}_1 \hat{\Sigma}^{-1} \mathbf{U}_1^H \mathbf{P}^H$$

Using this result we find with a similar argument to the one used in the proof of Lemma 2.6 that

$$\begin{aligned} \text{span}\{\mathbf{P}_i \mathbf{U}_1\}^\perp &= \mathcal{N}(\hat{\Psi}_i) & \text{and} & & \text{span}\{\mathbf{P}_i \mathbf{U}_1\} &= \mathcal{R}(\hat{\Psi}_i^H) \\ \text{span}\{\mathbf{Q}_i \mathbf{V}_1\} &= \mathcal{R}(\hat{\Psi}_i) & \text{and} & & \text{span}\{\mathbf{Q}_i \mathbf{V}_1\}^\perp &= \mathcal{N}(\hat{\Psi}_i^H) \end{aligned}$$

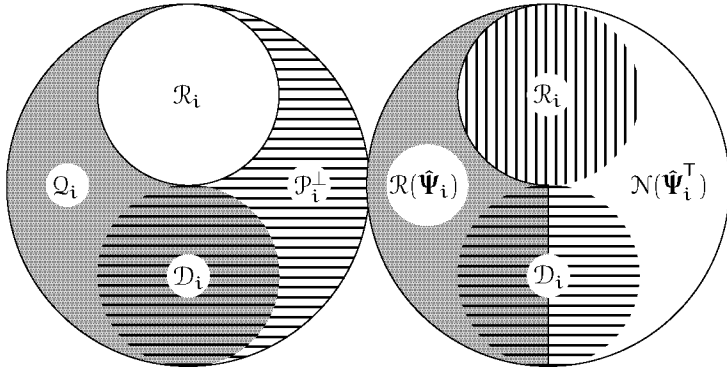
Moreover $\mathcal{D}_i := \mathcal{P}_i^\perp \cap \mathcal{Q}_i = \left\{ \mathbf{x} \in \mathcal{Q}_i \mid \mathbf{P}^H \mathbf{x} = 0 \right\}$. Using the ansatz $\mathbf{x} = \mathbf{Q}_i \mathbf{a}$ for a vector $\mathbf{x} \in \mathbb{R}^n$ we see that \mathbf{x} is in $\mathcal{N}(\mathbf{P}^H \mathbf{Q}_i)$. Then $\mathcal{D}_i := \mathbf{Q}_i \mathbf{V}_2$ must be a basis of \mathcal{D}_i . \mathcal{D}_i is a subspace of $\mathcal{N}(\hat{\Psi}_i)$ since

$$(\mathbf{P}_i \mathbf{U}_1)^H \mathcal{D}_i = \mathbf{U}_1^H \mathbf{P}^H \mathbf{Q}_i \mathbf{V}_2 = \mathbf{U}_1^H \mathbf{U}_1 \hat{\Sigma} \mathbf{V}_1^H \mathbf{V}_2 = 0.$$

If we apply the same argument to $\mathcal{R}_i := \mathcal{Q}_i^\perp \cap \mathcal{P}_i$ we find that its basis must be $\mathcal{R}_i := \mathbf{P}_i \mathbf{U}_2$ and $\mathcal{R}_i \in \mathcal{N}(\hat{\Psi}_i^H)$. However, nothing can be said in general about \mathcal{D}_i in connection with $\hat{\Psi}_i^H$ and \mathcal{R}_i in connection with $\hat{\Psi}_i$ respectively.

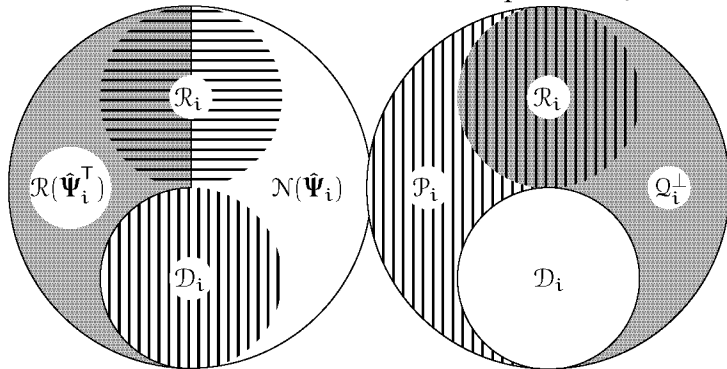
WHAT ABOUT
 \mathcal{D}_i AND \mathcal{R}_i ?

Figure 2.3 Four symbolic views of \mathbb{R}^i showing spaces involved in a Galerkin breakdown of a projection method.



\mathcal{D}_i is the intersection of \mathcal{Q}_i and \mathcal{P}_i^\perp , \mathcal{R}_i is \mathbb{R}^i without the space spanned by \mathcal{Q}_i and \mathcal{P}_i^\perp together.

\mathcal{R}_i is a subspace of $\mathcal{N}(\hat{\Psi}_i^H)$, \mathcal{D}_i is shared by $\mathcal{N}(\hat{\Psi}_i^H)$ and $\mathcal{R}(\hat{\Psi}_i)$. $\mathcal{R}(\hat{\Psi}_i)$ is a subspace of \mathcal{Q}_i .



\mathcal{D}_i is a subspace of $\mathcal{N}(\hat{\Psi}_i)$, \mathcal{R}_i is shared by $\mathcal{N}(\hat{\Psi}_i)$ and $\mathcal{R}(\hat{\Psi}_i^H)$. $\mathcal{R}(\hat{\Psi}_i^H)$ is a subspace of \mathcal{P}_i .

\mathcal{R}_i is the intersection of \mathcal{P}_i and \mathcal{Q}_i^\perp , \mathcal{D}_i is \mathbb{R}^i without the space spanned by \mathcal{P}_i and \mathcal{Q}_i^\perp together.

Due to the facts that $(\mathbf{Q}_i^\perp)^\mathbf{H} \mathbf{Q}_i \mathbf{V}_1 = 0$ and that $\mathbf{Q}_i \mathbf{V}_1$ is the basis of $\mathcal{R}(\hat{\Psi}_i)$ we can conclude that the range of $\hat{\Psi}_i$ is a subspace of \mathcal{Q}_i . An analogous consideration leads to the result that the range of $\hat{\Psi}_i^\mathbf{H}$ is a subspace of \mathcal{P}_i .

WHAT ABOUT
 $\mathcal{R}(\hat{\Psi}_i)$ AND
 $\mathcal{R}(\hat{\Psi}_i^\mathbf{H})$?

Figure 2.3 shows the relations between the spaces involved in a Galerkin breakdown of a projection method.

Example 2.8 Let us consider the following configuration of the bases $\mathbf{P}_i, \mathbf{Q}_i, \mathbf{P}_i^\perp, \mathbf{Q}_i^\perp$ of the subspaces $\mathcal{P}_i, \mathcal{Q}_i$ and their complements $\mathcal{P}_i^\perp, \mathcal{Q}_i^\perp$ in \mathbb{R}^4 :

$$\mathbf{P}_i := \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 1 & -1 \end{bmatrix} \quad \mathbf{P}_i^\perp := \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}_i := \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{Q}_i^\perp := \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

The singular value decomposition of $\mathbf{P}^\mathbf{H} \mathbf{Q}_i$ is

$$\Sigma = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{U}_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \mathbf{V}_1 = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$\mathbf{U}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \mathbf{V}_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

With this decomposition we can calculate bases for \mathcal{D}_i , and \mathcal{R}_i

$$\mathbf{D}_i = \mathbf{Q}_i \mathbf{V}_2 = -\frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{R}_i = \mathbf{P}_i \mathbf{U}_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \\ -1 \end{bmatrix}.$$

\mathbf{D}_i can be obtained by subtracting the second column vector of \mathbf{Q}_i from the first one. It is equal to the first column vector of \mathbf{P}_i^\perp . Consequently \mathcal{D}_i is actually the intersection of \mathcal{Q}_i and \mathcal{P}_i^\perp .

\mathcal{R}_i can be obtained by adding the first column vector of \mathbf{Q}_i^\perp two times to the first one. It is equal to the first column vector of \mathbf{P}_i and therefore \mathcal{R}_i is the intersection of the two spaces \mathcal{Q}_i^\perp and \mathcal{P}_i .

The bases for $\mathcal{R}(\hat{\Psi}_i)$ and $\mathcal{R}(\hat{\Psi}_i^H)$ become

$$\text{span}\{\mathcal{R}(\hat{\Psi}_i)\} = \mathbf{Q}_i \mathbf{V}_1 = -\frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix},$$

$$\text{span}\{\mathcal{R}(\hat{\Psi}_i^H)\} = \mathbf{P}_i \mathbf{U}_1 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}.$$

The basis of $\mathcal{R}(\hat{\Psi}_i)$ can be obtained by adding the second column vector of \mathbf{Q}_i two times to the first one. This shows that $\mathcal{R}(\hat{\Psi}_i)$ is actually a subspace of \mathcal{Q}_i as stated before. Likewise the basis of $\mathcal{R}(\hat{\Psi}_i^H)$ is the negative second column vector of \mathbf{P}_i and therefore $\mathcal{R}(\hat{\Psi}_i^H)$ is a subspace of \mathcal{P}_i .

Bases for $\mathcal{N}(\hat{\Psi}_i)$ and $\mathcal{N}(\hat{\Psi}_i^H)$ would be

$$\text{span}\{\mathcal{N}(\hat{\Psi}_i)\} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\text{span}\{\mathcal{N}(\hat{\Psi}_i^H)\} = \begin{bmatrix} 0 & -1 & -2 \\ 0 & 0 & 2 \\ 0 & 1 & -2 \\ -1 & 0 & 0 \end{bmatrix}.$$

\mathbf{D}_i is the first column vector of the basis of $\mathcal{N}(\hat{\Psi}_i)$, which shows that it is a subspace of the kernel of $\hat{\Psi}_i$. \mathcal{R}_i is constructed by adding the first and second column vector of the basis of $\mathcal{N}(\hat{\Psi}_i^H)$. Therefore, \mathcal{R}_i is a subspace of the kernel of $\hat{\Psi}_i^H$. \square

Let us consider step n of a projection method. In this step the correction space and the projection space are of full dimension n . Their bases are invertible. We can therefore simplify Ψ_n as

$$\Psi_n = Q_n(\mathbf{P}_n^H Q_n)^{-1} \mathbf{P}_n^H = Q_n Q_n^{-1} \mathbf{P}_n^{-H} \mathbf{P}_n^H = \mathbf{1}.$$

Consequently $\mathbf{r}_n = (1 - \Psi_n)\mathbf{r}_0 = \mathbf{0}$ after n steps. This is a very important observation. We have proven the following theorem.

Theorem 2.9 *Supposed, there are no rounding errors involved: If the dimension of the correction space is consecutively increased by 1 in each step, projection methods will converge after at most n steps.* \square

Let the correction space \mathcal{C}_i be given together with its basis \mathbf{C}_i . We have already shown that a fully working projection method must assure that the projection space \mathcal{P}_i together with its basis \mathbf{P}_i is chosen in a way that it is

- of the same dimension as \mathcal{C}_i and
- $\mathbf{P}_i^H \mathbf{Q}_i$ is not singular.

The following lemma from Saad [32, p. 125] shows that it is sufficient to choose \mathbf{P}_i either as a basis of \mathcal{C}_i or \mathcal{Q}_i (e.g. \mathbf{C}_i resp. \mathbf{Q}_i) to solve the linear system (2.1):

Lemma 2.10 *Let \mathbf{A} , \mathcal{C} and \mathcal{P} satisfy either one of the following two conditions:*

1. \mathbf{A} is hermitian positive definite and $\mathcal{P} = \mathcal{C}$.
2. \mathbf{A} is nonsingular and $\mathcal{P} = \mathcal{Q}$.

Then the matrix $\mathbf{M} := \mathbf{P}_i^H \mathbf{Q}_i$ is nonsingular for any bases \mathbf{P} and \mathbf{C} of \mathcal{P} and \mathcal{C} , respectively.

CONVERGES
WITHIN n
STEPS!

HOW TO
CHOOSE THE
PROJECTION
SPACE?

Proof.

1. Since $\mathcal{P} = \mathcal{C}$ there exists a nonsingular matrix \mathbf{G} such that $\mathbf{P} = \mathbf{C}\mathbf{G}$. Therefore, we can write $\mathbf{M} = \mathbf{G}^H \mathbf{C}^H \mathbf{A} \mathbf{C}$. Since \mathbf{A} is positive definite, so must be $\mathbf{C}^H \mathbf{A} \mathbf{C}$. With \mathbf{G} being nonsingular, so must be \mathbf{M} .
2. Since $\mathcal{P} = \mathcal{Q}$ there exists a nonsingular matrix \mathbf{G} such that $\mathbf{P} = \mathbf{Q}\mathbf{G}$. Hence we can write \mathbf{M} as $\mathbf{M} = \mathbf{G}^H (\mathbf{C}\mathbf{A})^H \mathbf{A} \mathbf{C}$. Since \mathbf{A} is nonsingular, $(\mathbf{C}\mathbf{A})^H \mathbf{A} \mathbf{C}$ is of full rank and even positive definite. With \mathbf{G} being nonsingular, so must be \mathbf{M} .

□

With the last lemma we have only two cases of projection methods left to analyze: One for positive definite matrices and a more general method for nonsingular matrices.

Orthogonal Residual Methods

Definition 2.11 A projection method with $\mathcal{P}_i = \mathcal{C}_i$ is called orthogonal residual method (or OR method).

The algorithm for an OR method is a slight simplification of Algorithm 2.2. Because \mathcal{P}_i is identical to \mathcal{C}_i , line number 9 can be dropped and every occurrence of \mathbf{P}_i can be replaced by \mathbf{C}_i . The whole algorithm for an OR method is shown as Algorithm 2.3.

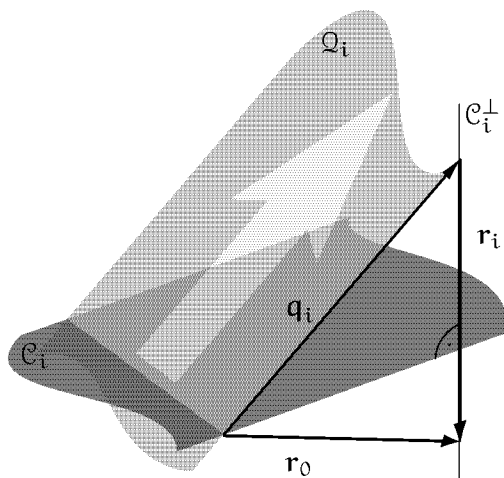
WHAT IS
MINIMIZED?

If the system matrix of the linear system \mathbf{A} is hermitian positive definite, an OR method is also a minimization method.

Lemma 2.12 Let \mathbf{A} be hermitian and positive definite.

An OR method minimizes the \mathbf{A} -norm of the error vector $\boldsymbol{\eta}_i$ over the correction space \mathcal{C}_i in each step i .

Figure 2.4 Orthogonal Residual (OR) method: This is a projection method with $\mathcal{P}_i = \mathcal{C}_i$: \mathbf{r}_0 is projected unto \mathcal{C}_i^\perp along \mathcal{Q}_i .



Proof. The error vector is given by $\boldsymbol{\eta}_i = \mathbf{s} - \mathbf{s}_i$. Since $\mathbf{s}_i = \mathbf{s}_0 + \mathbf{c}_i$ with $\mathbf{c}_i = \mathbf{C}_i \mathbf{z}_i$ in an OR method, the error vector can be expressed as

$$\boldsymbol{\eta}_i = \mathbf{s} - \mathbf{s}_0 - \mathbf{C}_i \mathbf{z}_i = \boldsymbol{\eta}_0 - \mathbf{C}_i \mathbf{z}_i$$

The \mathbf{A} -norm of the error vector is

$$\|\boldsymbol{\eta}_i\|_{\mathbf{A}}^2 = \boldsymbol{\eta}_i^{\mathbf{H}} \mathbf{A} \boldsymbol{\eta}_i = \boldsymbol{\eta}_0^{\mathbf{H}} \mathbf{A} \boldsymbol{\eta}_0 - 2\boldsymbol{\eta}_0^{\mathbf{H}} \mathbf{A} \mathbf{C}_i \mathbf{z}_i + \mathbf{z}_i^{\mathbf{H}} \mathbf{C}_i^{\mathbf{H}} \mathbf{A} \mathbf{C}_i \mathbf{z}_i$$

Minimizing this norm for \mathbf{z}_i results in

$$\mathbf{z}_i = (\mathbf{C}_i^{\mathbf{H}} \mathbf{A} \mathbf{C}_i)^{-1} \mathbf{C}_i^{\mathbf{H}} \mathbf{A} \boldsymbol{\eta}_0 = (\mathbf{C}_i^{\mathbf{H}} \mathbf{A} \mathbf{C}_i)^{-1} \mathbf{C}_i^{\mathbf{H}} \mathbf{r}_0.$$

This is the component vector that an OR method computes in every step. \square

Algorithm 2.3 Template algorithm for an Orthogonal Residual Method.

```

function ORM(A, b, x0, ε): vector;
  i:= 0;
  n:= dimension(A);
  r0:= b - Ax0;
  while ||ri|| > ε||b|| do
    choose Ci+1;
    Ci+1:= basis of Ci+1;
    Qi+1:= ACi+1;

    zi+1:= (Ci+1HQi+1)-1Ci+1Hr0;
    qi+1:= Qi+1zi+1;

    ri+1:= r0 - qi+1;
    i:= i + 1;
  end;
  ci:= Cizi;
  si:= x0 + ci;
  ORM:= si;
end ORM;

```

HOW TO
CHOOSE THE
CORRECTION
SPACE?

Let A be hermitian and positive definite. Lemma 2.12 tells us, that in every step of an OR-method the A -norm of the error vector is minimized. That is the A -norm of $\eta_i \in \{\eta_0\} + C_i$ is minimized in step i , whereas in step $i + 1$ the A -norm of $\eta_{i+1} \in \{\eta_0\} + C_{i+1}$ is minimized. A monotonous descending sequence of error (A -)norms can be obtained by claiming that C_i should be contained in C_{i+1} for all $i \geq 1$.

Lemma 2.13 *Given a linear system with hermitian positive definite system matrix A . An OR method converges monotonously, i.e. $\|\eta_{i+1}\|_A \leq \|\eta_i\|_A$ if the following rule holds*

$$C_i \subset C_{i+1} \quad (2.10)$$

Proof. Let us consider the difference of two subsequent error vectors

$$\mathbf{a} := \boldsymbol{\eta}_i - \boldsymbol{\eta}_{i+1} = \mathbf{c}_{i+1} - \mathbf{c}_i.$$

Equation (2.10) tells us that $\mathbf{a} \in \mathcal{C}_{i+1}$. Consequently the difference of two subsequent residual vectors

$$\mathbf{r}_i - \mathbf{r}_{i+1} = \mathbf{q}_{i+1} - \mathbf{q}_i = \mathbf{A}\mathbf{a}$$

is an element of \mathcal{Q}_{i+1} . An OR method determines \mathbf{r}_{i+1} such that it is an element of \mathcal{C}_{i+1}^\perp . Since \mathbf{A} is hermitian $\boldsymbol{\eta}_{i+1}$ is thus an element of \mathcal{Q}_{i+1}^\perp . Hence we have

$$\begin{aligned} \|\boldsymbol{\eta}_i\|_{\mathbf{A}}^2 &= \boldsymbol{\eta}_i^H \mathbf{r}_i = (\mathbf{a} + \boldsymbol{\eta}_{i+1})^H (\mathbf{A}\mathbf{a} + \mathbf{r}_{i+1}) \\ &= \mathbf{a}^H \mathbf{A}\mathbf{a} + \underbrace{\boldsymbol{\eta}_{i+1}^H \mathbf{A}\mathbf{a}}_{=0} + \underbrace{\mathbf{a}^H \mathbf{r}_{i+1}}_{=0} + \boldsymbol{\eta}_{i+1}^H \mathbf{r}_{i+1} \\ &= \|\mathbf{a}\|_{\mathbf{A}}^2 + \|\boldsymbol{\eta}_{i+1}\|_{\mathbf{A}}^2. \end{aligned}$$

Since $\|\mathbf{a}\|_{\mathbf{A}}^2 \geq 0$, $\|\boldsymbol{\eta}_{i+1}\|_{\mathbf{A}} \leq \|\boldsymbol{\eta}_i\|_{\mathbf{A}}$ as proclaimed. \square

Let us now define the scalar functional η :

$$\eta : \mathbb{R}^n \rightarrow \mathbb{R}, \mathbf{x} \mapsto \|\eta(\mathbf{x})\|_{\mathbf{A}}^2 = \mathbf{s}^H \mathbf{A}\mathbf{s} - \mathbf{x}^H \mathbf{A}\mathbf{s} - \mathbf{s}^H \mathbf{A}\mathbf{x} + \mathbf{x}^H \mathbf{A}\mathbf{x}.$$

The functional is chosen such that for hermitian \mathbf{A} $\eta(\mathbf{x}_i)$ is the square of the \mathbf{A} -norm of the error vector $\boldsymbol{\eta}_i$. If \mathbf{A} is split into its hermitian part \mathbf{S} and its skew hermitian part \mathbf{T} , the gradient of this functional becomes

$$\begin{aligned} \nabla \eta(\mathbf{x}) &= \nabla (\mathbf{s}^H \mathbf{A}\mathbf{s} - 2\operatorname{Re}(\mathbf{s}^H \mathbf{S}\mathbf{x}) - 2\operatorname{Im}(\mathbf{s}^H \mathbf{T}\mathbf{x}) + \mathbf{x}^H \mathbf{A}\mathbf{x}) \\ &= -2\operatorname{Re}(\mathbf{s}^H \mathbf{S}) - 2\operatorname{Im}(\mathbf{s}^H \mathbf{T}) + 2\mathbf{x}^H \mathbf{S}. \end{aligned}$$

In the hermitian case the gradient becomes $-2\mathbf{r}^H$. The local optimum direction for the change of \mathbf{x}_i is therefore \mathbf{r}_i . Hence the correction space should be chosen as follows

$$\begin{aligned}\mathcal{C}_1 &= \mathbf{r}_0 \\ \mathcal{C}_{i+1} &= \text{span}\{\mathcal{C}_i, \mathbf{r}_i\} \\ &= \text{span}\{\mathcal{C}_i, \mathbf{r}_0 + \mathbf{A}\mathcal{C}_i\mathbf{z}_i\} = \text{span}\left\{\mathcal{C}_i, \mathbf{A}^i\mathbf{r}_0\right\}.\end{aligned}$$

This scheme suggests choosing \mathcal{C}_i as a Krylov space with matrix \mathbf{A} and vector \mathbf{r}_0 . Optimization of an OR method on a linear system with a hermitian positive definite system matrix using a Krylov space leads to the well known *Conjugate Gradient (CG) method* (see for example [32, pp. 176]).

However, little can be said about the nonhermitian positive definite cases.

Minimum Residual Methods

Definition 2.14 A minimum residual (MR) method is a projection method with $\mathcal{P}_i = \mathcal{Q}_i$.

A general algorithm for an MR method is shown in algorithm 2.4.

WHAT IS
MINIMIZED?

Since \mathcal{Q}_i is the projection space of a minimum residual method, the component vector \mathbf{z}_i is obtained by

$$\mathbf{z}_i = \left(\mathbf{Q}_i^H \mathbf{Q}_i\right)^{-1} \mathbf{Q}_i^H \mathbf{r}_0$$

This is the solution of the normal equations of the minimization problem $\|\mathbf{r}_i\| = \min_{\mathbf{z} \in \mathbb{R}^i} \|\mathbf{r}_0 - \mathbf{Q}_i\mathbf{z}\|$. Hence in step i of an MR method the 2-norm of the residual is minimized over the projection space \mathcal{Q}_i .

Algorithm 2.4 Template algorithm for a Minimum Residual Method.

function MRM(A, b, x_0, ε): **vector**;

$i := 0$;

$n := \dim(A)$;

$r_0 := b - Ax_0$;

while $\|r_i\| > \varepsilon\|b\|$ **do**

 choose \mathcal{C}_{i+1} ;

$C_{i+1} := \text{basis of } \mathcal{C}_{i+1}$;

$Q_{i+1} := AC_{i+1}$;

$z_{i+1} := (Q_{i+1}^H Q_{i+1})^{-1} Q_{i+1}^H r_0$;

$q_{i+1} := Q_{i+1} z_{i+1}$;

$r_{i+1} := r_0 - q_{i+1}$;

$i := i + 1$;

end;

$c_i := C_i z_i$;

$s_i := x_0 + c_i$;

 MRM := s_i ;

end MRM;

Lemma 2.15 *An MR-method converges monotonously,* MONOTONOUS CONVERGENCE
i.e. $\|r_{i+1}\| \leq \|r_i\|$, if

$$\mathcal{Q}_i \subset \mathcal{Q}_{i+1}.$$

Proof. We have $r_{i+1} = r_0 - q_{i+1}$ and $r_i = r_0 - q_i$ with $q_{i+1} \in \mathcal{Q}_{i+1}$ and $q_i \in \mathcal{Q}_i$.

Let us consider $\alpha := r_{i+1} - r_i = q_i - q_{i+1}$.

- Since $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$, $\alpha \in \mathcal{Q}_{i+1}$.
- The MR-method determines r_{i+1} , such that $r_{i+1} \in \mathcal{Q}_{i+1}^\perp$.

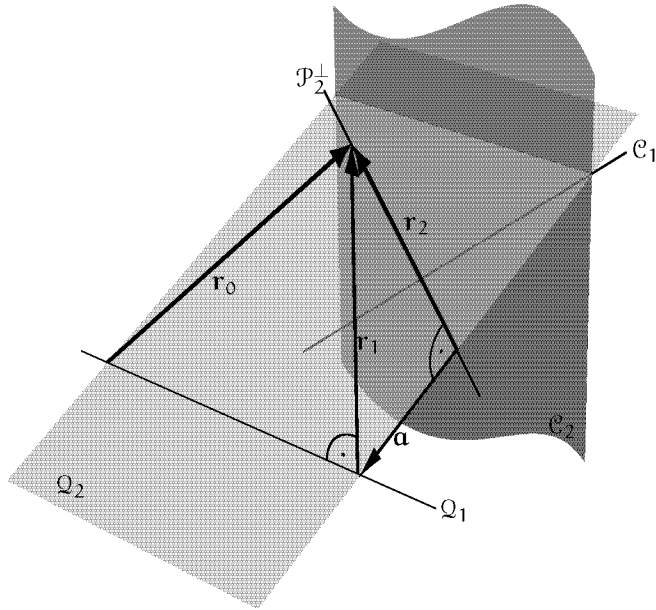
Using these two facts we conclude that \mathbf{a} is orthogonal to \mathbf{r}_{i+1} . Thus we get the proclaimed statement

$$\|\mathbf{r}_i\|^2 = \|\mathbf{r}_{i+1}\|^2 + \|\mathbf{a}\|^2 \geq \|\mathbf{r}_{i+1}\|^2.$$

□

In Figure 2.5 the statement of Lemma 2.15 is depicted for the first two steps of an MR-method.

Figure 2.5 Minimum residual (MR) method: This is a projection method with $\mathcal{P}_i = \mathcal{Q}_i$. Since \mathbf{r}_0 is the hypotenuse of a right angled triangle spanned by \mathbf{r}_0 and \mathbf{r}_1 , $\|\mathbf{r}_1\|$ is less or equal to $\|\mathbf{r}_0\|$. The same thing applies to \mathbf{r}_1 and \mathbf{r}_2 , and more generally to \mathbf{r}_i and \mathbf{r}_{i+1} .



Let us consider the scalar functional r :

$$r : \mathbb{R}^n \rightarrow \mathbb{R} :$$

$$x \mapsto \|r(\mathbf{A}, \mathbf{b}, x)\|_2^2 = \mathbf{b}^H \mathbf{b} - 2\text{Re}(\mathbf{b}^H \mathbf{A}x) + x^H \mathbf{A}^H \mathbf{A}x$$

HOW TO
CHOOSE THE
CORRECTION
SPACE?

The functional is chosen such that $r(\mathbf{s}_i)$ is the square of the 2-norm of the residual vector \mathbf{r}_i . The gradient of this functional is

$$\nabla r(x) = -2(\mathbf{b}^H - x^H \mathbf{A}^H) \mathbf{A} = -2r(\mathbf{A}, \mathbf{b}, x)^H \mathbf{A}$$

This equation tells us that the local optimum direction for the change of \mathbf{s}_i is $\mathbf{A}^H \mathbf{r}_i$. According to this the correction space should be chosen as follows:

$$\mathcal{C}_1 = \mathbf{A}^H \mathbf{r}_0 \quad (2.11)$$

$$\begin{aligned} \mathcal{C}_{i+1} &= \text{span} \left\{ \mathcal{C}_i, \mathbf{A}^H \mathbf{r}_i \right\} \\ &= \text{span} \left\{ \mathcal{C}_i, (\mathbf{A}^H \mathbf{A})^i \mathbf{A}^H \mathbf{r}_0 \right\} \end{aligned} \quad (2.12)$$

This scheme suggests choosing \mathcal{C}_i as a Krylov space with matrix $\mathbf{A}^H \mathbf{A}$ and vector $\mathbf{A}^H \mathbf{r}_0$.

Let us consider the normal equations of (2.1):

$$\hat{\mathbf{A}} \mathbf{s} = \hat{\mathbf{b}} \quad \text{with} \quad \hat{\mathbf{A}} := \mathbf{A}^H \mathbf{A} \quad \text{and} \quad \hat{\mathbf{b}} := \mathbf{A}^H \mathbf{b}$$

The residual vector of this system is

$$\hat{\mathbf{r}}_i = \mathbf{A}^H (\mathbf{b} - \mathbf{A} \mathbf{s}_i) = \mathbf{A}^H \mathbf{r}_i.$$

Applying an OR method to the normal equations leads to the following expression for finding the component vector \mathbf{z}_i :

$$\mathbf{z}_i = (\mathbf{C}_i^H \hat{\mathbf{A}} \mathbf{C}_i)^{-1} \mathbf{C}_i^H \hat{\mathbf{r}}_i = (\mathbf{Q}_i^H \mathbf{Q}_i)^{-1} \mathbf{Q}_i^H \mathbf{r}_i \quad (2.13)$$

The suggested correction space for this OR method is a Krylov space with matrix $\hat{\mathbf{A}}$ and $\hat{\mathbf{r}}_i$. This is the same

correction space we got for the MR method in equations (2.11) and (2.12). Furthermore equation (2.13) tells us that an MR method using such a correction space is in fact an OR method on the normal equations of the given linear system. Optimization on OR methods on the normal equations of the linear system using Krylov spaces leads to algorithms known as CGNR and CGNE [32, pp. 237].

It is well known that normal equations may cause numerical problems, because the condition of the system matrix is squared. However, we will see in the next section that using such a scheme can also have a bad algebraic impact on the convergence of a method and that a better *and* simpler choice for the correction space exists.

2.4 Krylov Space Methods

2.4.1 Motivation

Using the ansatz (2.4) of correction space methods for the residual in step i , we transform the linear system (2.1) into the equivalent system

$$\mathbf{A} \mathbf{c}_\infty = \mathbf{r}_0. \quad (2.14)$$

This system has to be solved for \mathbf{c}_∞ which is equivalent to the initial error vector $\boldsymbol{\eta}_0$. To solve this system \mathbf{A}^{-1} has to be approximated. This can be achieved by using the minimal polynomial $\mu_{\mathbf{A}}$ of the system matrix \mathbf{A} . Let

$$\mu_{\mathbf{A}}(x) := \sum_{i=0}^d \alpha_i x^i$$

be the minimal polynomial of \mathbf{A} . Then we have

$$\mu_{\lambda}(\mathbf{A}) = 1\alpha_0 + \mathbf{A} \sum_{i=0}^{d-1} \alpha_{i+1} \mathbf{A}^i = 0.$$

And consequently the following relation holds:

$$\mathbf{1} = \mathbf{A} \left(-\frac{1}{\alpha_0} \sum_{i=0}^{d-1} \alpha_{i+1} \mathbf{A}^i \right),$$

This implies

$$\mathbf{A}^{-1} = -\frac{1}{\alpha_0} \sum_{i=0}^{d-1} \alpha_{i+1} \mathbf{A}^i.$$

Thus the solution of (2.14),

$$\mathbf{c}_{\infty} = \mathbf{c}_{d-1} = \boldsymbol{\eta}_0 = -\frac{1}{\alpha_0} \sum_{i=0}^{d-1} \alpha_{i+1} \mathbf{A}^i \mathbf{r}_0, \quad (2.15)$$

is a linear combination of vectors $\mathbf{A}^i \mathbf{r}_0$, $0 \leq i \leq d-1$ and is therefore an element of the Krylov space $\mathcal{K}_{d-1}(\mathbf{A}, \mathbf{r}_0)$. With these observations we have shown already the following theorem taken from [20].

Theorem 2.16 *If the minimal polynomial of the nonsingular matrix \mathbf{A} has degree m , then the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ lies in the space $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$.* \square

Example 2.17 Consider the linear system

$$\begin{bmatrix} 3 & 1 & 2 & 1 & 1 & 1 \\ 0 & 3 & 0 & 1 & -2 & 3 \\ 0 & 0 & 2 & 0 & 0 & -4 \\ 0 & 0 & 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \quad (2.16)$$

Its matrix is positive definite. The minimal polynomial of the matrix is:

$$\begin{aligned}\mu_{\mathbf{A}}(x) &= (x-3)^2(x-2)(x-4) \\ &= x^4 - 12x^3 + 53x^2 - 102x + 72.\end{aligned}$$

This polynomial has degree 4. A Krylov space method should therefore converge within 4 steps. The results for OR methods and MR methods with and without using a Krylov space as the correction space with initial residual $\mathbf{r}_0 = [-2, 1, 1, 1, 1]^T$ in Matlab are shown in Table 2.1.

Table 2.1 Results for Example 2.17 with initial residual $\mathbf{r}_0 = [-2, 1, 1, 1, 1]^T$ in Matlab: M1 and M2 are OR methods, M3 and M4 are MR methods. The numbers listed in the table for OR methods are $\boldsymbol{\eta}_i^H \mathbf{A} \boldsymbol{\eta}_i$, whereas the numbers for MR methods are $\mathbf{r}_i^H \mathbf{r}_i$. The correction space used by M1 and M3 is a Krylov space $\mathcal{K}_i(\mathbf{A}, \mathbf{r}_0)$, M2 uses $\{\mathbf{S}\boldsymbol{\eta}_0, \dots, \mathbf{S}\boldsymbol{\eta}_{i-1}\}$, M4 uses $\{\mathbf{A}^H \mathbf{r}_0, \dots, \mathbf{A}^H \mathbf{r}_{i-1}\}$.

i	M1	M2	M3	M4
0	4.916667	4.916667	9.000000	9.000000
1	4.254902	2.557133	6.020619	5.071106
2	0.034812	0.362755	0.114399	3.150672
3	0.001845	0.220116	0.005666	1.915598
4	0.000000	0.056908	0.000000	1.504152
5		0.167435		0.037557
6		0.000000		0.000000

In this table the component vector \mathbf{z}_i in the two Krylov space methods M1 (OR method) and M3 (MR method) converge in different ways to $\mathbf{z}_4 = 1/72[102, -53, 12, -1]^T$, finally resulting in an initial error vector $\boldsymbol{\eta}_0$,

$$\boldsymbol{\eta}_0 = \mathbf{c}_4 = \mathbf{C}\mathbf{z}_4 = \frac{1}{72} \left(102 \cdot \mathbf{1} - 53\mathbf{A} + 12\mathbf{A}^2 - \mathbf{A}^3 \right) \mathbf{r}_0$$

and a residual vector

$$\mathbf{r}_4 = \mathbf{A}\boldsymbol{\eta}_4 = \frac{1}{72} \left(72 \cdot \mathbf{1} - 102\mathbf{A} + 53\mathbf{A}^2 - 12\mathbf{A}^3 + \mathbf{A}^4 \right) \mathbf{r}_0.$$

In the last expression the second factor is the minimal polynomial of \mathbf{A} making r_4 (and hence η_4) vanish.

M4 is a Krylov space method of OR type on the normal equations of (2.16) (and thus also an MR method). The eigenvalues of their matrix $\mathbf{A}^H \mathbf{A}$ are the singular values of \mathbf{A}

$$\sigma = [26.5880, 6.0739, 3.9748, 1.9830, 1.8426, 0.2456].$$

Because all of the singular values are different, the minimal polynomial of $\mathbf{A}^H \mathbf{A}$ has maximum degree 6. Consequently M4 converges within the maximum number of steps.

Similarly the eigenvalues of the symmetric part \mathbf{S} of \mathbf{A} are

$$\lambda(\mathbf{S}) = [0.0237, 0.5434, 2.1745, 3.8714, 5.4143, 5.9727].$$

Because all of the eigenvalues are different, the minimal polynomial of \mathbf{S} has maximum degree 6. Although method M2 builds a space that includes always the local optimum search direction as the next basis vector, it also converges not before the full dimension 6 of the correction space is reached.

Summing up the statements made so far in this section we can describe a Krylov space method as follows: It is a projection method that uses a Krylov space as its correction space to approximate the initial error η_0 , which is a member of the Krylov space.

Definition 2.18 A Krylov space method is a projection method with correction space $\mathcal{C}_j = \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$.

It has been mentioned by Wilkinson [39, pp. 369] that the simple approach of building a Krylov space by computing the vectors $\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{j-1}\mathbf{r}_0$ consecutively as a basis for $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ is numerically unstable. The column vectors become more and more linearly dependent, because they converge into the direction of an eigenvector associated with the largest absolute eigenvalue. To overcome this numerical problem several schemes have been developed for Krylov space generation for general matrices, one by Arnoldi involving orthogonalization, the other one by Lanczos involving biorthogonalization.

2.4.2 Arnoldi Process

Given a matrix \mathbf{A} and a vector \mathbf{a} we want to generate a sequence of orthonormal basis vectors for Krylov spaces $\mathcal{K}_i(\mathbf{A}, \mathbf{a})$ with $i = 1, 2, \dots, n$.

To obtain orthonormal vectors, we employ a scheme that differs from the normal Gram-Schmidt procedure in that we do not have a the full set of vectors to be orthogonalized at the beginning. Instead the basis vectors are generated by multiplication by \mathbf{A} when they are needed. The first basis vector is the normalized starting vector $\mathbf{v}_1 := \mathbf{a}/\|\mathbf{a}\|$. The other vectors $\mathbf{v}_2, \dots, \mathbf{v}_n$ are generated as follows.

$$\begin{aligned} \mathbf{v}_1 &:= \mathbf{a}/\|\mathbf{a}\| \\ \mathbf{t}_{j+1} &:= \mathbf{A}\mathbf{v}_j - \sum_{i=1}^j h_{i,j}\mathbf{v}_i \quad \text{with } h_{i,j} := \mathbf{v}_i^H \mathbf{A}\mathbf{v}_j \\ \mathbf{v}_{j+1} &:= \mathbf{t}_{j+1}/\|\mathbf{t}_{j+1}\| \end{aligned} \quad (2.17)$$

This scheme is called *Arnoldi Process*. An implementation is shown in Algorithm 2.5.

$$\|\mathbf{t}_{j+1}\| = h_{j+1,j}$$

It is natural to label $\|\mathbf{t}_{j+1}\|$ as $h_{j+1,j}$, because

$$\|\mathbf{t}_{j+1}\|^2 = \mathbf{t}_{j+1}^H \mathbf{t}_{j+1} = \mathbf{t}_{j+1}^H \mathbf{A}\mathbf{v}_j - \sum_{i=1}^j h_{i,j} \mathbf{t}_{j+1}^H \mathbf{v}_i. \quad (2.18)$$

The Gram-Schmidt procedure in the Arnoldi Process guarantees $\mathbf{t}_{j+1}^H \mathbf{v}_i = 0$ for $i \leq j$. Therefore, the terms in the sum of (2.18) vanish and we get

$$\|\mathbf{t}_{j+1}\| = \frac{1}{\|\mathbf{t}_{j+1}\|} \mathbf{t}_{j+1}^H \mathbf{A}\mathbf{v}_j = \mathbf{v}_{j+1}^H \mathbf{A}\mathbf{v}_j = h_{j+1,j} \quad (2.19)$$

Algorithm 2.5 Modified Gram-Schmidt version of the Arnoldi-Process.

```

function Arnoldi(A, a):[matrix, matrix];
  v1 := a/||a||;
  for j := 1 to n do
    tj+1 := Apj;
    for i := 1 to j do
      hi,j := viHtj+1;
      tj+1 := tj+1 - hi,jvi;
    end;
    hj+1,j := ||tj+1||2;
    vj+1 := tj+1/hj+1,j;
    j := j + 1;
  end
  Arnoldi := [V, H];
end Arnoldi;

```

Let us recall that a vector \mathbf{v}_j is an element of $\mathcal{K}_j(\mathbf{A}, \mathbf{a})$ if and only if it can be rewritten as a polynomial in \mathbf{A}

ARNOLDI
VECTORS SPAN
 $\mathcal{K}(\mathbf{A}, \mathbf{a})$

$$\mathbf{v}_j = \sum_{k=0}^{j-1} \alpha_{k,j} \mathbf{A}^k \mathbf{a} \quad \text{for some } \alpha_{k,j}. \quad (2.20)$$

The following lemma shows that the Arnoldi process generates vectors $\mathbf{v}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{a})$ and that every generated vector \mathbf{v}_j is necessary to span $\mathcal{K}_j(\mathbf{A}, \mathbf{a})$ as long as it is not an invariant subspace.

Lemma 2.19 *Given a matrix \mathbf{A} and a vector \mathbf{a} . The Arnoldi process described by (2.17) guarantees*

$$\mathbf{v}_{j+1} \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{a}).$$

Moreover, as long as $\mathcal{K}_j(\mathbf{A}, \mathbf{a})$ is not an invariant Krylov space, the leading coefficient $\alpha_{j-1,j}$ of \mathbf{v}_j does not vanish.

Proof. We prove the statement by induction with respect to $j \geq 1$.

Initialization $\mathbf{v}_1 = \frac{1}{\|\mathbf{a}\|} \mathbf{a} \in \mathcal{K}_1(\mathbf{A}, \mathbf{a})$ and $\alpha_{0,1} = \frac{1}{\|\mathbf{a}\|} \neq 0$.

Induction Step We now use the hypothesis $\mathbf{v}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{a})$ and the iterative scheme to show the statement. In (2.17) we replace \mathbf{v}_j and \mathbf{v}_i in the manner of (2.20) by polynomials and obtain \mathbf{v}_{j+1} as

$$\mathbf{v}_{j+1} = \frac{1}{\|\mathbf{t}_{j+1}\|} \left(\mathbf{A} \sum_{k=0}^{j-1} \alpha_{k,j} \mathbf{A}^k \mathbf{a} - \sum_{i=1}^j h_{i,j} \sum_{k=0}^{i-1} \alpha_{k,i} \mathbf{A}^k \mathbf{a} \right). \quad (2.21)$$

This equation shows that \mathbf{v}_{j+1} is a polynomial in \mathbf{A} of degree j . For some $\alpha_{k,j+1}$ it can be rewritten as

$$\mathbf{v}_{j+1} = \sum_{k=0}^j \alpha_{k,j+1} \mathbf{A}^k \mathbf{a} \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{a}).$$

Together with the hypothesis $\alpha_{j-1,j} \neq 0$ equation (2.21) also implies

$$\alpha_{j,j+1} = \alpha_{j-1,j} / \|\mathbf{t}_{j+1}\| \neq 0.$$

□

$h_{i,j} = 0$ FOR
 $i > j + 1$

As a consequence of this lemma and the polynomial representation (2.20) of \mathbf{v}_j ,

$$\mathbf{A}^e \mathbf{v}_j \in \mathcal{K}_{j+e}(\mathbf{A}, \mathbf{a}). \quad (2.22)$$

The usage of the Gram-Schmidt procedure guarantees

$$\mathbf{v}_j \perp \mathcal{K}_{j-1}(\mathbf{A}, \mathbf{a}) \supset \mathcal{K}_{j-2}(\mathbf{A}, \mathbf{a}) \supset \dots \supset \mathcal{K}_1(\mathbf{A}, \mathbf{a}).$$

Equation (2.22) tells us that $\mathbf{A}^e \mathbf{v}_j \in \mathcal{K}_{j+e}(\mathbf{A}, \mathbf{a})$. Putting all these things together, we get

$$\mathbf{v}_k^H \mathbf{A}^e \mathbf{v}_j = 0 \quad \text{for } k > j + e. \quad (2.23)$$

The Arnoldi process can be expressed in matrix form as MATRIX REPRESENTATION

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_j\mathbf{H}_j + h_{j+1,j}\mathbf{v}_{j+1}\mathbf{e}_j^H = \mathbf{V}_{j+1}\tilde{\mathbf{H}}_{j+1}. \quad (2.24)$$

In this equation \mathbf{H}_j is the matrix containing the values $h_{i,k}$ with $i \leq j$ and $k \leq j$. $\tilde{\mathbf{H}}_{j+1}$ contains additional values $h_{j+1,k}$ with $k \leq j$.

Equation (2.23) states that \mathbf{H}_j is a $(j \times j)$ upper Hessenberg matrix. Matrix $\tilde{\mathbf{H}}_{j+1}$ is \mathbf{H}_j extended by a row containing one value in the last column. Thus $\tilde{\mathbf{H}}_{j+1}$ is of size $(j+1 \times j)$.

Multiplying (2.24) by \mathbf{V}_j on the left side, we get

$$\mathbf{V}_j^H \mathbf{A} \mathbf{V}_j = \mathbf{H}_j \quad (2.25)$$

If $j = n$, $\mathbf{V}_j = (\mathbf{V}_j^H)^{-1}$ is an orthogonal matrix and this equation describes the Arnoldi Process performing a similarity transformation on \mathbf{A} . However, there is no guarantee that $j = n$ can be achieved.

2.4.3 Symmetric Krylov Space Generation

Although we will not deal with hermitian system matrices, it is necessary to understand how a Krylov space is constructed in the hermitian case for the understanding of the decisions in the next subsection. Hence let \mathbf{A} be a hermitian matrix.

With \mathbf{A} being hermitian, a consequence of (2.23) is

$$\mathbf{v}_k^H \mathbf{A}^e \mathbf{v}_j = \mathbf{v}_j^H \mathbf{A}^e \mathbf{v}_k = 0 \quad \text{for } k > j + e.$$

So the sum in the Arnoldi process can be replaced by only two terms. Scheme (2.17) becomes

$$\begin{aligned}
 \mathbf{v}_1 &:= \mathbf{a}/\|\mathbf{a}\| \\
 \mathbf{t}_{j+1} &:= \mathbf{A}\mathbf{v}_j - \mathbf{v}_j^H \mathbf{A}\mathbf{v}_j \mathbf{v}_j - \underbrace{\mathbf{v}_{j-1}^H \mathbf{A}\mathbf{v}_j}_{\|\mathbf{t}_j\|} \mathbf{v}_{j-1} \\
 \mathbf{v}_{j+1} &:= \mathbf{t}_{j+1}/\|\mathbf{t}_{j+1}\|
 \end{aligned}$$

This process is known as the symmetric *Lanczos Process*. It is hidden behind the CG algorithm, but it also has many other applications.

As a consequence of the general Arnoldi process (2.19), we find that $\|\mathbf{t}_{j+1}\| = \mathbf{v}_{j+1}^H \mathbf{A}\mathbf{v}_j = \mathbf{v}_j^H \mathbf{A}\mathbf{v}_{j+1}$. This value can be reused for determining \mathbf{t}_{j+2} .

With \mathbf{A} being hermitian, we conclude that the iterative scheme becomes less expensive in work and in memory. Each step of the scheme has equal costs. Only two basis vectors have to be stored to determine the next one.

The matrix representation \mathbf{H}_j reduces to a tridiagonal matrix.

2.4.4 Lanczos Biorthogonalization

The cost effectiveness of the symmetric Lanczos process compared to the general Arnoldi Process is so huge that people are willing to pay a price to keep it in the general case. If \mathbf{H}_j has to stay tridiagonal with \mathbf{A} being a general matrix, the usage of a Krylov space on \mathbf{A}^H suggests itself.

Let \mathbf{V}_j be a basis of $\mathcal{K}_j(\mathbf{A}, \mathbf{a})$ and \mathbf{W}_j a basis of $\mathcal{K}_j(\mathbf{A}^H, \mathbf{a})$. Following the strategy of the Arnoldi process let both bases be constructed such that the following expressions are true

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_j\mathbf{H}_j + h_{j+1,j}\mathbf{v}_{j+1}\mathbf{e}_j^H, \quad (2.26)$$

$$\mathbf{A}^H\mathbf{W}_j = \mathbf{W}_j\mathbf{K}_j + k_{j+1,j}\mathbf{w}_{j+1}\mathbf{e}_j^H. \quad (2.27)$$

Equation (2.24) tells us that \mathbf{H}_j and \mathbf{K}_j are upper Hessenberg matrices. However, since we want them to be tridiagonal, their components as well as the values $h_{j+1,j}$ and $k_{j+1,j}$ have to be determined in a different way in the Arnoldi process.

Multiplying (2.26) by \mathbf{W}_j^H and (2.27) by \mathbf{V}_j^H on the left side, both equations can be combined. DETERMINING
 \mathbf{H}_j AND \mathbf{K}_j

$$\begin{aligned} \mathbf{W}_j^H(\mathbf{V}_j\mathbf{H}_j + h_{j+1,j}\mathbf{v}_{j+1}\mathbf{e}_j^H) &= \\ \mathbf{W}_j^H\mathbf{A}\mathbf{V}_j &= (\mathbf{V}_j^H\mathbf{A}^H\mathbf{W}_j)^H \\ &= (k_{j+1,j}\mathbf{e}_j\mathbf{w}_{j+1}^H + \mathbf{K}_j^H\mathbf{W}_j^H)\mathbf{V}_j \end{aligned}$$

Claiming that the bases \mathbf{W}_j and \mathbf{V}_j are biorthogonal, i.e. $\mathbf{W}_j^H\mathbf{V}_j = \mathbf{I}$, cleans up this equation and we get

$$\mathbf{W}_j^H\mathbf{A}\mathbf{V}_j = \mathbf{H}_j = \mathbf{K}_j^H = (\mathbf{V}_j^H\mathbf{A}^H\mathbf{W}_j)^H. \quad (2.28)$$

Since \mathbf{H}_j and \mathbf{K}_j are upper Hessenberg matrices, they must be tridiagonal. Equation (2.28) even reveals the values for their components: $h_{i,j} = \mathbf{w}_i^H\mathbf{A}\mathbf{v}_j$ and $k_{i,j} = \mathbf{v}_i^H\mathbf{A}^H\mathbf{w}_j$.

The price to pay for keeping the tridiagonal structure is: Two bases instead of one (including matrix multiplication by \mathbf{A}^H) and the replacement of orthogonality by biorthogonality. These considerations lead to the *Lanczos biorthogonalization*.

$\mathbf{v}_1 := \mathbf{a}/\ \mathbf{a}\ $	$\mathbf{w}_1 := \mathbf{a}/\ \mathbf{a}\ $
$\mathbf{t}_{j+1} := \mathbf{A}\mathbf{v}_j - \sum_{i=j-1}^j h_{i,j}\mathbf{v}_i$	$\mathbf{u}_{j+1} := \mathbf{A}^H\mathbf{w}_j - \sum_{i=j-1}^j h_{j,i}\mathbf{w}_i$
$\mathbf{v}_{j+1} := \mathbf{t}_{j+1}/h_{j+1,j}$	$\mathbf{w}_{j+1} := \mathbf{u}_{j+1}/h_{j,j+1}$
with $h_{i,j} = \mathbf{w}_i^H \mathbf{A}\mathbf{v}_j$	

DETERMINING Values $h_{j+1,j}$ and $h_{j,j+1}$ are chosen with respect to the condition $\mathbf{v}_{j+1}^H \mathbf{w}_{j+1} = 1$, which results in

$$h_{j+1,j}h_{j,j+1} = \mathbf{t}_{j+1}^H \mathbf{u}_{j+1}.$$

One of these two values can be arbitrarily chosen, the other one is bound by this equation. Common values are

$$h_{j+1,j} = \sqrt{|\mathbf{t}_{j+1}^H \mathbf{u}_{j+1}|} \quad \text{and} \quad h_{j,j+1} = \mathbf{t}_{j+1}^H \mathbf{u}_{j+1} / h_{j+1,j}.$$

With this choice \mathbf{H}_j is symmetric with respect to absolute values.

As in the symmetric Lanczos algorithm, values $h_{j+1,j}$ and $h_{j,j+1}$ can be reused for the determination of \mathbf{t}_{j+2} and \mathbf{u}_{j+2} . An algorithm for the Lanczos biorthogonalization is shown in Algorithm 2.6.

Whereas the Arnoldi process ends exactly when $h_{j+1,j}$ vanishes, this is not true for Lanczos biorthogonalization. In fact the Lanczos biorthogonalization can have a serious problem, when $|\mathbf{t}_j^H \mathbf{u}_j|$ becomes small. Such an event is called *breakdown* or *near breakdown* of the biorthogonalization. A number of approaches has been

Algorithm 2.6 Lanczos Biorthogonalization.

```

function BiOrth(A, a):[matrix,matrix,matrix];
  v1 := w1 := a/||a||;
  h1,1 := w1HA v1;
  t2 := A v1 - h1,1 v1; u2 := AHw1 - h1,1 w1;
  for j := 2 to n do
    hj,j-1 := √|tjHuj|;
    hj-1,j := 1/hj,j-1 tjHuj;

    vj := tj/hj,j-1;
    wj := uj/hj-1,j;

    hj,j := wjHA vj;

    tj+1 := A vj - hj,j vj - hj-1,j vj-1;
    uj+1 := AHwj - hj,j wj - hj,j-1 wj-1
  end
  BiOrth := [W, V, H]
end BiOrth;

```

developed to *avoid* such breakdowns, other approaches try to *deal with it*. Saad [32, pp. 208] gives an overview of these methods.

2.4.5 General Arnoldi Type Methods

Full Orthogonalization Method The full orthogonalization method combines the Arnoldi process with the OR approach.

Definition 2.20 *The Full Orthogonalization Method (FOM) is a Krylov space method based on the Arnoldi process with projection space $\mathcal{P}_j = \mathcal{C}_j = \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$.*

In step j the Arnoldi process generates an orthonormal basis \mathbf{V}_j . Using $\mathcal{C}_j = \text{span}\{\mathbf{V}_j\}$ the residual of FOM becomes

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_j\mathbf{z}_j$$

The OR approach requests $\mathcal{P}_j = \text{span}\{\mathbf{V}_j\}$. Since \mathbf{r}_0 is an element of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$, (2.25) and (2.7) imply that

$$\mathbf{z}_j = \mathbf{H}_j^{-1} \mathbf{e}_1 \|\mathbf{r}_0\|. \quad (2.29)$$

WHEN TO
STOP?

Evaluating the residual norm in each step by

$$\|\mathbf{r}_j\| = \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_j\mathbf{z}_j\|$$

is expensive. However, we know by (2.24) that the residual can be expressed as

$$\mathbf{r}_j = \underbrace{(\mathbf{r}_0 - \mathbf{V}_j\mathbf{H}_j\mathbf{z}_j)}_{=0} - h_{j+1,j}\mathbf{v}_{j+1}\mathbf{e}_j^H\mathbf{z}_j. \quad (2.30)$$

By the choice of \mathbf{z}_j in (2.29) FOM cares for the bracketed expression to vanish and hence

$$\rho_j := \|\mathbf{r}_j\| = h_{j+1,j}|\mathbf{e}_j^H\mathbf{z}_j|,$$

which is less expensive to calculate.

The considerations made so far are summarized in Algorithm 2.7. Some improvements in this algorithm could be made with respect to evaluating the component vector \mathbf{z}_j by (2.29). However, these improvements have been initially developed for GMRES treated in the next subsection and their necessity is much better motivated by that algorithm. That is why this technique is explained in the next subsection.

Equation (2.30) gives rise to the two following properties of the FOM Algorithm.

Algorithm 2.7 Full orthogonalization method (FOM).

function FOM(\mathbf{A} , \mathbf{b} , \mathbf{s}_0 , ε):**vector**;

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$; $\rho_0 := \|\mathbf{r}_0\|$; $\mathbf{v}_1 := \mathbf{r}_0/\rho_0$;

$j := 1$;

while $\rho_{j-1} > \varepsilon\|\mathbf{b}\|$ **and** $j \leq n$ **do**

$\mathbf{t}_{j+1} := \mathbf{A}\mathbf{v}_j$;

for $i := 1$ **to** j **do**

$h_{i,j} := \mathbf{v}_i^H \mathbf{t}_{j+1}$;

$\mathbf{t}_{j+1} := \mathbf{t}_{j+1} - h_{i,j}\mathbf{v}_i$;

end;

$\mathbf{z}_j := \mathbf{H}_j^{-1} \mathbf{e}_1 \|\mathbf{r}_0\|$;

$h_{j+1,j} := \|\mathbf{t}_{j+1}\|$;

$\mathbf{v}_{j+1} := \mathbf{t}_{j+1}/h_{j+1,j}$;

$\rho_j := h_{j+1,j} |\mathbf{e}_j^H \mathbf{z}_j|$;

$j := j + 1$

end;

FOM := $\mathbf{s}_0 + \mathbf{V}_{j-1}\mathbf{z}_{j-1}$

end FOM;

Property 2.21 *The residual vectors $\mathbf{r}_0 \dots \mathbf{r}_{j-1}$ generated by the FOM algorithm span the Krylov space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$.*

Proof. Equation (2.30) makes sure that $\mathbf{r}_i = c_i \mathbf{v}_{i+1}$ for some constant value c_i and $i = 1 \dots j-1$. By construction \mathbf{r}_0 is an element of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and $\mathbf{v}_i, i = 1 \dots j$ span $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$. Therefore, the residual vectors $\mathbf{r}_0 \dots \mathbf{r}_{j-1}$ also span $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$. \square

Property 2.22 *The residual vectors generated by FOM are orthogonal.* \square

Additional properties include the possible Galerkin breakdown of FOM mentioned in section 2.3 and the rather unpredictable behavior of the residual norm.

Generalized Minimum Residual Method The generalized minimum residual method combines the MR approach with of the projection methods with the Arnoldi process.

Definition 2.23 *The Generalized Minimum Residual Method (GMRES) is a Krylov space method based on the Arnoldi process with projection space $\mathcal{P}_j = \mathcal{Q}_j = \text{span}\{\mathbf{A}\mathbf{V}_j\}$.*

In each step j the 2-norm of the residual vector $\mathbf{r}_j := \mathbf{r}_0 - \mathbf{A}\mathbf{V}_j\mathbf{z}_j$ is minimized with respect to \mathbf{z}_j . With the help of (2.24) and the orthogonality of \mathbf{V}_{j+1} this norm is transformed into

$$\|\mathbf{r}_j\| = \|\mathbf{r}_0 - \mathbf{V}_{j+1}\tilde{\mathbf{H}}_j\mathbf{z}_j\| = \|\mathbf{e}_1\|\mathbf{r}_0\| - \tilde{\mathbf{H}}_j\mathbf{z}_j\|.$$

SOLVING THE
LEAST
SQUARES
PROBLEM

Due to the special structure of $\tilde{\mathbf{H}}_j$ the least squares problem $\tilde{\mathbf{H}}_j\mathbf{z}_j \simeq \mathbf{e}_1\|\mathbf{r}_0\|$ can be solved by reusing much information from step $j - 1$.

Let

$$\mathbf{Q}_j \begin{bmatrix} \mathbf{R}_j \\ 0 \end{bmatrix} = \tilde{\mathbf{H}}_j \quad (2.31)$$

denote the QR decomposition of $\tilde{\mathbf{H}}_j$. Because $\tilde{\mathbf{H}}_j$ contains just one lower diagonal to be eliminated, only one Givens rotation per step is necessary to gain the decomposition. Let $\mathbf{G}_j(k)$ denote the Givens matrix in step j that eliminates $h_{j+1,j}$, precisely

$$\mathbf{G}_j(k) := \begin{bmatrix} \mathbf{1}_{(k-2)} & & & \\ & \cos_k & \sin_k & \\ & -\sin_k & \cos_k & \\ & & & \mathbf{1}_{(j-k)} \end{bmatrix}. \quad (2.32)$$

Then \mathbf{Q}_j becomes $\mathbf{Q}_j = \mathbf{G}_j(1)^H \times \dots \times \mathbf{G}_j(j)^H$. Let also

$$\begin{bmatrix} \mathbf{g}_j \\ \rho_{j+1} \end{bmatrix} := \mathbf{Q}_j^H \mathbf{e}_1 \|\mathbf{r}_0\|. \quad (2.33)$$

Thus the least squares problem has been transformed into a linear system $\mathbf{R}_j \mathbf{z}_j = \mathbf{g}_j$ that has to be solved in each step j and cannot be simplified any further. \mathbf{R}_j is updated columnwise from step to step. To do this all further cosines and sines from the Givens matrices have to be stored. However, \mathbf{g}_j and ρ_{j+1} can easily be updated from one step to the other by the following recursion

$$\begin{aligned} \rho_1 &:= \|\mathbf{r}_0\|, & \mathbf{g}_0 &:= \mathbf{0} \\ \rho_j &:= -\rho_{j-1} \sin_j, & \mathbf{g}_j &:= \begin{bmatrix} \mathbf{g}_{j-1} \\ \rho_j \cos_j \end{bmatrix}. \end{aligned} \quad (2.34)$$

This update procedure can also be applied to FOM.

The residual norm of the least squares solution is

$$\|\mathbf{r}_j\| = \|\mathbf{e}_1 \|\mathbf{r}_0\| - \tilde{\mathbf{H}}_j \mathbf{z}_j\| = |\rho_{j+1}|,$$

and therefore its norm can be easily obtained by $\|\mathbf{r}_j\| = |\rho_{j+1}|$.

The considerations made so far are summarized as a function in Algorithm 2.8. In this algorithm the elements of \mathbf{R}_j are denoted by $r_{i,j}$. In a practical implementation of the algorithm one would not store $\tilde{\mathbf{H}}_j$ and \mathbf{R}_j but rather replace the components of $\tilde{\mathbf{H}}_j$ with the values of \mathbf{R}_j .

Property 2.24 *The residual norm $\|\mathbf{r}_j\|$ of GMRES converges monotonously.*

Proof. Since GMRES satisfies all the conditions of Lemma 2.15, it has a guaranteed monotonous convergence. \square

Algorithm 2.8 Generalized minimum residual method (GMRES).

function GMRES(\mathbf{A} , \mathbf{b} , \mathbf{s}_0 , ε):**vector**;
 $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$; $\rho_0 := \|\mathbf{r}_0\|$; $\mathbf{v}_1 := \mathbf{r}_0/\rho_0$;
 $j := 1$;
while $|\rho_{j-1}| > \varepsilon\|\mathbf{b}\|$ **and** $j \leq n$ **do**
 {Arnoldi Process}
 $\mathbf{t}_{j+1} := \mathbf{A}\mathbf{v}_j$;
 for $i := 1$ **to** j **do**
 $h_{i,j} := \mathbf{v}_i^H \mathbf{t}_{j+1}$; $\mathbf{t}_{j+1} := \mathbf{t}_{j+1} - h_{i,j}\mathbf{v}_i$;
 end;
 $h_{j+1,j} := \|\mathbf{t}_{j+1}\|$; $\mathbf{v}_{j+1} := \mathbf{t}_{j+1}/h_{j+1,j}$;

 {Update previous Givens Rotations}
 $r_{1,j} := h_{1,j}$;
 for $i := 1$ **to** $j-1$,
 $\begin{bmatrix} r_{i,j} \\ r_{i+1,j} \end{bmatrix} := \begin{bmatrix} \cos_i & \sin_i \\ -\sin_i & \cos_i \end{bmatrix} \begin{bmatrix} r_{i,j} \\ h_{i+1,j} \end{bmatrix}$
 end;

 {Actual Givens Rotation}
 $\tan := h_{j+1,j}/r_{j,j}$;
 $\cos_j := 1/\sqrt{1 + \tan^2}$; $\sin_j := \cos_j \tan$;
 $r_{j,j} := r_{j,j} \cos_j + h_{j+1,j} \sin_j$;

 {Right hand side and component vector}
 $g_j^{(j)} := \rho_j \cos_j$; $\rho_{j+1} := -\rho_j \sin_j$;
 $\mathbf{z}_j := \mathbf{R}_j^{-1} \mathbf{g}^{(j)}$;

 $j := j + 1$
end;
 GMRES := $\mathbf{s}_0 + \mathbf{V}_{j-1}\mathbf{z}_{j-1}$
end GMRES;

2.4.6 General Lanczos Type Methods

Biconjugate Gradient Method The Biconjugate Gradient Method (BiCG) combines the OR approach of the projection methods with the Lanczos biorthogonalization. However, to make use of the advantages of this process, some changes have to be introduced to the OR paradigm.

Let V_j be the basis of $\mathcal{C}_j = \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and W_j the basis of $\mathcal{K}_j(\mathbf{A}^H, \mathbf{r}_0)$ generated by a biorthogonal Lanczos process. In step j an OR method would determine the residual vector \mathbf{r}_j by

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{A}V_j z_j \quad \text{with} \quad z_j := (V_j^H \mathbf{A} V_j)^{-1} V_j^H \mathbf{r}_0.$$

However, to make use of the tridiagonal structure of H_j generated by the Lanczos process, z_j is determined by BiCG rather the following way.

$$z_j := (W_j^H \mathbf{A} V_j)^{-1} W_j^H \mathbf{r}_0 = H_j^{-1} \mathbf{e}_1 \|\mathbf{r}_0\|$$

This way \mathbf{r}_j is determined such that it is orthogonal to $\mathcal{K}_j(\mathbf{A}^H, \mathbf{r}_0)$ rather than the correction space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$.

Definition 2.25 *The Biconjugate Gradient Method (BiCG) is a Krylov space method based on Lanczos biorthogonalization with projection space $\mathcal{K}_j(\mathbf{A}^H, \mathbf{r}_0)$.*

Due to the usage of the Lanczos biorthogonalization neither V_j nor W_j is needed to determine z_j . But so far V_j is still needed to determine \mathbf{r}_j . Techniques developed for CG however allow us to eliminate V_j in \mathbf{r}_j and therefore to save memory.

ELIMINATING
THE FULL
LANCZOS
BASES

Since \mathbf{H}_j is tridiagonal, it can be decomposed very easily into matrices \mathbf{L}_j and \mathbf{U}_j .

$$\mathbf{H}_j = \begin{bmatrix} \beta_1 & \gamma_1 & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{j-1} & \beta_{j-1} & \gamma_{j-1} \\ & & & \alpha_j & \beta_j \end{bmatrix} =$$

$$\mathbf{L}_j \mathbf{U}_j = \begin{bmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \lambda_j & 1 \end{bmatrix} \begin{bmatrix} \mu_1 & \gamma_1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \mu_{j-1} & \gamma_{j-1} \\ & & & & \mu_j \end{bmatrix}$$

with λ_k, μ_k recursively being defined by

$$\begin{aligned} \mu_1 &:= \beta_1, \\ \lambda_{k+1} &:= \alpha_{k+1}/\mu_k, \quad \mu_{k+1} := \beta_{k+1} - \lambda_{k+1}\gamma_k, \quad 1 \leq k < j. \end{aligned}$$

Let us define now a basis of direction vectors

$$\mathbf{p}_j := \mathbf{V}_j \mathbf{U}_j^{-1} \quad (2.35)$$

and an associated coordinate vector $\mathbf{y}_j := \mathbf{L}_j^{-1} \mathbf{e}_1 \|\mathbf{r}_0\|$. With these definitions \mathbf{r}_j reads as

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{A} \mathbf{p}_j \mathbf{y}_j.$$

Evaluating $\mathbf{p}_j \mathbf{U}_j = \mathbf{V}_j$ we see that $\mathbf{v}_j = \gamma_{j-1} \mathbf{p}_{j-1} + \mu_j \mathbf{p}_j$. Thus \mathbf{p}_j can be generated recursively by

$$\mathbf{p}_1 := \frac{1}{\mu_1} \mathbf{v}_1 = \frac{\mathbf{r}_0^H \mathbf{r}_0}{\mathbf{r}_0^H \mathbf{A} \mathbf{r}_0} \mathbf{r}_0, \quad (2.36)$$

$$\mathbf{p}_{k+1} := \frac{1}{\mu_{k+1}} (\mathbf{v}_{k+1} - \gamma_k \mathbf{p}_k). \quad (2.37)$$

Furthermore the structure of L_j makes it also possible to determine the components of \mathbf{y}_j in recursive way.

$$\begin{aligned} y_1 &:= \|\mathbf{r}_0\| \\ y_{k+1} &:= -\lambda_{k+1} y_k \end{aligned}$$

With these recursions the residual vector \mathbf{r}_j can now also be recursively obtained by

$$\begin{aligned} \mathbf{r}_0 &:= \mathbf{b} - \mathbf{A}\mathbf{s}_0 \\ \mathbf{r}_{j+1} &:= \mathbf{r}_j - \mathbf{A}\mathbf{p}_{j+1}y_{j+1} \end{aligned}$$

The considerations made so far result in a first draft of BiCG shown in Algorithm 2.9.

The residual vectors \mathbf{r}_j and the direction vectors \mathbf{p}_j have some important properties that allow us to improve this algorithm.

PROPERTIES
OF THE
ALGORITHM

Let us consider the additional linear system

$$\mathbf{A}^H \tilde{\mathbf{s}} = \mathbf{b}. \quad (2.38)$$

Solving this system in the same manner as (2.1), the residual vector for $\tilde{\mathbf{s}}_j$ becomes

$$\tilde{\mathbf{r}}_j = \tilde{\mathbf{r}}_0 - \mathbf{A}^H \mathbf{W}_j \tilde{\mathbf{z}}_j \quad \text{with} \quad \tilde{\mathbf{z}}_j = \mathbf{H}_j^{-H} \mathbf{e}_1 \|\tilde{\mathbf{r}}_0\|. \quad (2.39)$$

Similarly as in (2.35) let us also define a basis of search directions $\tilde{\mathbf{P}}_j$ for the residual vector (2.39),

$$\tilde{\mathbf{P}}_j := \mathbf{V}_j \mathbf{L}_j^{-H} \quad (2.40)$$

as well as its associated coordinate vector

$$\tilde{\mathbf{y}}_j := \mathbf{U}_j^{-H} \mathbf{e}_1 \|\tilde{\mathbf{r}}_0\|,$$

such that the residual vector becomes $\tilde{\mathbf{r}}_j = \tilde{\mathbf{r}}_0 - \mathbf{A}^H \tilde{\mathbf{P}}_j \tilde{\mathbf{y}}_j$.

Algorithm 2.9 A first version of the biconjugate gradient method (BiCG).

function DraftBiCG(\mathbf{A} , \mathbf{b} , \mathbf{s}_0 , ε):**vector**;

$$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0; \mathbf{v}_0 := \mathbf{w}_0 := \mathbf{p}_0 := \mathbf{0};$$

$$\alpha_1 := \mu_0 := 1; \mathbf{y}_0 := -\|\mathbf{r}_0\|$$

$$\mathbf{v}_1 := \mathbf{w}_1 := \mathbf{r}_0 / \|\mathbf{r}_0\|;$$

$$j := 1;$$

while $\|\mathbf{r}_{j-1}\| > \varepsilon \|\mathbf{b}\|$ **do**

$$\beta_j := \mathbf{w}_j^H \mathbf{A} \mathbf{v}_j;$$

$$\mathbf{t}_{j+1} := \mathbf{A} \mathbf{v}_j - \beta_j \mathbf{v}_j - \gamma_{j-1} \mathbf{v}_{j-1};$$

$$\mathbf{u}_{j+1} := \mathbf{A}^H \mathbf{w}_j - \beta_j \mathbf{w}_j - \alpha_j \mathbf{w}_{j-1}$$

$$\lambda_j := \alpha_j / \mu_{j-1};$$

$$\mathbf{y}_j := -\lambda_j \mathbf{y}_{j-1};$$

$$\mu_j := \beta_j - \lambda_j \gamma_{j-1};$$

$$\mathbf{p}_j := \mathbf{v}_j / \mu_j - \gamma_{j-1} / \mu_j \mathbf{p}_{j-1};$$

$$\mathbf{r}_j := \mathbf{r}_{j-1} - \mathbf{A} \mathbf{p}_j \mathbf{y}_j;$$

$$\mathbf{s}_j := \mathbf{s}_{j-1} + \mathbf{p}_j \mathbf{y}_j;$$

$$\alpha_{j+1} := \sqrt{|\mathbf{t}_{j+1}^H \mathbf{u}_{j+1}|};$$

$$\gamma_j := \frac{1}{\alpha_{j+1}} \mathbf{t}_{j+1}^H \mathbf{u}_{j+1};$$

$$\mathbf{v}_{j+1} := \mathbf{t}_{j+1} / \alpha_{j+1};$$

$$\mathbf{w}_{j+1} := \mathbf{u}_{j+1} / \gamma_j;$$

$$j := j + 1$$

end;

$$\text{DraftBiCG} := \mathbf{s}_{j-1};$$

end DraftBiCG;

Property 2.26 The residual vectors $\mathbf{r}_0, \dots, \mathbf{r}_{j-1}$ span the Krylov space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and the residual vectors $\bar{\mathbf{r}}_0, \dots, \bar{\mathbf{r}}_{j-1}$ span the Krylov space $\mathcal{K}_j(\mathbf{A}^H, \bar{\mathbf{r}}_0)$. Specifically, $\mathbf{r}_j = \mathbf{c}_j \mathbf{v}_{j+1}$ and $\bar{\mathbf{r}}_j = \hat{\mathbf{c}}_j \mathbf{w}_{j+1}$.

Proof. Using (2.26) and (2.27) we get for the residual vectors \mathbf{r}_j and $\bar{\mathbf{r}}_j$

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{V}_j \mathbf{H}_j \mathbf{z}_j - h_{j+1,j} \mathbf{v}_{j+1} \mathbf{e}_j^H \mathbf{z}_j, \quad (2.41)$$

$$\bar{\mathbf{r}}_j = \bar{\mathbf{r}}_0 - \mathbf{W}_j \mathbf{H}_j^H \bar{\mathbf{z}}_j - h_{j,j+1} \mathbf{w}_{j+1} \mathbf{e}_j^H \bar{\mathbf{z}}_j. \quad (2.42)$$

By construction the initial residual vectors \mathbf{r}_0 and $\bar{\mathbf{r}}_0$ are elements of the space spanned by the column vectors of \mathbf{W}_j and \mathbf{V}_j respectively. $\mathbf{W}_j^H \mathbf{r}_j = 0$ and $\mathbf{V}_j^H \bar{\mathbf{r}}_j = 0$ make sure that the first two terms of (2.41) and (2.42) vanish, establishing $\mathbf{r}_j = c_j \mathbf{v}_{j+1}$ and $\bar{\mathbf{r}}_j = \hat{c}_j \mathbf{w}_{j+1}$ for some constant values c_j and \hat{c}_j . Since \mathbf{V}_j is a basis of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and \mathbf{W}_j is a basis of $\mathcal{K}_j(\mathbf{A}^H, \bar{\mathbf{r}}_0)$, $[\mathbf{r}_0 \dots \mathbf{r}_{j-1}]$ must be a basis of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and $[\bar{\mathbf{r}}_0 \dots \bar{\mathbf{r}}_{j-1}]$ must be a basis of $\mathcal{K}_j(\mathbf{A}^H, \bar{\mathbf{r}}_0)$. \square

Property 2.27 *The direction vectors $\mathbf{p}_1, \dots, \mathbf{p}_j$ span the Krylov space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and the direction vectors $\bar{\mathbf{p}}_1, \dots, \bar{\mathbf{p}}_j$ span the Krylov space $\mathcal{K}_j(\mathbf{A}^H, \bar{\mathbf{r}}_0)$.*

Proof. We prove the statement by induction with respect to j for vectors $\mathbf{p}_1, \dots, \mathbf{p}_j$. An analog proof exists for all quantities connected to the transposed linear system (2.38).

Initialization Equation (2.36) tells us that \mathbf{p}_1 is proportional to \mathbf{v}_1 .

Induction Step By hypothesis \mathbf{p}_j is an element of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ and $\bar{\mathbf{p}}_j$ is an element of $\mathcal{K}_j(\mathbf{A}^H, \bar{\mathbf{r}}_0)$. We show $\mathbf{p}_{j+1} \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{r}_0)$.

Equation (2.37) shows that \mathbf{p}_{j+1} is a linear combination of $\mathbf{v}_{j+1} \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{r}_0)$ and $\mathbf{p}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$. Therefore, $\mathbf{p}_{j+1} \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{r}_0)$.

Since \mathbf{v}_{j+1} contributes with a non zero portion, \mathbf{p}_{j+1} is a vector with a contribution in this direction and it is the only one. \square

Property 2.28 (Biorthogonality) *The residual vectors \mathbf{r}_j and $\bar{\mathbf{r}}_j$ are biorthogonal, i.e. $\bar{\mathbf{r}}_i^H \mathbf{r}_j = 0$ for $i \neq j$.*

Proof. The biorthogonality of $\bar{\mathbf{r}}_i$ and \mathbf{r}_j is a direct consequence of Property 2.26 and the biorthogonality of \mathbf{w}_i and \mathbf{v}_j . \square

Property 2.29 (Biconjugacy) *The bases of direction vectors $\bar{\mathbf{P}}_j$ and \mathbf{P}_j as defined by Equations (2.35) and (2.40) are biconjugate, i.e.*

$$\bar{\mathbf{P}}_j^H \mathbf{A} \mathbf{P}_j = 1.$$

Proof.

$$\bar{\mathbf{P}}_j^H \mathbf{A} \mathbf{P}_j = \mathbf{L}_j^{-1} \mathbf{W}_j^H \mathbf{A} \mathbf{V}_j \mathbf{U}_j^{-1} = \mathbf{L}_j^{-1} \mathbf{H}_j \mathbf{U}_j^{-1} = 1.$$

\square

DEVELOPING
THE COMMON
BiCG
ALGORITHM

With these properties we are able to derive the more elegant BiCG algorithm, which uses the sequence of the two types of residuals and the direction vectors also for the generation of the Krylov spaces. This way this algorithm gets on without the basis vectors \mathbf{v}_j and \mathbf{w}_j .

Properties 2.27 and 2.26 give rise to the following ansatz for a new sequence of direction vectors \mathbf{q}_j and $\bar{\mathbf{q}}_j$, $j = 1, 2, \dots$

$$\begin{aligned} \mathbf{q}_1 &:= \mathbf{r}_0, & \mathbf{q}_{j+1} &:= \mathbf{r}_j - \sum_{i=1}^j \psi_{i,j} \mathbf{q}_i \quad \text{with} \quad \psi_{i,j} := \frac{\bar{\mathbf{q}}_i^H \mathbf{A} \mathbf{r}_j}{\bar{\mathbf{q}}_i^H \mathbf{A} \mathbf{q}_i} \\ \bar{\mathbf{q}}_1 &:= \bar{\mathbf{r}}_0, & \bar{\mathbf{q}}_{j+1} &:= \bar{\mathbf{r}}_j - \sum_{i=1}^j \bar{\psi}_{i,j} \bar{\mathbf{q}}_i \quad \text{with} \quad \bar{\psi}_{i,j} := \frac{\bar{\mathbf{r}}_j^H \mathbf{A} \mathbf{q}_i}{\bar{\mathbf{q}}_i^H \mathbf{A} \mathbf{q}_i} \end{aligned}$$

The factors $\psi_{i,j}$ and $\bar{\psi}_{i,j}$ have already been chosen such that Property 2.29 is accomplished.

Since the new direction vectors have the same direction as the old ones, the residual vectors can be obtained the same way as before:

$$\begin{aligned} \mathbf{r}_0 &:= \bar{\mathbf{r}}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0, \\ \mathbf{r}_{j+1} &:= \mathbf{r}_j - \omega_j \mathbf{A}\mathbf{q}_{j+1} \quad \text{with } \omega_j := \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{r}}_j^H \mathbf{A}\mathbf{q}_{j+1}} \\ \bar{\mathbf{r}}_{j+1} &:= \bar{\mathbf{r}}_j - \bar{\omega}_j \mathbf{A}^H \bar{\mathbf{q}}_{j+1} \quad \text{with } \bar{\omega}_j := \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{q}}_{j+1}^H \mathbf{A}\mathbf{r}_j} \end{aligned}$$

The factors ω_j and $\bar{\omega}_j$ have been chosen in order to ensure $\bar{\mathbf{r}}_{j+1}^H \mathbf{r}_j = 0$ and $\mathbf{r}_j^H \mathbf{r}_{j+1} = 0$. The definitions of \mathbf{q}_{j+1} and $\bar{\mathbf{q}}_{j+1}$ imply

$$\begin{aligned} \bar{\mathbf{r}}_j^H \mathbf{A}\mathbf{q}_{j+1} &= \mathbf{q}_{j+1}^H \mathbf{A}\mathbf{q}_{j+1} + \sum_{i=1}^j \bar{\psi}_{i,j} \bar{\mathbf{q}}_i^H \mathbf{A}\mathbf{q}_{j+1} = \\ &\bar{\mathbf{q}}_{j+1}^H \mathbf{A}\mathbf{q}_{j+1} + \sum_{i=1}^j \psi_{i,j} \bar{\mathbf{q}}_{j+1}^H \mathbf{A}\mathbf{q}_i = \bar{\mathbf{q}}_{j+1}^H \mathbf{A}\mathbf{r}_j. \end{aligned}$$

This lets us even simplify

$$\omega_j := \bar{\omega}_j := \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{q}}_{j+1}^H \mathbf{A}\mathbf{q}_{j+1}}.$$

The next two lemmas show that the definition of \mathbf{r}_j and $\bar{\mathbf{r}}_j$ is enough to satisfy $\bar{\mathbf{r}}_k^H \mathbf{r}_j = 0$ for $k \neq j$.

Lemma 2.30 *The definitions of \mathbf{r}_j and $\bar{\mathbf{q}}_j$ imply $\bar{\mathbf{r}}_k^H \mathbf{r}_j = 0$ for $k < j$.*

Proof. We prove the statement by induction with respect to $d := j - k$.

Initialization For $d = 1$, $\bar{\mathbf{r}}_{j-1}^H \mathbf{r}_j = 0$ because ω_j has been chosen to ensure that.

Induction Step We suppose that $d > 1$. The induction hypothesis is $\bar{\mathbf{r}}_{j-m}^H \mathbf{r}_j = 0$ for $0 < m < d$. With the definition of \mathbf{r}_j we get

$$\bar{\mathbf{r}}_{j-d}^H \mathbf{r}_j = \bar{\mathbf{r}}_{j-d}^H \mathbf{r}_{j-1} - \omega_{j-1} \bar{\mathbf{r}}_{j-d}^H \mathbf{A} \mathbf{q}_j$$

The first term of the right hand side expression vanishes because of the induction hypothesis. The second term can be reduced to zero by using the definition of $\bar{\mathbf{q}}_{j-d+1}$.

$$\bar{\mathbf{r}}_{j-d}^H \mathbf{A} \mathbf{q}_j = \bar{\mathbf{q}}_{j-d+1}^H \mathbf{A} \mathbf{q}_j + \sum_{i=1}^{j-d} \bar{\psi}_{i,j-d} \bar{\mathbf{q}}_i^H \mathbf{A} \mathbf{q}_j$$

Since $j - d + 1 < j$ all the terms vanish, making $\bar{\mathbf{r}}_{j-d}^H \mathbf{A} \mathbf{q}_j = 0$ and hence $\bar{\mathbf{r}}_{j-d}^H \mathbf{r}_j = 0$. \square

Lemma 2.31 *The definitions of $\bar{\mathbf{r}}_j$ and \mathbf{q}_j imply $\bar{\mathbf{r}}_k^H \mathbf{r}_j = 0$ for $k > j$.*

Proof. We prove the statement by induction with respect to $d := k - j$.

Initialization For $d = 1$, $\bar{\mathbf{r}}_{j+1}^H \mathbf{r}_j = 0$ because ω_j has been chosen to ensure that.

Induction Step We suppose that $d > 1$. The induction hypothesis is $\bar{\mathbf{r}}_{j+m}^H \mathbf{r}_j = 0$ for $0 < m < d$. With the definition of $\bar{\mathbf{r}}_{j+d}$ we get

$$\bar{\mathbf{r}}_{j+d}^H \mathbf{r}_j = \bar{\mathbf{r}}_{j+d-1}^H \mathbf{r}_j - \omega_{j+d} \bar{\mathbf{q}}_{j+d}^H \mathbf{A} \mathbf{r}_j.$$

The first term on the right hand side expression vanishes because of the induction hypothesis. The second term can be reduced to zero by using the definition of $\bar{\mathbf{q}}_{j+1}$.

$$\bar{\mathbf{q}}_{j+d}^H \mathbf{A} \mathbf{r}_j = \bar{\mathbf{q}}_{j+d}^H \mathbf{A} \mathbf{q}_{j+1} + \sum_{i=1}^j \psi_{i,j} \bar{\mathbf{q}}_{j+d}^H \mathbf{A} \mathbf{q}_i$$

Since $j + d - 1 > j$ all the terms vanish, making $\bar{\mathbf{q}}_{j+d}^H \mathbf{A} \mathbf{r}_j = 0$ and hence $\bar{\mathbf{r}}_{j+d}^H \mathbf{r}_j = 0$.

□

The definitions of \mathbf{r}_j and $\bar{\mathbf{r}}_j$ imply that the sum necessary so far to evaluate \mathbf{q}_j and $\bar{\mathbf{q}}_j$ reduces to only two terms.

$$\bar{\mathbf{q}}_k^H \mathbf{A} \mathbf{r}_j = \frac{1}{\omega_{k-1}} (\bar{\mathbf{r}}_{k-1}^H \mathbf{r}_j - \bar{\mathbf{r}}_k^H \mathbf{r}_j) = \begin{cases} 0 & k < j \\ -\frac{1}{\omega_{j-1}} \bar{\mathbf{r}}_j^H \mathbf{r}_j & k = j \end{cases},$$

$$\bar{\mathbf{r}}_j^H \mathbf{A} \mathbf{q}_k = \frac{1}{\omega_{k-1}} (\bar{\mathbf{r}}_j^H \mathbf{r}_{k-1} - \bar{\mathbf{r}}_j^H \mathbf{r}_k) = \begin{cases} 0 & k < j \\ -\frac{1}{\omega_{j-1}} \bar{\mathbf{r}}_j^H \mathbf{r}_j & k = j \end{cases}.$$

This gives rise to the following simplification

$$\psi_j := -\psi_{j,j} = -\bar{\psi}_{j,j} = \frac{1}{\omega_{j-1}} \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{q}}_j^H \mathbf{A} \mathbf{q}_j} = \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{r}}_{j-1}^H \mathbf{r}_{j-1}}.$$

Let us sum up all these simplifications into the following scheme.

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A} \mathbf{s}_0$	$\bar{\mathbf{r}}_0 := \mathbf{b} - \mathbf{A} \mathbf{s}_0$
$\mathbf{q}_1 := \mathbf{r}_0$	$\bar{\mathbf{q}}_1 := \bar{\mathbf{r}}_0$
$\mathbf{r}_j := \mathbf{r}_{j-1} - \omega_{j-1} \mathbf{A} \mathbf{q}_j$	$\bar{\mathbf{r}}_j := \bar{\mathbf{r}}_{j-1} - \omega_{j-1} \mathbf{A} \bar{\mathbf{q}}_j$
$\mathbf{q}_{j+1} := \mathbf{r}_j + \psi_j \mathbf{q}_j$	$\bar{\mathbf{q}}_{j+1} := \bar{\mathbf{r}}_j + \psi_j \bar{\mathbf{q}}_j$
with $\psi_j := \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{r}}_{j-1}^H \mathbf{r}_{j-1}}$ and $\omega_j := \frac{\bar{\mathbf{r}}_j^H \mathbf{r}_j}{\bar{\mathbf{q}}_{j+1}^H \mathbf{q}_{j+1}}$	

The common BiCG function is directly derived from this scheme. It is presented in Algorithm 2.10.

Quasi Minimum Residual Method (QMR) The Quasi Minimum Residual Method (QMR) combines the MR approach of the projection methods with Lanczos biorthog-

Algorithm 2.10 Common BiCG algorithm.

```

function BiCG( $\mathbf{A}$ ,  $\mathbf{b}$ ,  $\mathbf{s}_0$ ,  $\varepsilon$ ):vector;
   $\mathbf{r}_0 := \bar{\mathbf{r}}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$ ;
   $\mathbf{q}_0 := \bar{\mathbf{q}}_0 := \mathbf{0}$ ;
   $\psi_0 := 0$ ;
   $\phi_0 := \bar{\mathbf{r}}_0^H \mathbf{r}_0$ ;
   $j := 0$ ;
  while  $\|\mathbf{r}_j\| > \varepsilon \|\mathbf{b}\|$  do
     $\mathbf{q}_{j+1} := \mathbf{r}_j + \psi_j \mathbf{q}_j$ ;  $\bar{\mathbf{q}}_{j+1} := \bar{\mathbf{r}}_j + \psi_j \bar{\mathbf{q}}_j$ ;
     $\mathbf{t}_{j+1} := \mathbf{A}\mathbf{q}_{j+1}$ ;  $\bar{\mathbf{t}}_{j+1} := \mathbf{A}^H \bar{\mathbf{q}}_{j+1}$ ;
     $\omega_j := \phi_j / (\bar{\mathbf{q}}_{j+1}^H \mathbf{t}_{j+1})$ ;
     $\mathbf{s}_{j+1} := \mathbf{s}_j + \omega_j \mathbf{q}_{j+1}$ ;
     $\mathbf{r}_{j+1} := \mathbf{r}_j - \omega_j \mathbf{t}_{j+1}$ ;  $\bar{\mathbf{r}}_{j+1} := \bar{\mathbf{r}}_j - \omega_j \bar{\mathbf{t}}_{j+1}$ ;
     $\phi_{j+1} := \bar{\mathbf{r}}_{j+1}^H \mathbf{r}_{j+1}$ ;
     $\psi_{j+1} := \phi_{j+1} / \phi_j$ ;
     $j := j + 1$ 
  end
  BiCG :=  $\mathbf{s}_{j-1}$ 
end BiCG;

```

onalization. As in GMRES the residual in step j is obtained by

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_j \mathbf{z}_j = \mathbf{r}_0 - \mathbf{V}_{j+1} \bar{\mathbf{H}}_j \mathbf{z}_j = \mathbf{V}_{j+1} (\mathbf{e}_1 \|\mathbf{r}_0\| - \bar{\mathbf{H}}_j \mathbf{z}_j).$$

However, \mathbf{V}_{j+1} is not an orthogonal matrix anymore. And so this matrix cannot be dropped, when $\|\mathbf{r}_j\|$ has to be evaluated. This is why in QMR $\|\mathbf{r}_j\|$ is not minimized directly. Instead only the norm of its component vector in $\mathcal{K}_{j+1}(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{V}_{j+1}\}$ is minimized, hence the name *Quasi* minimum residual method.

Since \mathbf{W}_{j+1} is orthogonal to \mathbf{V}_{j+1} this norm can be expressed by

$$\|\mathbf{W}_{j+1}^H \mathbf{r}_j\| = \|\mathbf{e}_1\|\mathbf{r}_0\| - \tilde{\mathbf{H}}_j \mathbf{z}_j\|.$$

Thus in every step of QMR the residual \mathbf{r}_j is determined such that it is orthogonal to $\text{span}\{\mathbf{W}_{j+1}\tilde{\mathbf{H}}_j\} = \text{span}\{\mathbf{W}_{j+1}\mathbf{W}_{j+1}^H \mathbf{A}\mathbf{V}_j\}$. Concluding all the statements made so far, we can describe QMR as follows.

Definition 2.32 *The Quasi Minimum Residual Method (QMR) is a Krylov space method, based on Lanczos bi-orthogonalization with projection space*

$$\mathcal{P}_j = \text{span}\{\mathbf{W}_{j+1}\mathbf{W}_{j+1}^H \mathbf{A}\mathbf{V}_j\}.$$

The minimization algorithm follows the lines of GMRES. However, due to the special structure of $\tilde{\mathbf{H}}_j$ it can be simplified in order to perform a constant number of operations in each step of the iteration.

Let us denote $\tilde{\mathbf{H}}_j$ as

$$\tilde{\mathbf{H}}_j = \begin{bmatrix} \beta_1 & \gamma_1 & & & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & & & \\ & \alpha_3 & \beta_3 & \gamma_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha_{j-1} & \beta_{j-1} & \gamma_{j-1} & \\ & & & & \alpha_j & \beta_j & \\ & & & & & \alpha_{j+1} & \end{bmatrix}$$

Like GMRES, QMR performs a QR-decomposition of $\tilde{\mathbf{H}}_j$ in each step that uses as much information from the previous step as possible. Therefore, let the definitions (2.31) and (2.32) also be applied in this context.

OBTAINING THE QR-DECOMP.

Due to its structure a column of $\tilde{\mathbf{H}}_j$ is affected by only three consequent Givens transformations. Hence the upper triangular matrix \mathbf{R}_j has the following structure.

$$\mathbf{R}_j = \begin{bmatrix}
 \hat{\beta}_1 & \hat{\gamma}_1 & \hat{\delta}_1 & & & & \\
 & \hat{\beta}_2 & \hat{\gamma}_2 & \hat{\delta}_2 & & & \\
 & & \hat{\beta}_3 & \hat{\gamma}_3 & \hat{\delta}_3 & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \hat{\beta}_{j-2} & \hat{\gamma}_{j-2} & \hat{\delta}_{j-2} \\
 & & & & & \hat{\beta}_{j-1} & \hat{\gamma}_{j-1} \\
 & & & & & & \hat{\beta}_j \\
 & & & & & & & 0
 \end{bmatrix}$$

The Givens matrices accordingly resized to 4×4 the following relation holds between the columns of $\tilde{\mathbf{H}}_j$ and \mathbf{R}_j .

$$\begin{bmatrix}
 \hat{\delta}_{i-2} \\
 \hat{\gamma}_{i-1} \\
 \hat{\beta}_i \\
 0
 \end{bmatrix}
 = \mathbf{G}(i)\mathbf{G}(i-1)\mathbf{G}(i-2)
 \begin{bmatrix}
 0 \\
 \gamma_{i-1} \\
 \beta_i \\
 \alpha_{i+1}
 \end{bmatrix}$$

The components of the right hand side vector $\mathbf{Q}^H \mathbf{e}_1 \| \mathbf{r}_0 \|$ are obtained recursively the same way as in GMRES, described in equations (2.33) and (2.34).

UPDATING THE SOLUTION VECTOR

The approximation of the solution in step j is determined by

$$\mathbf{s}_j = \mathbf{s}_0 + \mathbf{V}_j \mathbf{z}_j = \mathbf{s}_0 + \mathbf{V}_j \mathbf{R}_j^{-1} \mathbf{g}_j.$$

Let $\mathbf{P}_j := \mathbf{V}_j \mathbf{R}_j^{-1}$. Like in the derivation of BiCG we can evaluate $\mathbf{V}_j = \mathbf{P}_j \mathbf{R}_j$ to see that $\mathbf{v}_j = \mathbf{p}_j \hat{\beta}_j + \mathbf{p}_{j-1} \hat{\gamma}_{j-1} + \mathbf{p}_{j-2} \hat{\delta}_{j-2}$. Thus \mathbf{p}_j can be generated recursively

$$\begin{aligned}
 \mathbf{p}_1 &:= \mathbf{v}_1 / \hat{\beta}_1 \\
 \mathbf{p}_{j+1} &:= (\mathbf{v}_{j+1} - [\mathbf{p}_{j-1}, \mathbf{p}_j] [\hat{\delta}_{j-1}, \hat{\gamma}_j]^T) / \hat{\beta}_{j+1}
 \end{aligned}$$

Since \mathbf{g}_j only changes in its last component, \mathbf{s}_j can be obtained by updating \mathbf{s}_{j-1} with $\mathbf{p}_j \mathbf{g}_j^{(j)}$.

The Lanczos procedure, the recursive columnwise QR-decomposition, the recursive computation of the right hand side and the update of the solution are combined to the QMR function given in Algorithm 2.11. To improve the readability of the algorithm we introduced vectors $\mathbf{h}_j = [\gamma_{j-1}, \beta_j]^\top$, $\mathbf{h}_j = [\alpha_j, \beta_j]^\top$, and $\boldsymbol{\rho}_j = [\hat{\delta}_{j-2}, \hat{\gamma}_{j-1}]^\top$ to denote pieces of columns of the matrices $\bar{\mathbf{H}}$, $\bar{\mathbf{K}}$ and \mathbf{R} .

Algorithm 2.11 QMR algorithm.

function QMR(A, b, s₀, ε):**vector**;

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{s}_0$; $\rho_1 := \|\mathbf{r}_0\|$; $j := 1$;

$\alpha_1 := 0$; $\gamma_0 := 0$; $\cos_{-1} := \cos_0 := 1$; $\sin_{-1} := \sin_0 := 0$;

$\mathbf{p}_{-1} := \mathbf{p}_0 := \mathbf{0}$; $\mathbf{v}_0 := \mathbf{w}_0 := \mathbf{0}$; $\mathbf{v}_1 := \mathbf{w}_1 := \mathbf{r}_0/\rho_1$;

while $|\rho_j| > \varepsilon\|\mathbf{b}\|$ **do**

{*Lanczos Biorthogonalization*}

$\mathbf{t}_{j+1} := \mathbf{A}\mathbf{v}_j$; $\mathbf{u}_{j+1} := \mathbf{A}^H\mathbf{w}_j$;

$\beta_j := \mathbf{w}_j^H\mathbf{t}_{j+1}$;

$\mathbf{h}_j := [\gamma_{j-1}, \beta_j]^T$; $\mathbf{k}_j := [\alpha_j, \beta_j]^T$;

$\mathbf{t}_{j+1} := \mathbf{t}_{j+1} - [\mathbf{v}_{j-1}, \mathbf{v}_j]\mathbf{h}_j$;

$\mathbf{u}_{j+1} := \mathbf{u}_{j+1} - [\mathbf{w}_{j-1}, \mathbf{w}_j]\mathbf{k}_j$;

$\sigma_{j+1} := \mathbf{t}_{j+1}^H\mathbf{u}_{j+1}$;

$\alpha_{j+1} := \sqrt{|\sigma_{j+1}|}$; $\gamma_j := \sigma_{j+1}/\alpha_{j+1}$;

$\mathbf{v}_{j+1} := \mathbf{t}_{j+1}/\alpha_{j+1}$; $\mathbf{w}_{j+1} := \mathbf{u}_{j+1}/\gamma_j$;

{*Givens Rotations*}

$\rho_j := \begin{bmatrix} \sin_{j-2} & 0 \\ \cos_{j-2} \cos_{j-1} & \sin_{j-1} \end{bmatrix} \mathbf{h}_j$;

$\hat{\beta}_j := [-\cos_{j-2} \sin_{j-1}, \cos_{j-1}] \mathbf{h}_j$;

$\tan := \alpha_{j+1}/\hat{\beta}_j$;

$\cos_j := 1/\sqrt{1 + \tan^2}$; $\sin_j := \cos_j \tan$;

$\hat{\beta}_j := \hat{\beta}_j \cos_j + \alpha_{j+1} \sin_j$;

{*RHS, gamma, dir. vect. and sol. update*}

$g_j^{(j)} := \rho_j \cos_j$; $\rho_{j+1} := -\rho_j \sin_j$;

$\mathbf{p}_j := (\mathbf{v}_j - [\mathbf{p}_{j-2}, \mathbf{p}_{j-1}] \rho_j) / \hat{\beta}_j$;

$\mathbf{s}_j := \mathbf{s}_j + \mathbf{p}_j g_j^{(j)}$;

$j := j + 1$

end;


QMR := \mathbf{s}_{j-1} ;

end QMR;

Chapter 3

Vector Extrapolation Methods

3.1 Introduction

ector extrapolation methods are the key technique to develop nonlinear Krylov space methods. They are capable to accelerate the convergence of any given vector sequence $\{\mathbf{x}_i\}$, $i = 0, 1, 2, \dots, j$ by extrapolating a limit or an anti-limit respectively. Especially they are able to accelerate linearly generated vector sequences. This way they can help solving a linear system more quickly. In this case it is known that they are mathematically equivalent to preconditioned Krylov space methods. On the other hand they are also able to accelerate nonlinear vector sequences.

In this chapter we give a survey of the most important vector extrapolation methods. We show their connection to linear Krylov space methods. Also a large part of this chapter deals with their ability to accelerate nonlinear vector sequences. We motivate the nonlinear behavior

by first studying scalar sequences and then transforming our results to the multivariate case.

3.2 General Theory

GIVEN A VECTOR SEQUENCE ... Let us consider a sequence of vectors $\{x_i\}$, $i = 0, 1, 2, \dots, j$. We suppose that this sequence is generated by a linear process

$$x_{i+1} := g(x_i) := Gx_i + d. \quad (3.1)$$

The vector sequence may not necessarily converge. But we assume that G has no eigenvalue 1. This condition implies that Iteration (3.1) has a *unique* fixed point s with $s = Gs + d$, which is the solution of the linear system

$$As = d \quad \text{with} \quad A := 1 - G. \quad (3.2)$$

COMMUTATION PROPERTY OF A AND G Because of their close relationship, the iteration matrix G and the system matrix A commute with each other with respect to matrix multiplication.

$$AG = (1 - G)G = G(1 - G) = GA. \quad (3.3)$$

AIM OF VECTOR EXTRAPOLATION METHODS

Definition 3.1 Given a sequence of vectors $\{x_i\}$, $i = 0, 1, 2, \dots, j$, with the property that the unknown operator that generates the vector sequence has a unique fixed point s on a given set \mathcal{S} .

A vector extrapolation method *extrapolates approximations* $s_j \in \mathcal{S}$ of s using the known elements of the given vector sequence.

By Definition 3.1 vector extrapolation methods approximate s . For this purpose the distance ε_i between s and

an element of the given vector sequence \mathbf{x}_i is an important term. We refer to this value as the *error of a vector sequence element*. It is defined by

$$\varepsilon_i := \boldsymbol{\eta}(\mathbf{x}_i) = \mathbf{s} - \mathbf{x}_i.$$

If the vector sequence is generated by a linear iteration (3.1), the error in step i can easily be connected to the initial error

$$\varepsilon_i = \mathbf{s} - \mathbf{x}_i = \mathbf{G}\mathbf{s} + \mathbf{d} - \mathbf{G}\mathbf{x}_{i-1} - \mathbf{d} = \mathbf{G}\varepsilon_{i-1} = \mathbf{G}^i\varepsilon_0.$$

With this observation \mathbf{x}_i can be rewritten as $\mathbf{x}_i = \mathbf{s} - \mathbf{G}^i\varepsilon_0$. Thus a linear combination of the sequence elements \mathbf{x}_i results in

$$\sum_{i=0}^d \beta_i \mathbf{x}_i = \sum_{i=0}^d \beta_i (\mathbf{s} - \mathbf{G}^i \varepsilon_0) = \mathbf{s} \sum_{i=0}^d \beta_i - \sum_{i=0}^d \beta_i \mathbf{G}^i \varepsilon_0. \quad (3.4)$$

This leads to the basic idea of vector extrapolation methods: Find a linear combination of certain elements of the vector sequence such that the error terms are eliminated. That is: find $\beta_i, i = 0 \dots d$ such that

$$\sum_{i=0}^d \beta_i = 1 \quad \text{and} \quad \sum_{i=0}^d \beta_i \mathbf{G}^i \varepsilon_0 = 0. \quad (3.5)$$

This way the linear combination of the sequence elements (3.4) reduces to \mathbf{s} . Thus the problem of vector extrapolation can be reduced to finding the minimal polynomial of \mathbf{G} with respect to the initial error ε_0 . If the sum of the coefficients of the minimal polynomial $\beta_i, i = 0 \dots d$ is normalized to 1, the extrapolated limit (or anti-limit) \mathbf{s} is given by

$$\mathbf{s} = \sum_{i=0}^d \beta_i \mathbf{x}_i.$$

ELIMINATING
THE UNKNOWN
INITIAL ERROR

There are two problems in the proposed procedure: *First* we do not know ε_0 , *secondly* we do not know \mathbf{G} . Both problems can be solved with the help of the difference vectors $\delta\mathbf{x}_i := \mathbf{g}(\mathbf{x}_i) - \mathbf{x}_i$ between subsequent elements of the given vector sequence.

One notable property of the difference vectors is their equivalence to the residual vectors of the sequence elements $\boldsymbol{\rho}_i := \mathbf{r}(\mathbf{A}, \mathbf{d}, \mathbf{x}_i)$,

$$\delta\mathbf{x}_i = \mathbf{G}\mathbf{x}_i + \mathbf{d} - \mathbf{x}_i = \mathbf{d} - \mathbf{A}\mathbf{x}_i = \boldsymbol{\rho}_i. \quad (3.6)$$

This equivalence implies another important property of the difference vectors — their connection to the error,

$$\delta\mathbf{x}_i = \mathbf{d} - \mathbf{A}\mathbf{x}_i = \mathbf{A}\mathbf{s} - \mathbf{A}\mathbf{x}_i = \mathbf{A}\varepsilon_i. \quad (3.7)$$

This last property combined with Property (3.3) gives rise to the following theorem.

Theorem 3.2 *The minimal polynomial of \mathbf{G} with respect to ε_i is also the minimal polynomial of \mathbf{G} with respect to $\delta\mathbf{x}_i$.*

Proof. From (3.7) we know $\delta\mathbf{x}_i := \mathbf{A}\varepsilon_i$. Let μ_c be the minimal polynomial of degree c of \mathbf{G} with respect to $\delta\mathbf{x}_i$. Likewise let ν_d be the minimal polynomial of degree d of \mathbf{G} with respect to ε_i . Using the commutation property (3.3) of \mathbf{G} and \mathbf{A} we have on the one hand

$$0 = \mu_c(\mathbf{G})\delta\mathbf{x}_i = \mathbf{A}\mu_c(\mathbf{G})\varepsilon_i.$$

Since \mathbf{A} is nonsingular, $\mu_c(\mathbf{G})\varepsilon_i$ must be zero. Consequently ν_d must divide μ_c .

On the other hand

$$\nu_d(\mathbf{G})\varepsilon_i = \mathbf{0} = \mathbf{A}\nu_d(\mathbf{G})\varepsilon_i = \nu_d(\mathbf{G})\delta\mathbf{x}_i.$$

Therefore μ_c divides ν_d and hence $\mu_c = \nu_d$. \square

Theorem 3.2 tells us that finding the minimal polynomial of \mathbf{G} with respect to the unknown initial error is equivalent to finding the minimal polynomial of \mathbf{G} with respect to the known initial difference vector. Therefore the above mentioned problem is solved by taking differences.

Also the unknown \mathbf{G} can be eliminated from the proposed method by using the difference vectors. Note the following property. ELIMINATING
THE ITERATION
MATRIX

$$\begin{aligned}\delta\mathbf{x}_i &= \mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{G}\mathbf{x}_i + \mathbf{d} - \mathbf{G}\mathbf{x}_{i-1} - \mathbf{d} \\ &= \mathbf{G}\delta\mathbf{x}_{i-1} = \mathbf{G}^i\delta\mathbf{x}_0.\end{aligned}\quad (3.8)$$

Before we proceed, let us first introduce the notation

$$\mathbf{g}^i(\mathbf{x}) := \underbrace{\mathbf{g}(\dots(\mathbf{g}(\mathbf{x}))\dots)}_{i \text{ times}}, \quad (3.9)$$

i.e. the i times repeated application of the operator \mathbf{g} to an argument \mathbf{x} is abbreviated by the notation $\mathbf{g}^i(\mathbf{x})$.

Let us define a matrix operator

$$\begin{aligned}\mathbf{X}_d &: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times (d+1)}, \\ \mathbf{x} &\mapsto \mathbf{X}_d(\mathbf{x}) := [\mathbf{x}, \mathbf{g}(\mathbf{x}), \dots, \mathbf{g}^d(\mathbf{x})],\end{aligned}\quad (3.10)$$

This notation allows us to gather consecutive iterates into a matrix, e.g. $\mathbf{X}_d(\mathbf{x}_0) = [\mathbf{x}_0, \dots, \mathbf{x}_d]$.

Let us define a second matrix operator

$$\begin{aligned}\Delta_d &: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times (d+1)}, \\ \mathbf{x} &\mapsto \Delta_d(\mathbf{x}) := \mathbf{X}_d(\mathbf{g}(\mathbf{x})) - \mathbf{X}_d(\mathbf{x}).\end{aligned}$$

This notation allows us to gather consecutive differences into a matrix, e.g. $\Delta_d(\mathbf{x}_0) = [\delta\mathbf{x}_0, \dots, \delta\mathbf{x}_d]$.

Let $\mathbf{b} := [\beta_0, \dots, \beta_d]^\top$. Then, using (3.8) vector extrapolation methods can be reformulated in the following way:

Definition 3.3 A vector extrapolation method constructs $\mathbf{b} \in \mathbb{R}^{d+1}$ with the smallest possible $d \in \mathbb{N}$, $d \geq 0$, such that

$$\begin{bmatrix} 1 & \dots & 1 \\ \Delta_d(\mathbf{x}_0) \end{bmatrix} \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (3.11)$$

The extrapolated solution \mathbf{s} is given by

$$\mathbf{s} = \mathbf{X}_d(\mathbf{x}_0)\mathbf{b}.$$

In order to find \mathbf{b} and d the methods start with the smallest possible degree $j = 0$. In each step j is raised by one, and one more vector of the sequence \mathbf{x}_j is taken into account. However, system (3.11) cannot be solved exactly for $j < d$ and has to be replaced by the following formulation

$$\Delta_j(\mathbf{x}_0)\mathbf{b}_j \approx 0 \quad \text{s.t.} \quad [1 \quad \dots \quad 1] \mathbf{b}_j = 1, \quad (3.12)$$

with $\Delta_d(\mathbf{x}_0) = [\delta\mathbf{x}_0, \dots, \delta\mathbf{x}_d]$ and $\mathbf{b}_j := [\beta_{0,j}, \dots, \beta_{j,j}]^\top$ containing the coefficients of the extrapolating polynomial to determine the approximate solution \mathbf{s}_j , given by

$$\mathbf{s}_j := \mathbf{X}_j(\mathbf{x}_0)\mathbf{b}_j = \sum_{i=0}^j \beta_{i,j} \mathbf{x}_i. \quad (3.13)$$

Associated with the approximate solution is an error defined by

$$\boldsymbol{\eta}_j := \mathbf{s} - \mathbf{s}_j.$$

We define the residual \mathbf{r}_j of the approximate limit (or anti-limit) as

$$\mathbf{r}_j := \mathbf{r}(\mathbf{A}, \mathbf{d}, \mathbf{s}_j) = \mathbf{d} - \mathbf{A}\mathbf{s}_j$$

By (3.13) the residual turns into

$$\mathbf{r}_j = \mathbf{d} - \mathbf{A} \sum_{i=0}^j \beta_{i,j} \mathbf{x}_i = \sum_{i=0}^j \beta_{i,j} (\mathbf{d} - \mathbf{A}\mathbf{x}_i),$$

since the sum of the coefficients $\beta_{i,j}$ is 1. The bracketed expression is the residual of the sequence elements $\boldsymbol{\rho}_i$, which by (3.6) is equivalent to the difference vector $\delta\mathbf{x}_i$. Thus the residual of the approximate solution turns finally into

$$\mathbf{r}_j = \sum_{i=0}^j \beta_{i,j} \delta\mathbf{x}_i = \boldsymbol{\Delta}_j(\mathbf{x}_0) \mathbf{b}_j. \quad (3.14)$$

If $j = 0$ the best approximation for \mathbf{s} is $\mathbf{s}_0 = \mathbf{x}_0$. Therefore $\mathbf{r}_0 = \boldsymbol{\rho}_0$.

If $j < d$ it is not clear how to determine the coefficients. Like in Chapter 2 this problem is solved by projection. Let \mathcal{P}_j be a space of dimension j and \mathbf{P}_j a matrix containing a base of \mathcal{P}_j . System (3.12) is solved by

HOW TO
SOLVE THE
APPROXIMATE
SYSTEM (3.12)

$$\begin{bmatrix} 1 & \dots & 1 \\ \mathbf{P}_j^H \boldsymbol{\Delta}_j(\mathbf{x}_0) \end{bmatrix} \mathbf{b}_j = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \quad (3.15)$$

Thus the coefficients $\beta_{i,j}, i = 0 \dots j$ are determined by projecting \mathbf{r}_j into \mathcal{P}_j^\perp subject to

$$\sum_{i=0}^j \beta_{i,j} = 1. \quad (3.16)$$

Lemma 3.4 *Vector extrapolation methods are correction space methods. In step j their correction space is the Krylov space $\mathcal{C}_j = \mathcal{K}_j(\mathbf{G}, \delta\mathbf{x}_0)$.*

Proof. Let us define $\gamma_{k,j} := \sum_{i=k+1}^j \beta_{i,j}$. The definition of the difference vectors implies $\mathbf{x}_i = \mathbf{x}_0 + \sum_{k=0}^{i-1} \delta\mathbf{x}_k$. Inserting both values into (3.13) yields in

$$\mathbf{s}_j = \mathbf{x}_0 + \sum_{i=1}^j \beta_{i,j} \sum_{k=0}^{i-1} \delta\mathbf{x}_k = \mathbf{x}_0 + \sum_{k=0}^{j-1} \gamma_{k,j} \mathbf{G}^k \delta\mathbf{x}_0. \quad (3.17)$$

This result can be written as $\mathbf{s}_j = \mathbf{x}_0 + \Delta_{j-1}(\mathbf{x}_0) \mathbf{c}_{j-1}$, with $\Delta_{j-1}(\mathbf{x}_0)$ being the basis of $\mathcal{K}_j(\mathbf{G}, \delta\mathbf{x}_0)$ and $\mathbf{c}_{j-1} := [\gamma_{0,j}, \dots, \gamma_{j-1,j}]^\top$. \square

As an immediate consequence of this lemma, we can formulate an expression for \mathbf{A}^{-1} . From Equation (3.7) we know that $\mathbf{A}^{-1} \delta\mathbf{x}_0 = \boldsymbol{\varepsilon}_0$. Therefore $\sum_{k=0}^{j-1} \gamma_{k,j} \mathbf{G}^k$ in Equation (3.17) is an approximation of \mathbf{A}^{-1} restricted to the span of $\delta\mathbf{x}_0$.

By the way, approximating the inverse of the system matrix of a linear system by a matrix polynomial is also the basic idea of polynomial preconditioning.

CONVER-
GENCE WITHIN
AT MOST n
STEPS

Lemma 3.5 *Let $\{\mathbf{x}_0, \mathbf{x}_1, \dots\}$ be a vector sequence generated by a linear process $\mathbf{x}_{i+1} = \mathbf{G}\mathbf{x}_i + \mathbf{d}$, where $\mathbf{G} \in \mathbb{R}^{n \times n}$.*

Vector extrapolation methods accelerating this vector sequence converge mathematically within at most n steps.

Proof. According to (3.5) vector extrapolation methods find the minimal polynomial of \mathbf{G} with respect to the initial error $\boldsymbol{\varepsilon}_0$. The Cayley-Hamilton theorem [19, p. 86] states that any square matrix is annihilated by its characteristic polynomial, that is $\chi(\mathbf{G}) = 0$, and consequently, $\chi(\mathbf{G})\boldsymbol{\varepsilon}_0 = 0$. The degree of the characteristic polynomial is at most the order of \mathbf{G} . Hence the degree of the minimal polynomial of \mathbf{G} with respect to $\boldsymbol{\varepsilon}_0$ can be at most n . \square

3.2.1 Mathematical Equivalence to Linear Krylov Space Methods

As shown in Section 2.4.1 by (2.15), Krylov space methods solve the linear system (3.2) by approximating the initial error η_0 by a polynomial in \mathbf{A} .

$$\eta_0 \approx \sum_{i=0}^j \alpha_{i,j} \mathbf{A}^i \mathbf{r}_0$$

Equation (3.17) implies that the error of the first vector sequence element $\varepsilon_0 \approx \mathbf{s}_{j+1} - \mathbf{x}_0 = \sum_{k=0}^j \gamma_{k,j} \mathbf{G}^k \delta \mathbf{x}_0$ is also approximated by a polynomial, but rather in \mathbf{G} than in \mathbf{A} . However the close relationship between the two matrices suggests that those methods are mathematically equivalent as shown by the following lemma.

Lemma 3.6 $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) = \mathcal{K}_n(\mathbf{G}, \delta \mathbf{x}_0)$.

Proof. Let η_0 be any element in $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$, i.e. there exist α_i , $i = 0 \dots n-1$ such that

$$\begin{aligned} \eta_0 &= \sum_{i=0}^{n-1} \alpha_i \mathbf{A}^i \mathbf{r}_0 = \sum_{i=0}^{n-1} \alpha_i (1 - \mathbf{G})^i \delta \mathbf{x}_0 = \\ &= \sum_{i=0}^{n-1} \alpha_i \sum_{k=0}^i (-1)^k \binom{i}{k} \mathbf{G}^k \delta \mathbf{x}_0 = \sum_{k=0}^{n-1} (-1)^k \sum_{i=k}^j \binom{i}{k} \alpha_i \mathbf{G}^k \delta \mathbf{x}_0. \end{aligned}$$

Thus η_0 is also an element of $\mathcal{K}_n(\mathbf{G}, \delta \mathbf{x}_0)$. On the other hand let ε_0 be any element in $\mathcal{K}_n(\mathbf{G}, \delta \mathbf{x}_0)$, i.e. there exist γ_k , $k = 0 \dots n-1$ such that

$$\begin{aligned}\varepsilon_0 &= \sum_{k=0}^{n-1} \gamma_k \mathbf{G}^k \delta \mathbf{x}_0 = \sum_{k=0}^{n-1} \gamma_k (\mathbf{I} - \mathbf{A})^k \mathbf{r}_0 = \\ &= \sum_{k=0}^{n-1} \gamma_k \sum_{i=0}^k (-1)^i \binom{k}{i} \mathbf{A}^i \mathbf{r}_0 = \sum_{i=0}^{n-1} (-1)^i \sum_{k=i}^{n-1} \binom{k}{i} \gamma_k \mathbf{A}^i \mathbf{r}_0,\end{aligned}$$

which shows that ε_0 is an element of $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. Since both vectors $\boldsymbol{\eta}_0$ and ε_0 were chosen arbitrarily, $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ must be equivalent to $\mathcal{K}_n(\mathbf{G}, \delta \mathbf{x}_0)$. \square

We have already seen that vector extrapolation methods are projection methods using $\mathcal{C}_j = \mathcal{K}_j(\mathbf{G}, \delta \mathbf{x}_0)$ (3.4), which by the last theorem immediately leads to the following corollary.

Corollary 3.7 *Vector extrapolation methods are Krylov space methods.* \square

The proof of Lemma 3.6 gives also a relation between the coefficients of the minimal polynomial determined by a vector extrapolation method and the coefficients of the minimal polynomial determined by a Krylov space method,

$$\alpha_{i,j} = (-1)^i \sum_{k=i}^j \binom{k}{i} \gamma_{k,j} \quad \text{and} \quad \gamma_{k,j} = (-1)^k \sum_{i=k}^j \binom{i}{k} \alpha_{i,j}.$$

The binomial coefficients in these formulas indicate one reason for the worse numerical behavior of the vector extrapolation methods compared to Krylov space methods, when they are applied to solve linear systems of equations.

3.2.2 Choosing the Projection Space \mathcal{P}_j

Since vector extrapolation methods are correction space methods and their coefficients are determined by projection, they are projection methods according to Definition 2.3. By Lemma 2.10 we know that the choice for the projection space \mathcal{P}_j can therefore be restricted to $\mathcal{K}_j(\mathbf{G}, \delta\mathbf{x}_0) = \text{span}\{\Delta_{j-1}(\mathbf{x}_0)\}$ for a positive definite matrix \mathbf{A} and to $\text{span}\{\mathbf{A}\Delta_{j-1}(\mathbf{x}_0)\}$ for a nonsingular matrix \mathbf{A} . The latter base can easily be obtained by computing second order differences.

Let us define second order difference vectors,

$$\delta^2\mathbf{x}_j := \delta\mathbf{g}(\mathbf{x}_j) - \delta\mathbf{x}_j,$$

and the corresponding matrix operator

$$\begin{aligned} \Delta_d^2 : \mathbb{R}^n &\rightarrow \mathbb{R}^{n \times (d+1)}, \\ \mathbf{x} &\mapsto \Delta_d^2(\mathbf{x}) := \Delta(\mathbf{g}(\mathbf{x})) - \Delta(\mathbf{x}). \end{aligned} \quad (3.18)$$

Then we have

$$\begin{aligned} \mathbf{A}\Delta_j(\mathbf{x}_0) &= [(1 - \mathbf{G})\delta\mathbf{x}_0, \dots, (1 - \mathbf{G})\delta\mathbf{x}_j] \\ &= [\delta\mathbf{x}_0 - \delta\mathbf{x}_1, \dots, \delta\mathbf{x}_j - \delta\mathbf{x}_{j+1}] = -\Delta_j^2(\mathbf{x}_0). \end{aligned} \quad (3.19)$$

Although it is sufficient to choose the two spaces mentioned above as projection spaces, we have already seen in the chapter about linear Krylov space methods that from an algorithmic point of view it makes also sense to consider the space $\mathcal{K}_j(\mathbf{A}^H, \mathbf{r}_0)$ which is by Lemma 3.6 equivalent to $\mathcal{K}_j(\mathbf{G}^H, \delta\mathbf{x}_0)$. Indeed these considerations lead to another class of vector extrapolation algorithms, the *epsilon algorithms*.

3.2.3 $\Delta_j(x_0)$ Base Construction

As we have already mentioned in the last chapter $\Delta_j(x_0)$ is ill conditioned for large values j . To deal with this problem Sidi [37] suggests a successive QR decomposition of $\Delta_j(x_0)$ by a modified Gram-Schmidt process.

Let us denote the QR decomposition by $\Delta_j(x_0) = Q_j R_j$, with $Q_j = [q_0, \dots, q_j]$ and the components of R_j as $r_{i,k}$. An algorithm for obtaining this decomposition from a sequence of column vectors $\delta x_i, i = 0 \dots j$ is given in Algorithm 3.1.

Algorithm 3.1 Modified Gram-Schmidt QR decomposition of $\Delta_j(x_0)$.

```

function MGS( $\Delta_j(x_0), j$ ):[matrix,matrix];
  for  $k := 0$  to  $j$  do
     $q_k := \delta x_k$ ;
    for  $i := 0$  to  $k - 1$  do
       $r_{i,k} := q_i^H q_k$ ;
       $q_k := q_k - r_{i,k} q_i$ 
    end;
     $r_{k,k} := \|q_k\|$ ;
     $q_k := q_k / r_{k,k}$ 
  end;
  MGS := [ $Q_j, R_j$ ];
end MGS;

```

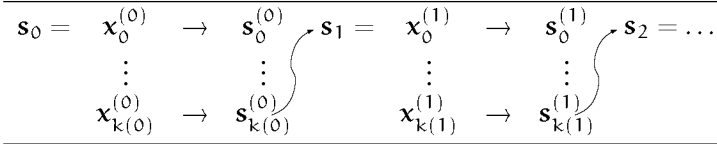
3.3 Nonlinear Behavior

In order to apply vector extrapolation methods to a nonlinear problem, we suppose that there exists already a nonlinear iterative method

$$x_{i+1} = g(x_i), \quad (3.20)$$

\mathbf{g} being a nonlinear vector valued operator. This operator has the property that the solution of the nonlinear problem \mathbf{s} is a fixed point of \mathbf{g} .

Figure 3.1 Scheme for applying a vector extrapolation method to a nonlinear problem.



A vector extrapolation method is applied as follows (see also Figure 3.1):

1. We start with $j = 0$ and an initial guess \mathbf{s}_0 which acts as a starting vector $\mathbf{x}_0^{(0)}$ for the sequence to be generated.
2. We use the existing iterative method (3.20) and $\mathbf{x}_0^{(j)}$ to generate a sequence of vectors $\mathbf{x}_0^{(j)}, \dots, \mathbf{x}_{k(j)}^{(j)}$.
3. A vector extrapolation method is used to extrapolate intermediate approximations $\mathbf{s}_0^{(j)}, \dots, \mathbf{s}_{k(j)}^{(j)}$ of \mathbf{s} from the given sequence.
4. The last intermediate approximation $\mathbf{s}_{j+1} := \mathbf{s}_{k(j)}^{(j)}$ is used as a new starting point $\mathbf{x}_0^{(j+1)}$ for a new sequence.
5. The procedure is repeated from step 2 until convergence.

Sidi and Brezinski use the term *cycling* to refer to such a scheme. In the context of linear iterative solvers such as FOM and GMRES such a scheme is described by the term *restart*.

Two other terms frequently used in this context are *inner* and *outer iteration*. The inner iteration uses the gen-

erated sequence elements $\mathbf{x}_0^{(j)}, \dots, \mathbf{x}_{k^{(j)}}^{(j)}$ to extrapolate intermediate approximations $\mathbf{s}_0^{(j)}, \dots, \mathbf{s}_{k^{(j)}}^{(j)}$. The outer iteration generates the iterates $\mathbf{s}_0, \mathbf{s}_1$, etc by the help of the inner iterations.

In this thesis we will prefer to use the term *restart*. To save memory, restarted schemes are also used for solving linear systems of equations.

3.3.1 One Dimensional Motivation: Steffensen's Method

In order to get an idea of what happens when we apply this procedure, let us first study the nonlinear behavior of extrapolation methods for one dimensional problems. In this case all the vectors become scalar values. The consequence is that there are no intermediate approximations. Assuming that we consider the step from j to $j + 1$ we simplify the notation by dropping the (j) and $(j+1)$ in the exponent of all values involved in the further analysis of the one dimensional case.

Nearly all vector extrapolation methods rely on the assumption that the sequence is produced by a linear iteration $x_{i+1} = cx_i + d$. This means that they determine the intersection between the bisecting line $y = x$ and the line $h : y = cx + d$ ¹. In the scalar case always the full "linear system" is solved. Therefore all of them generate the same approximate solution $s_{j+1} = \beta_{0,1}x_0^{(j)} + \beta_{1,1}x_1^{(j)}$ with

$$\beta_{0,1} = \frac{-\delta x_1}{\delta x_0 - \delta x_1} \quad \text{and} \quad \beta_{1,1} = \frac{\delta x_0}{\delta x_0 - \delta x_1}.$$

¹There is one algorithm that is not known to rely on a linear system of equations. This algorithm is called *vector epsilon algorithm (VEA)*. Since this algorithm does not fit into our framework, we won't deal with it.

Since the linear system to be solved is of full rank, the minimal polynomial vanishes,

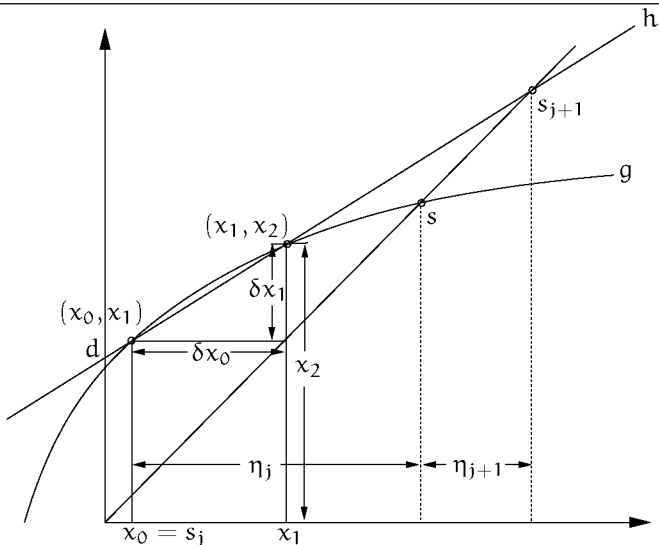
$$0 = \mu_1(c) = \beta_{1,1}c + \beta_{0,1} = \delta x_0 c - \delta x_1.$$

Hence the slope of h is

$$c = \delta x_1 / \delta x_0$$

and $d = x_1 - cx_0$. Thus line h is a secant through the points (x_0, x_1) and (x_1, x_2) as depicted in Figure 3.2.

Figure 3.2 Picture of values involved in a step from j to $j + 1$ of a one dimensional nonlinear extrapolation method. We see that the error value η_{j+1} of the extrapolated solution s_{j+1} is smaller than the error value η_j of the initial guess $x_0 = s_j$.



With these values the approximate solution s_{j+1} can be expressed after one step as

$$s_{j+1} = \frac{d}{1-c} = \frac{x_1 - cx_0}{1-c} = \frac{\delta x_0 x_1 - \delta x_1 x_0}{\delta x_0 - \delta x_1} \quad (3.21)$$

$$= s_j - \frac{\delta s_j}{\delta^2 s_j} \delta s_j =: b(s_j), \quad (3.22)$$

with $\delta s_j = g(s_j) - s_j$ and $\delta^2 s_j = \delta g(s_j) - \delta s_j$. The step from (3.21) to (3.22) is possible because $x_0^{(j)} = s_j$

Equation (3.22) identifies the extrapolation method for scalar sequences as *Aitken's Δ^2 method* [17, p. 72]. Aitken's method is a scheme for accelerating the convergence, given the discrete values of any sequence. In the context of accelerating a nonlinear iterative scheme $x_{i+1} = g(x_i)$ it is known as the *Diagonal Aitken Procedure* or *Steffensen's method* respectively [17, p. 90].

Steffensen's method rewritten in terms of g is

$$\begin{aligned} b(x) &:= x - \frac{(g(x) - x)^2}{g(g(x)) - 2g(x) + x} \\ &= \frac{xg(g(x)) - g(x)^2}{g(g(x)) - 2g(x) + x}. \end{aligned} \quad (3.23)$$

In order for Steffensen's method to work as desired, it is essential that both functionals g and b have exactly the same fixed points. This requirement is established by the following lemma.

Lemma 3.8 *Let g be a given scalar and differentiable functional.*

Functional b as defined by (3.23) has a fixed point at s if and only if g has a fixed point at s .

Proof. By (3.23) we have

$$b(x) = x - \frac{(g(x) - x)^2}{v(x)} \quad \text{with} \quad v(x) := g(g(x)) - 2g(x) + x. \quad (3.24)$$

The mean value theorem allows to reformulate $g(g(x))$ the following way

$$g(g(x)) = g(x) + g'(\xi)(g(x) - x),$$

with ξ lying between x and $g(x)$. This formulation of $g(g(x))$ turns the denominator of (3.24) into

$$v(x) = (g(x) - x)(g'(\xi) - 1),$$

and hence

$$b(x) = x - \frac{g(x) - x}{g'(\xi) - 1}.$$

We observe: For $x \rightarrow s$, ξ tends also to s . If $g'(s) \neq 1$ then s is a fixed point of b if and only if it is a fixed point of g .

Let now x^* be any argument for which $g'(x^*) = 1$ and let $c = g(x^*) - x^*$. $g(x) - x - c$ must have a multiple zero at x^* . In this case we can express g as

$$g(x) := c + x + f(x)(x - x^*)^k$$

with $k > 1$ and $f(x^*) \neq 0$. The derivative of g is

$$g'(x) = 1 + f'(x)(x - x^*)^k + kf(x)(x - x^*)^{k-1}.$$

Using these expressions for b we get

$$b(x) = x - \frac{c + f(x)(x - x^*)}{f'(x)(x - x^*) + kf(x)}.$$

Let now $s = x^*$. If $g(s) = s$, c must vanish. Since $f(s) \neq 0$, b has a fixed point at s . On the other hand if $g(s) \neq s$, $c \neq 0$ and b becomes

$$b(s) = s - \frac{c}{kf(s)} \neq s.$$

□

LOCAL
CONVERGENCE

Ostrowski [29] has intensively analyzed the local behavior of Steffensen's method. The following considerations are an extract of his results.

First we observe that Steffensen's method is invariant against translating g on the bisecting line. To show this, let us define a functional $\gamma(x) := g(x + \alpha) - \alpha$. Steffensen's method applied to this functional becomes

$$\begin{aligned} \beta(x) &= x - \frac{(\gamma(x) - x)^2}{\gamma(\gamma(x)) - 2\gamma(x) + x} \\ &= (x + \alpha) - \frac{(g(x + \alpha) - (x + \alpha))^2}{g(g(x + \alpha)) - 2g(x + \alpha) + (x + \alpha)} - \alpha, \end{aligned} \quad (3.25)$$

which is equal to $b(x + \alpha) - \alpha$. This is why we can restrict our study to the case $s = 0$. We assume that $g'(s)$ exists. However, we will still use the error $\eta := x - s$ in our considerations to ease the transition to the general case. Let us use the following ansatz for g in the vicinity of s ,

$$g(x) := \alpha\eta + R(x)\eta^\lambda, \quad (3.26)$$

with $\lambda > 1$ and $R(x)$ bounded for $x \rightarrow s$, i.e. $\lim_{x \rightarrow s} R(x) = \rho$, such that $g'(s) = \alpha$. Note that a particular choice for $R(x)$ would be

$$R(x) = \sum_{i=\lambda}^{\infty} \frac{g^{(i)}(s)}{i!} \eta^{i-\lambda}$$

if g has a Taylor expansion at s . In this case ρ becomes

$$\rho = \frac{g^{(\lambda)}(s)}{\lambda!}. \tag{3.27}$$

The ansatz for g with $s = 0$ turns b into

$$b(x) = \frac{-\alpha R(x)x\eta^\lambda + R(g(x))xg(x)^\lambda - R(x)^2\eta^{2\lambda}}{(\alpha - 1)^2x + (\alpha - 2)R(x)\eta^\lambda + R(g(x))g(x)^\lambda}. \tag{3.28}$$

If the first derivative of g at s is not equal to 1, b turns into SUPERLINEAR
(QUADRATIC)
CONVERGENCE

$$b(x) = \frac{O(\eta^{\lambda+1})}{O(\eta)} = O(\eta^\lambda).$$

This result shows that s is a point of attraction and the method converges with a superlinear rate. In particular if the Taylor expansion of g around s exists, the method of Steffensen converges at least quadratically.

Due to the geometrical construction it is easy to see that the method is not defined at s if $g'(s) = 1$. Nevertheless s is also a point of attraction in that case. To show this, the denominator of b has to be further examined. The ansatz of (3.26) already contains the derivative of g at s , which is $g'(s) = \alpha$. For the actual case we assume $\alpha = 1$. This choice of α turns g into LINEAR
CONVERGENCE

$$g(x) = \eta(1 + R(x)\eta^{\lambda-1})$$

and hence the denominator of b in (3.28) into

$$\begin{aligned} \nu(b(x)) &= -R(x)\eta^\lambda + R(g(x))g(x)^\lambda \\ &= \eta^\lambda (R(g(x))(1 + R(x)\eta^{\lambda-1})^\lambda - R(x)) \\ &= \eta^\lambda (R(g(x)) - R(x) + \lambda R(g(x))R(x)\eta^{\lambda-1} + O(\eta^{2\lambda-2})). \end{aligned}$$

For our purposes we now have to develop $R(g(x))$. Using (3.26) we get by the mean value theorem

$$\begin{aligned} R(g(x)) &= R(x) + R'(\xi)(g(x) - x) \\ &= R(x) + R'(\xi)R(x)\eta^\lambda \quad \text{with } x < \xi < g(x). \end{aligned}$$

Since $\xi \sim \eta$ for $x \rightarrow s$ we can estimate $R(g(x))$ as

$$R(g(x)) = R(x) + \eta^{\lambda-1} R(x) \frac{\eta}{\xi} \xi R'(\xi) = R(x) + \eta^{\lambda-1} O(\xi R'(\xi)).$$

Inserting this result into the last expression for $v(b(x))$ we get

$$v(b(x)) = \eta^\lambda (\lambda R(x)^2 \eta^{\lambda-1} + \eta^{\lambda-1} O(\xi R'(\xi)) + O(\eta^{2\lambda-2})).$$

If $\eta R'(x) \rightarrow 0$ for $x \rightarrow s$ we get

$$v(b(x)) = \lambda R(x)^2 \eta^{2\lambda-1} (1 + o(1)). \quad (3.29)$$

Using this result and (3.26) for the numerator of b in (3.25) we get for b in the vicinity of s

$$b(x) = x - \frac{R(x)^2 \eta^{2\lambda}}{\lambda R(x)^2 \eta^{2\lambda-1} (1 + o(1))} = x - \frac{\eta}{\lambda} + o(\eta).$$

Finally we are able to estimate the error of b in the vicinity of s

$$b(x) - s = (1 - \frac{1}{\lambda})\eta + o(\eta).$$

This result shows that s is a point of attraction in the case $\alpha = 1$ if $\eta R'(x) \rightarrow 0$ for $x \rightarrow s$. The convergence however, is linear with factor $1 - 1/\lambda$.

The latter condition $\eta R'(x) \rightarrow 0$ for $x \rightarrow s$ can be reformulated by considering g' in the vicinity of s

$$\begin{aligned} \lim_{x \rightarrow s} g'(x) &= \lim_{x \rightarrow s} \alpha + (R'(x)\eta + \lambda R(x))\eta^{\lambda-1} \\ &= \alpha + \lambda \rho \eta^{\lambda-1}. \end{aligned}$$

Example 3.9 To illustrate the quadratic convergence rate we compare a fixed point iteration,

$$x_{i+1} = g(x_i) := 3 \sin(x_i/3 - 0.66) + 2,$$

with the iteration accelerated by Steffensen’s method. We start both methods at $s_0 = 0$ and stop them if the relative error goes below 10^{-10} . The fixed point of g is approximately $s = 3.0079956588341082$, and $g'(s) = .9418625018648241$ indicating a slow convergence rate for the fixed point iteration.

Table 3.1 Relative errors of a scalar fixed point iteration and its acceleration by Steffensen’s method. The basic iteration $x_{i+1} = 3 \sin(x_i/3 - 0.66) + 2$ converges linearly, the accelerated method converges quadratically.

Step	Basic iteration	Steffensen
0	1.00000000000000	1.00000000000000
1	0.94659252794868	0.72466550539586
2	0.90354466621430	0.49575283569324
3	0.86765481338535	0.14411511166047
4	0.83697156346778	0.03533181499826
5	0.81022278285217	0.00298120684933
6	0.78653620118510	0.00002396860169
7	0.76528963796686	0.00000000156753
8	0.74602495796382	0.00000000000009
...	...	
447	0.00000000009691	

Table 3.1 shows the results of both methods. One should see from these results that the basic iteration has a slow linear convergence rate while Steffensen’s method converges quadratically.

Steffensen’s method has a singularity in s if $g'(s) = 1$. On computer systems with finite arithmetic the method will break down, before the solution is obtained precisely. The question arises: How close can the method get?

NUMERICAL
ANALYSIS OF
THE SINGULAR
CASE $g'(s) = 1$

The break down happens, when the denominator of b vanishes by cancellation, i.e. if x becomes numerically

of the same size as $2g(x) - g(g(x))$. Let ϵ be the machine epsilon. Then the following condition must hold in order for cancellation to happen,

$$|x|(1 + \epsilon) \geq |2g(x) - g(g(x))| \geq |x|(1 - \epsilon).$$

In other words, as long as the following condition holds there will be no cancellation

$$\begin{aligned} |x|\epsilon &< |2g(x) - g(g(x))| - |x| \\ &\leq |2g(x) - g(g(x)) - x| = |\nu(b(x))|. \end{aligned}$$

With $x \rightarrow s$ using (3.29) to estimate the denominator we get the following condition for the absolute error η ,

$$|s|\epsilon < \lambda\rho^2\eta^{2\lambda-1}.$$

Solving for the error we find

$$\eta > \sqrt[2\lambda-1]{\frac{|s|\epsilon}{\lambda\rho^2}}. \quad (3.30)$$

To interpret this inequality let us suppose the following situation. We want to find the approximate fixed point of g with $g'(s) = 1$. The result is required to meet a given tolerance. The inequality guarantees that Steffensen's method will deliver such a result without a break down if the right hand side of (3.30) is less than the requested tolerance. However, if we are lucky, we might get even closer to the solution. Therefore the right hand side of (3.30) is an *upper bound* for the achievable error of Steffensen's method in the case $g'(s) = 1$.

We can specify a more concrete upper bound if g has a Taylor expansion around s by using (3.27),

$$\eta > \sqrt[2\lambda-1]{(\lambda-1)!\lambda! \frac{|s|\epsilon}{g^{(\lambda)}(s)^2}}.$$

Due to the factorials we see that for higher values of λ Steffensen's method becomes quickly useless in the case $g'(s) = 1$.

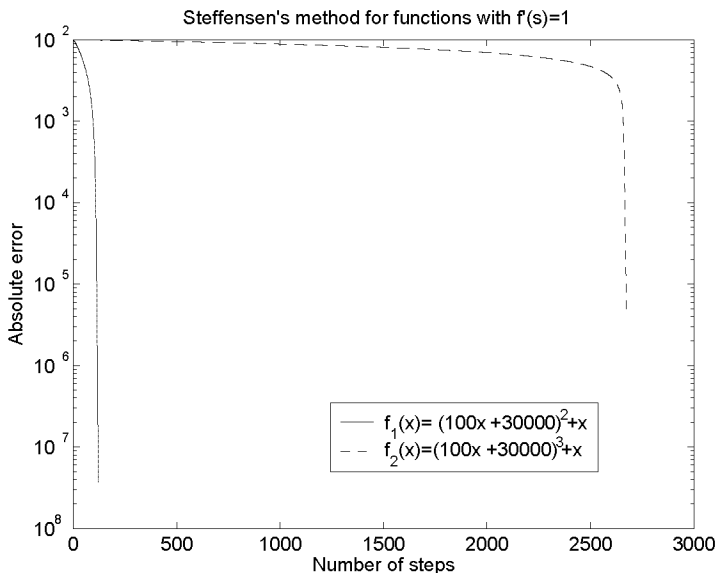
Example 3.10 Let us present an example for the linear convergence in the case of $g'(s) = 1$. We compare the performance of Steffensen's method for two functionals

$$g_1(x) := (100x - 3 \cdot 10^4)^2 + x = 300 + (x - 300) + 10^4(x - 300)^2,$$

$$g_2(x) := (100x - 3 \cdot 10^4)^3 + x = 300 + (x - 300) + 10^6(x - 300)^2.$$

Both functionals have the same fixed point $s = 300$. The difference between the two functionals is: g_1 has $\lambda = 2$ and $\rho = 10^4$, whereas g_2 has $\lambda = 3$ and $\rho = 10^6$. Another difference is that the fixed point iteration for g_2 is not able to converge to s , because $|g'(x)| \geq 1$ for every x .

Figure 3.3 Absolute error of Steffensen's method applied to two functionals with $g'(s) = 1$. The convergence rate is linear. The maximum achievable accuracy is dependent on the functional.



In Figure 3.3 the absolute errors of Steffensen's method for the functionals g_1 and g_2 are plotted. We let the method run until break down in MATLAB on a machine with IEEE arithmetic and its double precision machine epsilon $\epsilon = 2.220446049250313 \cdot 10^{-16}$. The starting value is chosen very near the solution, $s_0 = 300.01$, because the global convergence of the method is very slow for both functionals.

The method breaks down after having obtained the following absolute errors

$$g_1 : \eta_{123} = 3.691820893436670 \cdot 10^{-8},$$

$$g_2 : \eta_{2675} = 4.961853846907616 \cdot 10^{-6}.$$

The smallest absolute error guaranteed by (3.30) is

$$g_1 : \sqrt[3]{\frac{300\epsilon}{2 \cdot 10^8}} = 6.931764956787648 \cdot 10^{-8},$$

$$g_2 : \sqrt[5]{\frac{300\epsilon}{3 \cdot 10^{12}}} = 7.40095979741404 \cdot 10^{-6}.$$

Both error bounds match the effectively obtained error very well.

NONLOCAL
CONVER-
GENCE

In [17, pp. 93] Henrici has proven a non local convergence theorem for Steffensen's method. However, he only gives a proof for one case² out of actually six, two others³ are left for the students to prove. In Section 3.3.2 we provide a Kantorovich-like theorem that proves that under certain conditions the sequence generated by Steffensen's method is well defined and converges to a fixed point of g .

3.3.2 Multivariate Case

In [17, pp. 115 – 118] Henrici generalizes the one dimensional Steffensen method to the multivariate case.

² $g(x) > \alpha$, $g'(x) < 0$, $g''(x) > 0$

³(a) $0 \leq g'(x) < 1$, $g''(x) > 0$; (b) $g'(x) > 1$, $g''(x) > 0$

Given a nonlinear operator $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^n$, the fixed point of which has to be found. As in the scalar case the method constructs in step j a hyperplane $\pi : \mathbf{y} = \mathbf{G}^{(j)}\mathbf{x} + \mathbf{d}^{(j)}$ interpolating $n + 1$ consecutive operator values $\mathbf{s}_j = \mathbf{x}_0^{(j)}, \mathbf{x}_1^{(j)} = \mathbf{g}(\mathbf{x}_0^{(j)}), \dots, \mathbf{x}_{n+1}^{(j)} = \mathbf{g}(\mathbf{x}_n^{(j)})$. As before we suppose that the method performs a step from j to $j + 1$, and we drop these indices in the further analysis in order to improve the readability of the presented formulas.

Let us first define the *generalized divided difference* — a term used by Johnson and Scholz [24] motivated by the definition of the *Steigung* by Schmidt in [33].

Definition 3.11 *Given an operator $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and a matrix operator $\mathbf{X}_{n-1} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ as defined by (3.10). The matrix operator $\mathbf{G}(\mathbf{x}, \mathbf{y}) : \mathbb{R}^{n \times 2} \rightarrow \mathbb{R}^{n \times n}$ implicitly defined by*

$$\mathbf{G}(\mathbf{x}, \mathbf{y})(\mathbf{X}_{n-1}(\mathbf{y}) - \mathbf{X}_{n-1}(\mathbf{x})) = \mathbf{X}_{n-1}(\mathbf{g}(\mathbf{y})) - \mathbf{X}_{n-1}(\mathbf{g}(\mathbf{x}))$$

is called generalized divided difference.

Note that $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{G}(\mathbf{y}, \mathbf{x})$. Also note that Definition 3.11 is a special case of the divided difference $\Gamma(\mathbf{x}, \mathbf{y}) : \mathbb{R}^{n \times 2} \rightarrow \mathbb{R}^{n \times 2}$,

$$\Gamma(\mathbf{x}, \mathbf{y})(\mathbf{y} - \mathbf{x}) = \mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{x}),$$

as defined by Johnson and Scholz.

This special case implies

$$\lim_{\mathbf{h} \rightarrow 0} \frac{\|\mathbf{g}(\mathbf{x} + \mathbf{h}) - \mathbf{g}(\mathbf{x}) - \mathbf{G}(\mathbf{x}, \mathbf{x} + \mathbf{h})\mathbf{h}\|}{\|\mathbf{h}\|} = 0.$$

Therefore, if the Fréchet derivative of \mathbf{g} exists at \mathbf{x} , it can be obtained by $\lim_{\mathbf{h} \rightarrow 0} \mathbf{G}(\mathbf{x}, \mathbf{x} + \mathbf{h})$. Ortega and Rheinboldt call \mathbf{G} a *consistent approximation* of \mathbf{Dg} [28, p. 355]. Moreover, according to Ortega and Rheinboldt, \mathbf{G} is even

a *strongly consistent approximation* of \mathbf{Dg} , because the following inequality holds in a vicinity of \mathbf{x} ,

$$\|\mathbf{Dg}(\mathbf{x}) - \mathbf{G}(\mathbf{x}, \mathbf{x} + \mathbf{h})\| \leq c\|\mathbf{h}\|.$$

As shown in Equation (3.8) the linear operator $\mathbf{G}^{(j)}$ of the hyperplane π satisfies $\delta\mathbf{x}_i = \mathbf{G}^{(j)}\delta\mathbf{x}_{i-1}$, and therefore $\mathbf{G}^{(j)}$ satisfies

$$\begin{aligned} \Delta_{n-1}(\mathbf{g}(\mathbf{s}_j)) &= \mathbf{X}_{n-1}(\mathbf{g}^2(\mathbf{x}_0)) - \mathbf{X}_{n-1}(\mathbf{g}(\mathbf{x}_0)) = \\ &[\delta\mathbf{x}_0, \delta\mathbf{x}_1, \dots, \delta\mathbf{x}_{n-1}] = \mathbf{G}^{(j)} \cdot [\delta\mathbf{x}_0, \delta\mathbf{x}_1, \dots, \delta\mathbf{x}_{n-1}] \\ &= \mathbf{X}_{n-1}(\mathbf{g}(\mathbf{x}_0)) - \mathbf{X}_{n-1}(\mathbf{x}_0) = \mathbf{G}^{(j)} \cdot \Delta_{n-1}(\mathbf{s}_j). \end{aligned} \quad (3.31)$$

This equation shows that $\mathbf{G}^{(j)}$ is actually a generalized divided difference. In particular

$$\mathbf{G}^{(j)} := \mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j))$$

The offset \mathbf{d} of the hyperplane to be determined can be obtained analogously to the scalar case,

$$\mathbf{d} := \mathbf{x}_1 - \mathbf{G}^{(j)}\mathbf{x}_0 = \mathbf{g}(\mathbf{s}_j) - \mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j))\mathbf{s}_j.$$

Henrici obtains a slightly different formula by using the same dependency for \mathbf{x}_{n+1} and \mathbf{x}_n , which makes no difference mathematically because of the constructed linear relation of the operator values. However, our approach will show that Steffensen's method and vector extrapolation methods are actually equivalent.

From step j to $j+1$ we determine the intersection \mathbf{s}_{j+1} of the hyperplane determined by \mathbf{s}_j with the space diagonal,

$$\mathbf{s}_{j+1} = \mathbf{G}\mathbf{s}_{j+1} + \mathbf{d} = \mathbf{G}\mathbf{s}_{j+1} + \mathbf{g}(\mathbf{s}_j) - \mathbf{G}\mathbf{s}_j.$$

Using $\mathbf{g}(\mathbf{s}_j) = (\mathbf{g}(\mathbf{s}_j) - \mathbf{s}_j) + \mathbf{s}_j$ and solving for \mathbf{s}_{j+1} the equation reveals the next iterate as

$$\mathbf{s}_{j+1} = \mathbf{s}_j + [\mathbf{1} - \mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j))]^{-1}(\mathbf{g}(\mathbf{s}_j) - \mathbf{s}_j). \quad (3.32)$$

It is possible to determine the inverse of $1 - \mathbf{G}$ without constructing \mathbf{G} . By Equation (3.31) we know that $\mathbf{G} = \Delta_{n-1}(\mathbf{g}(\mathbf{s}_j))\Delta_{n-1}(\mathbf{s}_j)^{-1}$. Thus we can rewrite $1 - \mathbf{G}$ as

$$1 - \mathbf{G} = [\Delta_{n-1}(\mathbf{s}_j) - \Delta_{n-1}(\mathbf{g}(\mathbf{s}_j))]\Delta_{n-1}(\mathbf{s}_j)^{-1}.$$

Note that by (3.18) $\Delta_{n-1}(\mathbf{s}_j) - \Delta_{n-1}(\mathbf{g}(\mathbf{s}_j))$ is the negative matrix of second order differences $-\Delta_{n-1}^2(\mathbf{s}_j)$. Therefore the inverse of $1 - \mathbf{G}$ is $-\Delta_{n-1}^2(\mathbf{s}_j)\Delta_{n-1}(\mathbf{s}_j)^{-1}$. Hence Stef-fensen's method for multivariate systems becomes

$$\mathbf{s}_{j+1} = \mathbf{s}_j - \Delta_{n-1}(\mathbf{s}_j)\Delta_{n-1}^2(\mathbf{s}_j)^{-1}\delta\mathbf{s}_j. \tag{3.33}$$

Note that this expression is indeed a generalization of the scalar case (3.22).

Definition 3.12 *The iteration defined by (3.32) or (3.33) respectively is called Henrici's method.*

An implementation for Henrici's method according to this definition is given in Algorithm 3.2.

If we examine Equation (3.17) we see that in each step vector extrapolation methods determine

$$\mathbf{s}_{j+1} = \mathbf{x}_0 + \tilde{\mathbf{A}}\delta\mathbf{x}_0 = \mathbf{s}_j + \tilde{\mathbf{A}}\delta\mathbf{s}_j,$$

EQUIVALENCE
OF VECTOR
EXTRAPOLA-
TION

with $\tilde{\mathbf{A}}$ denoting an approximation of \mathbf{A}^{-1} restricted to the span of $\delta\mathbf{x}_0$ or $\delta\mathbf{s}_j$ respectively. On the other hand Henrici's method determines the approximate solution by (3.32), which is an equivalent formulation if $\Delta_{n-1}(\mathbf{s}_j)$ is invertible. We have proven the following lemma.

Lemma 3.13 *Given a vector extrapolation method accelerating the convergence of the given nonlinear sequence (3.20). If $k(j)$ is always chosen such that the degree of the minimal polynomial is obtained at the end of each step, i.e. $\tilde{\mathbf{A}}_{(j)} = \mathbf{A}_{(j)}^{-1}$, the vector extrapolation method generates the same iterates as Henrici's method for multivariate systems. \square*

Algorithm 3.2 Henrici's method according to (3.32) and (3.33).

function Henrici(**in:** $\mathbf{g}()$, \mathbf{s}_0 , ε): **vector:**

$\mathbf{r}_0 := \mathbf{g}(\mathbf{s}_0) - \mathbf{s}_0$;

$j := 0$;

while $\|\mathbf{r}_0\| > \varepsilon \|\mathbf{r}_j\|$ **do**

{Compute sequence elements and differences}

$\mathbf{x}_0^{(j)} := \mathbf{s}_j$; $\mathbf{x}_1^{(j)} := \mathbf{g}(\mathbf{x}_0^{(j)})$;

$\delta\mathbf{x}_0^{(j)} := \mathbf{x}_1^{(j)} - \mathbf{x}_0^{(j)}$;

for $i := 2$ **to** $n + 1$ **do**

$\mathbf{x}_i^{(j)} := \mathbf{g}(\mathbf{x}_{i-1}^{(j)})$;

$\delta\mathbf{x}_{i-1}^{(j)} := \mathbf{x}_i^{(j)} - \mathbf{x}_{i-1}^{(j)}$;

$\delta^2\mathbf{x}_{i-2}^{(j)} := \delta\mathbf{x}_{i-1}^{(j)} - \delta\mathbf{x}_{i-2}^{(j)}$;

end;

{Variant (3.32)}

$\mathbf{C} := \Delta_n^2(\mathbf{s}_j)\Delta_n(\mathbf{s}_j)^{-1}$;

$\mathbf{h} := \mathbf{C}^{-1}\delta\mathbf{x}_0^{(j)}$;

{Variant (3.33)}

$\mathbf{t} := \Delta_n^2(\mathbf{s}_j)^{-1}\delta\mathbf{x}_0^{(j)}$;

$\mathbf{h} := -\Delta_n(\mathbf{s}_j)\mathbf{t}$;

{Compute new solution and residual}

$\mathbf{s}_{j+1} := \mathbf{s}_j + \mathbf{h}$;

$\mathbf{r}_{j+1} := \mathbf{g}(\mathbf{s}_{j+1}) - \mathbf{s}_{j+1}$;

$j := j + 1$;

end;

Henrici := \mathbf{s}_j

end Henrici;

Note the difference between Henrici's method and vector extrapolation methods if $\Delta_{n-1}(\mathbf{s}_j)$ is not of full rank. In this case \mathbf{G} cannot be constructed. Therefore (3.32) cannot be solved and (3.33) has to be solved approximately. Vector extrapolation methods, however, do not need to solve the full rank system of (3.32) in order to get a good approximation for the next step.

On the other hand Krylov space methods also solve linear systems by approximation. Since vector extrapola-

tion methods are equivalent to Krylov space methods (Corollary 3.7) the following Lemma holds.

Theorem 3.14 *Given a vector extrapolation method and its corresponding Krylov space method.*

Henrici's method applying the given Krylov space method to solve (3.32) approximately with starting vector 0 generates the same iterates as the given vector extrapolation method if both methods use the same number of intermediate steps. \square

Example 3.15 Let us illustrate the last lemma by the example of the Chandrasekhar H-equation introduced in Chapter 1. The following methods are used to find a solution:

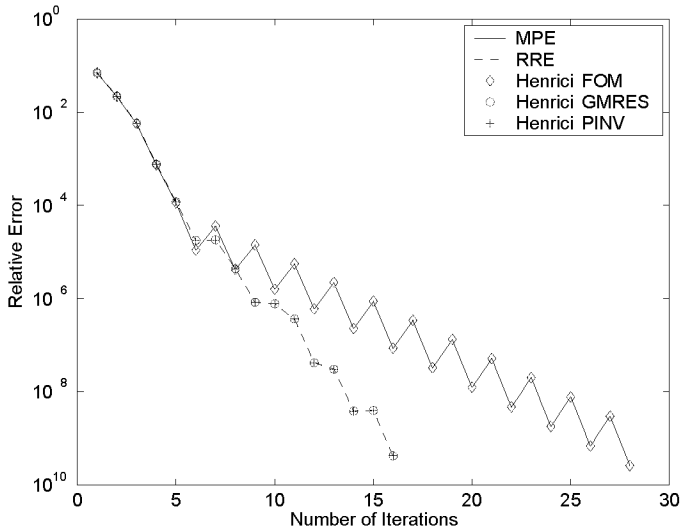
1. MPE - this is the restarted vector extrapolation method equivalent to FOM in the linear case (see 3.4.1),
2. RRE - this is the restarted vector extrapolation method equivalent to GMRES in the linear case (see 3.4.2),
3. Henrici's method using FOM to solve (3.32) (with starting vector 0 in each step),
4. like in the last item Henrici's method, but using GMRES rather than FOM,
5. Henrici's method using the pseudo inverse of $\Delta_{d-1}^2(\mathbf{s}_j)$ to solve (3.33), i.e. $\mathbf{s}_{j+1} = \mathbf{s}_j - \Delta_{n-1}(\mathbf{s}_j)\Delta_{d-1}^2(\mathbf{s}_j)^+ \delta \mathbf{s}_j$.

Due to the very bad condition of $(1 - G) \sim 10^{17}$ to 10^{22} — we use Maple 8 on a Linux workstation with an accuracy of 38 digits so that we are able to compare the algorithms. The problem size is $n = 10$, and the constant c is set to 0.9999. For all algorithms the number of intermediate steps is 2. This number seems low. However, for bigger numbers the algorithms become indistinguishable in the plots.

The plot of the evolution of the relative errors during the algorithms is shown in Figure 3.4. By varying the accuracy we observed that the oscillating behavior of MPE and Henrici-FOM is not a numeric effect. We see:

- Algorithms based on (3.32) and on a least squares solution of (3.33) are not equivalent in general.

Figure 3.4 Error plots of four different algorithms for finding the solution of the Chandrasekhar H-equation: two vector extrapolation methods and three variants of Henrici’s method are implemented and run in Maple 8 with $\text{Digits} = 38$, $n = 10$, $c = 0.9999$.



- A vector extrapolation method can be formulated as a Henrici method based on (3.32) using the corresponding linear solver. Henrici using FOM is equivalent to MPE, Henrici using GMRES is equivalent to RRE.
- The least squares solution of (3.33) is equivalent to a Henrici method using GMRES and to the corresponding vector extrapolation method RRE.

WELL DEFINED METHOD, CONVERGENCE

In [24] Johnson and Scholz explicitly state that their Theorem 1 — a Kantorovich-like theorem for another multivariate Steffensen method — is not applicable to Henrici’s method. However, by the help of the generalized divided difference we have been able to reformulate their theorem such that it is applicable to this method. Our theorem states conditions such that the method

is well defined and converges R-superlinearly to a fixed point of \mathbf{g} .

Theorem 3.16 *Let $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a given operator with a generalized divided difference defined on $\mathcal{M} \subseteq \mathbb{R}^n$. Let $\mathbf{s}_0 \in \mathbb{R}^n$ be an initial guess. Suppose for $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}' \in \overline{\mathcal{B}}(\mathbf{s}_0, \max(\alpha, 2\alpha\beta))$*

$$\|\mathbf{g}(\mathbf{s}_0) - \mathbf{s}_0\| \leq \alpha, \quad (3.34)$$

$$\|[\mathbf{1} - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))]^{-1}\| \leq \beta, \quad (3.35)$$

$$\|\mathbf{G}(\mathbf{x}, \mathbf{y}) - \mathbf{G}(\mathbf{x}', \mathbf{y}')\| \leq \sigma(\|\mathbf{x} - \mathbf{x}'\| + \|\mathbf{y} - \mathbf{y}'\|). \quad (3.36)$$

Let $h := 2\sigma\alpha\beta(1 + \beta)$.

Then,

1. if $\overline{\mathcal{B}}(\mathbf{s}_0, \max(\alpha, 2\alpha\beta)) \subseteq \mathcal{M}$ and $h \leq 1/2$ then the sequence, generated by Henrici's method,

$$\mathbf{s}_{j+1} := \mathbf{b}(\mathbf{s}_j),$$

$$\mathbf{b}(\mathbf{s}_j) := \mathbf{s}_j + [\mathbf{1} - \mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j))]^{-1}(\mathbf{g}(\mathbf{s}_j) - \mathbf{s}_j), \quad (3.37)$$

is well defined and converges to a fixed point $\mathbf{s} \in \overline{\mathcal{B}}(\mathbf{s}_0, \max(\alpha, 2\alpha\beta))$ of \mathbf{g} .

2. The fixed point lies in the sphere

$$\overline{\mathcal{B}}(\mathbf{s}_0, \rho_1) \quad \text{with} \quad \rho_1 := \frac{\alpha\beta(1 - \sqrt{1 - 2h})}{h}, \quad (3.38)$$

and the R-convergence rate is given by

$$\|\boldsymbol{\eta}_j\| = \|\mathbf{s} - \mathbf{s}_j\| \leq \frac{\alpha\beta(2h)^{2^j}}{2^j h}. \quad (3.39)$$

3. Let $\rho_2 := \alpha\beta(1 + \sqrt{1 - 2h})/h$. If (3.36) holds in the larger sphere $\mathcal{B}(\mathbf{s}_0, \rho_2 + \alpha)$ then the fixed point is unique in the ball $\mathcal{B}(\mathbf{s}_0, \rho_2)$.

Proof.

1. Let us define $\alpha_0 := \alpha$, $\beta_0 := \beta$, $h_0 := h$ and

$$\alpha_1 := \frac{\alpha_0 h_0}{2}, \quad \beta_1 := \frac{\beta_0}{1 - h_0}, \quad h_1 := 2\sigma\alpha_1\beta_1(1 + \beta_1).$$

For simplicity let us also define

$$\mathcal{B}_j := \overline{\mathcal{B}}(\mathbf{s}_j, \max(\alpha_j, 2\alpha_j\beta_j)).$$

First we show that \mathcal{B}_1 is a subset of \mathcal{B}_0 : Since $h \leq 1/2$ we have

$$\alpha_1 \leq \frac{\alpha_0}{4} \leq \alpha_0 \quad \text{and} \quad (3.40)$$

$$\alpha_1\beta_1 = \frac{\alpha_0\beta_0 h_0}{2(1 - h_0)} \leq \frac{\alpha_0\beta_0}{2}. \quad (3.41)$$

By the two bounds (3.34) and (3.35) and the definition of \mathbf{b} (3.37) the following inequality holds

$$\begin{aligned} \|\mathbf{s}_1 - \mathbf{s}_0\| &\leq \|[\mathbf{I} - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))]^{-1}\| \|\mathbf{g}(\mathbf{s}_0) - \mathbf{s}_0\| \\ &\leq \alpha_0\beta_0. \end{aligned} \quad (3.42)$$

Thus we have $\mathcal{B}_1 \subseteq \mathcal{B}_0$.

Secondly we show that (3.34) holds with respect to \mathbf{s}_1 and α_1 . We use (3.42), the definition of the generalized divided difference and an implication of (3.37),

$$\mathbf{s}_1 - \mathbf{g}(\mathbf{s}_0) = \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))(\mathbf{s}_1 - \mathbf{s}_0),$$

to show

$$\begin{aligned} \|\mathbf{g}(\mathbf{s}_1) - \mathbf{s}_1\| &= \|\mathbf{g}(\mathbf{s}_1) - \mathbf{g}(\mathbf{s}_0) + \mathbf{g}(\mathbf{s}_0) - \mathbf{s}_1\| \\ &\leq \|\mathbf{G}(\mathbf{s}_1, \mathbf{s}_0) - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))\| \|\mathbf{s}_1 - \mathbf{s}_0\| \\ &\leq \sigma(\|\mathbf{s}_1 - \mathbf{s}_0\| + \|\mathbf{s}_0 - \mathbf{g}(\mathbf{s}_0)\|) \|\mathbf{s}_1 - \mathbf{s}_0\| \\ &\leq \sigma\alpha_0^2\beta_0(1 + \beta_0) = \alpha_1. \end{aligned} \quad (3.43)$$

Thirdly we show that (3.35) holds with respect to \mathbf{s}_1 and β_1 . We use (3.34), (3.36), (3.42) and (3.43) to show

$$\begin{aligned}
 & \| \mathbf{G}(\mathbf{s}_1, \mathbf{g}(\mathbf{s}_1)) - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0)) \| \\
 & \leq \sigma (\| \mathbf{s}_1 - \mathbf{s}_0 \| + \| \mathbf{g}(\mathbf{s}_1) - \mathbf{g}(\mathbf{s}_0) \|) \\
 & = \sigma (\| \mathbf{g}(\mathbf{s}_1) - \mathbf{s}_1 + \mathbf{s}_1 - \mathbf{s}_0 + \mathbf{s}_0 - \mathbf{g}(\mathbf{s}_0) \| + \| \mathbf{s}_1 - \mathbf{s}_0 \|) \\
 & \leq \sigma (\| \mathbf{g}(\mathbf{s}_1) - \mathbf{s}_1 \| + 2 \| \mathbf{s}_1 - \mathbf{s}_0 \| + \| \mathbf{s}_0 - \mathbf{g}(\mathbf{s}_0) \|) \\
 & \leq \sigma \left(\frac{\alpha_0}{4} + 2\alpha_0\beta_0 + \alpha_0 \right) \\
 & \leq \sigma\alpha_0 \left(\frac{5}{4} + 2\beta_0 \right) \\
 & < 2\sigma\alpha_0(1 + \beta_0) = \frac{h_0}{\beta_0}.
 \end{aligned}$$

By the help of this estimate, (3.35) and $h_0 \leq 1/2$ we are now able to apply the perturbation lemma (Lemma 1.16) and conclude

$$\| (1 - \mathbf{G}(\mathbf{g}(\mathbf{s}_1), \mathbf{s}_1))^{-1} \| \leq \frac{\beta_0}{1 - h_0} = \beta_1.$$

Thus all hypotheses of the theorem are satisfied with respect to \mathbf{s}_1 and the constants indexed by 1. By induction, it follows for all $j > 1$ that \mathbf{s}_j exists, that \mathbf{s}_j satisfies the hypotheses of the theorem with respect to the constants α_j , β_j and h_j and that $\mathcal{B}_j \subseteq \mathcal{B}_{j-1}$. Since the radius of \mathcal{B}_j decreases in each step, $\mathbf{s} := \mathbf{s}_\infty$ is the only element in $\lim_{j \rightarrow \infty} \mathcal{B}_j$ and thus is a fixed point of \mathbf{b} . As the induction on (3.40) shows, $\alpha_j \leq \alpha_0/4^j$, and

$$\| \mathbf{g}(\mathbf{s}_j) - \mathbf{s}_j \| \leq \frac{\alpha_0}{4^j}.$$

As j tends to infinity, the right hand side of this equation tends to zero and \mathbf{s}_j tends to \mathbf{s} . Therefore \mathbf{s} is also a fixed point of \mathbf{g} .

2. By the definitions of the constants h_j , α_j and β_j we get from the induction in proof of statement 1

$$h_{j+1} = \frac{2\sigma\alpha_j\beta_j(1+\beta_j)h_j(1-h_j/(1+\beta_j))}{2(1-h_j)^2} \leq \frac{h_j^2}{2(1-h_j)^2}. \quad (3.44)$$

From (3.41) we know

$$v_{j+1} := \alpha_{j+1}\beta_{j+1} = v_j \frac{h_j}{2(1-h_j)}. \quad (3.45)$$

Let us define $\rho_1(h) := (1 - \sqrt{1-2h})/h$. For $0 < h \leq \frac{1}{2}$ this functional is greater than 1 and increases monotonously. Hence we can write

$$\begin{aligned} v_j &\leq v_j \rho_1(h_j) \leq v_j \rho_1\left(\frac{h_{j-1}^2}{2(1-h_{j-1})^2}\right) \\ &= v_{j-1} \rho_1(h_{j-1}) - v_{j-1} \leq v_0 \rho_1(h_0) - \sum_{i=0}^{j-1} v_i. \end{aligned}$$

By (3.42) and the latter relation we conclude

$$\|s_{j+1} - s_0\| \leq \sum_{i=0}^j v_i \leq v_0 \rho_1(h_0) \quad \text{for all } j > 0.$$

With $j \rightarrow \infty$ we get (3.38).

For the convergence rate we need two inequalities that can be proven by simple induction using the inequalities (3.44) and (3.45)

$$\begin{aligned} h_j &\leq \frac{(2h_0)^{2^j}}{2}, \\ v_j &\leq \frac{v_0(2h_0)^{2^j-1}}{2^j}. \end{aligned}$$

From the latter we get the estimate

$$\begin{aligned} \|\mathbf{s}_{j+k+1} - \mathbf{s}_j\| &\leq \sum_{i=0}^k v_{j+i} \leq \frac{v_0(2h_0)^{2^j}}{2^{j+1}h_0} \sum_{i=0}^k \frac{(2h_0)^{2^i}}{2^i} \\ &\leq \frac{v_0(2h_0)^{2^j}}{2^j h_0}, \end{aligned}$$

which, for $k \rightarrow \infty$, yields (3.39).

3. The uniqueness of the fixed point is proved by contradiction. Let us suppose that there exists another fixed point $\bar{\mathbf{s}} \neq \mathbf{s}$ in $\mathcal{B}(\mathbf{s}_0, \rho_2)$ satisfying $\mathbf{g}(\bar{\mathbf{s}}) = \bar{\mathbf{s}}$.

Let us define $\rho_2(h) := (1 + \sqrt{1 - 2h})/h$. For $0 < h \leq 1/2$ this functional is greater than 2 and decreases monotonously. We use the following ansatz

$$\|\bar{\mathbf{s}} - \mathbf{s}_0\| = \theta_0 \rho_2(h_0) v_0 \quad \text{with} \quad 0 \leq \theta_0 < 1.$$

Let us define two auxiliary operators

$$\begin{aligned} \boldsymbol{\varphi}(\mathbf{x}) &:= \mathbf{x} + [\mathbf{1} - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))]^{-1}(\mathbf{g}(\mathbf{x}) - \mathbf{x}), \\ \boldsymbol{\Phi}(\mathbf{x}) &:= \mathbf{1} + [\mathbf{1} - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))]^{-1}(\mathbf{G}(\mathbf{x}, \mathbf{g}(\mathbf{x})) - \mathbf{1}). \end{aligned}$$

Note that $\mathbf{s}_1 = \boldsymbol{\varphi}(\mathbf{s}_0)$, $\bar{\mathbf{s}} = \boldsymbol{\varphi}(\bar{\mathbf{s}})$ and $\boldsymbol{\Phi}(\mathbf{s}_0) = \mathbf{0}$. We construct

$$\begin{aligned} \|\bar{\mathbf{s}} - \mathbf{s}_1\| &= \|\boldsymbol{\varphi}(\bar{\mathbf{s}}) - \boldsymbol{\varphi}(\mathbf{s}_0) - \boldsymbol{\Phi}(\mathbf{s}_0)(\bar{\mathbf{s}} - \mathbf{s}_0)\| \\ &\leq \beta_0 \|\mathbf{g}(\bar{\mathbf{s}}) - \mathbf{g}(\mathbf{s}_0) - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))(\bar{\mathbf{s}} - \mathbf{s}_0)\| \\ &= \beta_0 \|[\mathbf{G}(\mathbf{s}_0, \bar{\mathbf{s}}) - \mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0))](\bar{\mathbf{s}} - \mathbf{s}_0)\| \\ &\leq \beta_0 \sigma \|\bar{\mathbf{s}} - \mathbf{g}(\mathbf{s}_0)\| \|\bar{\mathbf{s}} - \mathbf{s}_0\| \\ &\leq \beta_0 \sigma (\|\bar{\mathbf{s}} - \mathbf{s}_0\| + \alpha_0) \|\bar{\mathbf{s}} - \mathbf{s}_0\|. \end{aligned}$$

Let us set $\theta_1 := \max(\theta_0, 1/2) < 1$. Since $\rho_2(h_0) > 2$ we have $\alpha_0 \leq \alpha_0 \theta_1 \rho_2(h_0)$. We also have $\|\bar{s} - s_0\| \leq \theta_1 \alpha_0 \beta_0 \rho_2(h_0)$. Therefore

$$\begin{aligned} \|\bar{s} - s_1\| &\leq \theta_1 \sigma \alpha_0 \beta_0 (1 + \beta_0) \rho_2(h_0) h_0 \|\bar{s} - s_0\| \\ &= \theta_1 \theta_0 \frac{h_0}{2} \rho_2(h_0)^2 v_0 \\ &= \theta_1 \theta_0 (\rho_2(h_0) - 1) v_0 \\ &\leq \theta_1 \theta_0 \rho_2(h_1) v_1. \end{aligned}$$

Let us define $\theta_i := \max(\prod_{k=0}^i \theta_k, 1/2)$ for $i > 1$. By induction we get

$$\begin{aligned} \|\bar{s} - s_j\| &\leq \rho_2(h_j) v_j \prod_{i=0}^j \theta_i \leq \frac{1}{(1 + \beta_j) \sigma} \prod_{i=0}^j \theta_i \\ &\leq \frac{1}{(1 + \beta_0) \sigma} \prod_{i=0}^j \theta_i. \end{aligned}$$

Hence $\lim_{j \rightarrow \infty} \|\bar{s} - s_j\| = 0$, and the sequence $\{s_j\}$ converges to \bar{s} . But this is absurd because it has been proved in Step 2 that this sequence converges to s . Therefore \bar{s} must equal s . This proves the uniqueness of the fixed point in $\mathcal{B}(s_0, \rho_2)$.

□

The convergence rate given by (3.39) can be analyzed as follows. Let $\xi_j := \alpha \beta (2h)^{2^j} / (2^j h)$. We can express ξ_{j+1} by

$$\xi_{j+1} = \frac{\alpha \beta (2h)^{2^{j+1}}}{2^j h} = \frac{(2h)^{2^j} \alpha \beta (2h)^{2^j}}{2 \cdot 2^j h} = \frac{(2h)^{2^j}}{2} \xi_j.$$

For $h = 1/2$ this expression becomes

$$\xi_{j+1} = \frac{1}{2} \xi_j,$$

indicating linear convergence. However, if h is less than $1/2$, the factor $\frac{(2h)^2}{2}$ converges very fast to 0, showing that the R-convergence is superlinear but not quadratic in this case.

Moreover, Ortega and Rheinboldt proved quadratic local R- and \mathcal{Q} -convergence of (3.37) if \mathbf{Dg} is Lipschitz-continuous in a vicinity of \mathbf{s} and $\mathbf{Dg}(\mathbf{s})$ is not singular [28, pp. 369 – 374].

The proof of Ortega and Rheinboldt is based on the exact solution of the linear system of (3.37) in every step. However, to prove local quadratic convergence this condition is too restrictive. Let $d \leq n$ denote the degree of the minimal polynomial of $\mathbf{Dg}(\mathbf{s})$. A sufficient condition for quadratic convergence is that the linear system in the “very last” step is solved exactly. Therefore if a Krylov space method is used to solve the linear system of (3.37), it only has to find a solution within a Krylov space of dimension d . This is basically what Jbilou and Sadok [23] proved for the vector extrapolation methods MPE, RRE and MMPE. Based on our consideration we state that this principle holds for *all* vector extrapolation methods used to solve the linear system of (3.37). However, if the linear system is solved less exactly, the quadratic convergence rate gets lost immediately and only superlinear convergence is retained.

Example 3.17 Let us illustrate Theorem 3.16 by the original example of Henrici (17, p.117). A fixed point of the following operator has to be found

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} x_1^2 + x_2^2 \\ x_1^2 - x_2^2 \end{bmatrix},$$

the solution of which is $\mathbf{s} = [0.7718, 0.4196]^\top$. The derivative of \mathbf{g} is $\mathbf{Dg}(\mathbf{x}) = \mathbf{BD}(\mathbf{x})$ with

$$\mathbf{D}(\mathbf{x}) := \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix} \quad \text{and} \quad \mathbf{B} := \begin{bmatrix} 2 & 2 \\ 2 & -2 \end{bmatrix}.$$

We are now able to estimate

$$\|\mathbf{D}\mathbf{g}(\mathbf{x}) - \mathbf{D}\mathbf{g}(\mathbf{y})\| = \|\mathbf{B}\mathbf{D}(\mathbf{x} - \mathbf{y})\| \leq \|\mathbf{B}\|\|\mathbf{x} - \mathbf{y}\|.$$

This way we get an upper bound for $\sigma = \|\mathbf{B}\| = 2.8284$ in this example. Let us choose $\mathbf{s}_0 = [0.79, 0.41]^T$ as the starting vector for Henrici's method. Then we have $\alpha = \|\mathbf{g}(\mathbf{s}_0) - \mathbf{s}_0\| = 0.04605$ and $\beta = \|[\mathbf{G}(\mathbf{s}_0, \mathbf{g}(\mathbf{s}_0)) - \mathbf{1}]^{-1}\| = 0.9725$ and finally $h = 2\sigma\alpha\beta(1 + \beta) = 0.4997 < 0.5$. Hence all the conditions for Theorem 3.16 are satisfied. We can be sure that Henrici's method applied to the problem converges.

Table 3.2 Development of the relative error, $\alpha_j = \|\mathbf{g}(\mathbf{s}_j) - \mathbf{s}_j\|$, $\beta_j = \|[\mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j)) - \mathbf{1}]^{-1}\|$ and $h_j = 2\sigma\alpha_j\beta_j(1 + \beta_j)$ of Henrici's method applied to a fixed point problem $\mathbf{g}(\mathbf{x}) = \mathbf{x}$ originally published by Henrici. All the values stay below their bounds given in Theorem 3.16.

j	$\ \mathbf{n}_j\ /\ \mathbf{s}\ $	α_j	β_j	h_j
0	$2.3400 \cdot 10^{-02}$	$4.6053 \cdot 10^{-02}$	0.9725	$4.9971 \cdot 10^{-1}$
1	$1.2149 \cdot 10^{-03}$	$1.6989 \cdot 10^{-03}$	1.0497	$2.0678 \cdot 10^{-2}$
2	$6.4185 \cdot 10^{-06}$	$6.4185 \cdot 10^{-06}$	1.0552	$8.0097 \cdot 10^{-5}$
3	$1.9606 \cdot 10^{-10}$	$1.7933 \cdot 10^{-10}$	1.0552	$2.2000 \cdot 10^{-9}$
4	$2.1184 \cdot 10^{-15}$	0	-	0

The results are shown in Table 3.2. Although the conditions for Q-quadratic convergence are satisfied, we only see super-linear convergence in the relative error. At the fixed point \mathbf{s} $\mathbf{G}(\mathbf{s}, \mathbf{g}(\mathbf{s}))$ is not defined. Therefore β_4 could not be determined by our algorithm. However, since $\mathbf{G}(\mathbf{x}, \mathbf{y})$ is a consistent approximation of $\mathbf{D}\mathbf{g}$ we can determine $\lim_{j \rightarrow \infty} \beta_j = \|[\mathbf{D}\mathbf{g}(\mathbf{s}) - \mathbf{1}]^{-1}\| = 1.0552$. The sequence $\{\beta_j\}$ in Table 3.2 acknowledges this consideration.

As we have mentioned already in the last two examples the evaluation of $\mathbf{G}(\mathbf{s}_j, \mathbf{g}(\mathbf{s}_j))$ causes numerical problems. Vector extrapolation methods do not need to evaluate this matrix. They implicitly determine it. In the next section let us therefore describe two particular vector extrapolation methods in closer detail.

3.4 Methods Based on Polynomial Extrapolation

This section deals with the details of implementations of particular vector extrapolation methods based on polynomial extrapolation. The considerations made in the following subsections are based on the considerations made for the linear case. However, as we have already seen, the resulting algorithms can easily be adopted for the nonlinear case.

3.4.1 Minimal Polynomial Extrapolation

The Minimal Polynomial Extrapolation Method (MPE) is based on the fundamental idea that the leading coefficient of the minimal polynomial does not vanish. If it did, a polynomial of lesser degree would be the minimal polynomial. The not vanishing leading coefficient of the extrapolating polynomial is used as an invariant during the process of finding the degree of the minimal polynomial.

Let $\mu_j := \sum_{i=0}^j \beta_{i,j} \delta x_i$ be the extrapolating polynomial for finding s_j . Equation (3.11) requires that the sum of the coefficients of the minimal polynomial be 1. To find such coefficients, MPE defines $\alpha_{j,j} := 1$ and solves the least squares problem

$$\sum_{i=0}^{j-1} \alpha_{i,j} \delta x_i = \Delta_{j-1}(x_0) \mathbf{a}_j \simeq -\delta x_j \quad (3.46)$$

for $\mathbf{a}_j := [\alpha_{0,j}, \dots, \alpha_{j-1,j}]^T$. To fulfill the constraint (3.16), the solution of this problem is divided by the sum of all coefficients $\sigma_j := 1 + \sum_{i=0}^{j-1} \alpha_{i,j}$. Thus the coefficients of μ_j are obtained by

$$\beta_{i,j} = \frac{\alpha_{i,j}}{\sigma_j} \quad \text{for } i = 0, \dots, j.$$

By the help of (3.17) the approximate solution \mathbf{s}_j is then determined by

$$\mathbf{s}_j = \sum_{i=0}^j \frac{\alpha_{i,j}}{\sigma_j} \mathbf{x}_i = \mathbf{x}_0 + \sum_{i=0}^{j-1} \gamma_{i,j} \delta \mathbf{x}_i$$

with

$$\gamma_{i,j} := \sum_{k=i+1}^j \frac{\alpha_{k,j}}{\sigma_j} \quad \text{for } i = 0, \dots, j-1, \quad (3.47)$$

or simpler

$$\mathbf{s}_j = \mathbf{x}_0 + \Delta_{j-1}(\mathbf{x}_0) \mathbf{c}_j \quad \text{with } \mathbf{c}_j := [\gamma_{0,j}, \dots, \gamma_{j-1,j}]^T.$$

SAVING
MEMORY

Applying the following procedure neither the sequence vectors nor the difference vectors have to be stored. Let us denote the QR decomposition of $\Delta_j(\mathbf{x}_0)$ by $\mathbf{Q}_{j+1} \mathbf{R}_{j+1} = \Delta_j(\mathbf{x}_0)$. Then

$$\mathbf{Q}_{j+1} \mathbf{R}_{j+1} = \Delta_j(\mathbf{x}_0) = [\Delta_{j-1}(\mathbf{x}_0) | \delta \mathbf{x}_j] = [\mathbf{Q}_j \mathbf{R}_j | \mathbf{Q}_j \boldsymbol{\rho}_{j+1}],$$

with $\boldsymbol{\rho}_{j+1}$ being the last column vector of \mathbf{R}_{j+1} . The least squares problem (3.46) can now be transformed into a linear system

$$\mathbf{R}_j \mathbf{a}_j = -\mathbf{Q}_j^H \delta \mathbf{x}_j = -\mathbf{Q}_j^H \mathbf{Q}_j \boldsymbol{\rho}_{j+1} = -\bar{\boldsymbol{\rho}}_{j+1}. \quad (3.48)$$

In the last equation $\bar{\boldsymbol{\rho}}_{j+1} := [r_{1,j+1}, \dots, r_{j,j+1}]^T$ holds the first j components of the last column vector of \mathbf{R}_{j+1} . The update procedure used for solving the linear systems of FOM and GMRES (Section 2.4.5) can also be used to solve this linear system.

DETERMINING
THE RESIDUAL
NORM

Evaluating the norm of the residual by determining the residual vector according to (3.14) and then taking its norm in each step is a rather expensive way. Sidi [37] has shown that this job can be done much cheaper.

With the help of (3.48) the residual vector in (3.14) becomes

$$\begin{aligned} \mathbf{r}_j &= \frac{1}{\sigma_j} \Delta_j(\mathbf{x}_0) \begin{bmatrix} \mathbf{a}_j \\ 1 \end{bmatrix} = \frac{1}{\sigma_j} \mathbf{Q}_{j+1} \left(\begin{bmatrix} \mathbf{R}_j \mathbf{a}_j \\ 0 \end{bmatrix} + \boldsymbol{\rho}_{j+1} \right) \\ &= \frac{1}{\sigma_j} \mathbf{Q}_{j+1} \left(- \begin{bmatrix} \bar{\boldsymbol{\rho}}_{j+1} \\ 0 \end{bmatrix} + \begin{bmatrix} \bar{\boldsymbol{\rho}}_{j+1} \\ \mathbf{r}_{j+1,j+1} \end{bmatrix} \right) = \frac{1}{\sigma_j} \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_{j+1,j+1} \end{bmatrix}. \end{aligned}$$

Hence the residual norm becomes $\|\mathbf{r}_j\| = r_{j+1,j+1}/|\sigma_j|$, which by (3.47) turns into $\|\mathbf{r}_j\| = r_{j+1,j+1}|\gamma_{j-1,j}|$.

All the statements made so far result in a version of MPE that is shown in Algorithm 3.3. This algorithm uses an operator $\mathbf{g}()$ and the initial guess to generate the vector sequence on the fly. The QR decomposition in the algorithm is performed by the modified Gram Schmidt process of Algorithm 3.1.

Solving the linear least squares problem (3.46) can be described algebraically by its normal equations. These can easily be transformed into

EQUIVALENT
KRYLOV
SPACE
METHOD:
FOM

$$0 = \sigma_j \Delta_{j-1}(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) \mathbf{b}_j = \sigma_j \Delta_j(\mathbf{x}_0)^H \mathbf{r}_j.$$

Thus the projection method consists of projecting the residual vector \mathbf{r}_j unto $\text{span}\{\Delta_{j-1}(\mathbf{x}_0)\}^\perp$. As (3.17) tells us, the associated correction space is $\text{span}\{\Delta_{j-1}(\mathbf{x}_0)\}$. Therefore $\mathcal{C}_j = \mathcal{P}_j$, which proves the following theorem.

Theorem 3.18 *The minimal polynomial extrapolation method (MPE) is mathematically equivalent to FOM applied to System (3.2). \square*

3.4.2 Reduced Rank Extrapolation

The idea of the Reduced Rank Extrapolation method (RRE) is to minimize the residual of (3.14) by applying

Algorithm 3.3 Minimal Polynomial Extrapolation.

```

function MPE(in:  $g(\cdot)$ ,  $x_0$ ,  $\varepsilon$ ): vector;
   $x_1 := g(x_0)$ ;  $\delta x_0 := x_1 - x_0$ ;  $c_{-1} := []$ ;
   $R_0 := []$ ;  $Q_0 := []$ ;
   $R_1 := \|\delta x_0\|$ ;  $Q_1 := \delta x_0 / r_{1,1}$ ;
   $\rho_0 := \|\delta x_0\|$ ;
   $j := 0$ ;
  while  $\rho_j > \rho_0 \varepsilon$  do
     $x_{j+2} := g(x_{j+1})$ ;           {new sequence vector}
     $\delta x_{j+1} := x_{j+2} - x_{j+1}$ ;   {new difference vector}
     $Q_{j+2} R_{j+2} := \Delta_{j+1}(x_0)$ ; {successive QR decomp.}
     $\hat{\rho}_{j+2} := [r_{1,j+2}, \dots, r_{j,j+2}]^T$ ; {lsq rhs vector}
     $a_{j+1} := -R_{j+1}^{-1} \hat{\rho}_{j+2}$ ;   {least squares solution}
     $\sigma_{j+1} := 1 + \sum_{i=0}^j \alpha_{i,j+1}$ ; {sum of coefficients}

    {transform coefficients}
     $\gamma_{j,j+1} := \frac{1}{\sigma_{j+1}}$ ;
    for  $i := j - 1$  to 0 by -1 do
       $\gamma_{i,j+1} := \gamma_{i+1,j+1} + \alpha_{i+1,j+1} / \sigma_{j+1}$ ;
    end;
     $\rho_{j+1} := |\gamma_{j,j+1}| r_{j+2,j+2}$ ;   {residual norm}
     $j := j + 1$ ;
  end;
   $s_j := x_0 + Q_j R_j c_j$ ;
  MPE :=  $s_j$ 
end MPE;

```

the constraint (3.16). That is

$$r_j = \Delta_j(x_0) b_j \simeq 0 \quad \text{s.t.} \quad \sum_{i=0}^j \beta_{i,j} = 1. \quad (3.49)$$

To solve this constrained least squares problem the following trick is used: Define a matrix $B_{j+1} \in \mathbb{R}^{(j+1) \times (j+1)}$

$$\mathbf{B}_{j+1} := \begin{bmatrix} 1 & \dots & \dots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{bmatrix} \quad \text{with } \mathbf{B}_{j+1}^{-1} = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix}.$$

With the definition of this matrix the residual can be transformed into

$$\mathbf{r}_j = \Delta_j(\mathbf{x}_0)\mathbf{b}_j = \Delta_j(\mathbf{x}_0)\mathbf{B}_{j+1}^{-1}\mathbf{B}_{j+1}\mathbf{b}_j = \delta\mathbf{x}_0 + \Delta_j^2(\mathbf{x}_0)\mathbf{c}_j.$$

Note that the components of $\mathbf{B}_{j+1}\mathbf{b}_j = [1, \gamma_{0,j}, \dots, \gamma_{j-1,j}]^T$ indeed meet the definition of $\gamma_{k,j}$ in Lemma 3.4. The constraint is used to obtain the first component of this vector. Now $\Delta_j(\mathbf{x}_0)\mathbf{B}_{j+1}^{-1} = [\delta\mathbf{x}_0, \Delta_j^2(\mathbf{x}_0)]$ which results in the given formula for \mathbf{r}_j . This result allows us to solve the constrained least squares problem of (3.49) by the unconstrained least squares problem

$$\Delta_j^2(\mathbf{x}_0)\mathbf{c}_j \simeq -\delta\mathbf{x}_0. \tag{3.50}$$

This derivation is used for theoretical purposes only. Sidi [37] uses another way to solve (3.49). Let us rewrite the system as

$$\begin{aligned} \mathbf{r}_j = \Delta_j(\mathbf{x}_0)\mathbf{b}_j \simeq \mathbf{0} \quad \text{s.t.} \quad \mathbf{W}_{j+1}\mathbf{b}_{i,j} = 1 \\ \text{with } \mathbf{W}_{j+1} := [1, \dots, 1]. \end{aligned}$$

This system can be solved by the classical method of Lagrange for solving a minimization problem with constraints. Let us define the principal functional to be minimized

$$\phi(\mathbf{b}_j, \lambda) := \mathbf{b}_j^H \Delta_j(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) \mathbf{b}_j + 2\lambda(1 - \mathbf{b}_j^H \mathbf{W}_{j+1}^H).$$

In this principal functional we introduced a factor 2 for connecting the linear constraint. This is a legal action, because we may argue that we “borrowed” this factor from λ . We mainly introduced it to avoid a nasty number in the final system to be solved. The gradient of ϕ is given by

$$\nabla\phi(\mathbf{b}_j, \lambda) = \left[2\mathbf{b}_j^H \Delta_j(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) + 2\lambda \mathbf{W}_{j+1, 2} - 2\mathbf{b}_j^H \mathbf{W}_{j+1}^H \right].$$

The gradient has to vanish for a particular pair of \mathbf{b}_j and λ . The system to be solved is therefore the following linear system,

$$\begin{bmatrix} \Delta_j(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) & -\mathbf{W}_{j+1}^H \\ \mathbf{W}_{j+1} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_j \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (3.51)$$

System (3.51) can now be solved within the following steps:

1. Solve $\Delta_j(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) \mathbf{z} = \mathbf{W}_{j+1}^H$ for \mathbf{z} .
2. Determine $\lambda = 1/(\mathbf{W}_{j+1} \mathbf{z})$.
3. Determine $\mathbf{b}_j = \lambda \mathbf{z}$.

With the QR decomposition of $\Delta_j(\mathbf{x}_0)$ the system of the first step becomes $\mathbf{R}_{j+1}^H \mathbf{R}_{j+1} \mathbf{z} = \mathbf{W}_{j+1}^H$.

To determine the residual norm, let us note the following result that arises from inserting the lower part of (3.51) into its upper part,

$$\|\mathbf{r}_j\|^2 = \mathbf{b}_j^H \Delta_j(\mathbf{x}_0)^H \Delta_j(\mathbf{x}_0) \mathbf{b}_j = \mathbf{b}_j^H \mathbf{W}_{j+1}^H \lambda = \lambda.$$

Hence $\|\mathbf{r}_j\| = \sqrt{\lambda}$.

We have tested newer solution methods to solve the constrained least squares problem (3.49) such as direct elimination and the nullspace method. However system (3.49) is so ill conditioned that these methods usually do not have any impact on the accuracy of the solution.

The observations made so far are summed up into a function RRE in Algorithm 3.4.

Algorithm 3.4 Reduced Rank Extrapolation.

```

function RRE(in:  $g(\cdot)$ ,  $\mathbf{x}_0$ ,  $\varepsilon$ ): vector;
   $\mathbf{x}_1 := g(\mathbf{x}_0)$ ;  $\delta\mathbf{x}_0 := \mathbf{x}_1 - \mathbf{x}_0$ ;  $\mathbf{c}_{-1} := \mathbf{0}$ ;
   $\mathbf{R}_0 := \mathbf{0}$ ;  $\mathbf{Q}_0 := \mathbf{0}$ ;
   $\mathbf{R}_1 := \|\delta\mathbf{x}_0\|$ ;  $\mathbf{Q}_1 := \delta\mathbf{x}_0/r_{1,1}$ ;
   $\rho_0 := \|\delta\mathbf{x}_0\|$ ;
   $j := 0$ ;
  while  $\rho_j > \rho_0 \varepsilon$  do
     $\mathbf{x}_{j+2} := g(\mathbf{x}_{j+1})$ ;           {new sequence vector}
     $\delta\mathbf{x}_{j+1} := \mathbf{x}_{j+2} - \mathbf{x}_{j+1}$ ;   {new difference vector}
     $\mathbf{Q}_{j+2}\mathbf{R}_{j+2} := \Delta_{j+1}(\mathbf{x}_0)$ ; {successive QR decomp.}
     $\bar{\mathbf{z}} := \mathbf{R}_{j+2}^{-H}[1, \dots, 1]^H$ ;       {step 1}
     $\mathbf{z} := \mathbf{R}_{j+2}^{-1}\bar{\mathbf{z}}$ ;
     $\lambda := 1/([1, \dots, 1] \cdot \mathbf{z})$ ;       {step 2}

    {step 3: transform coefficients}
     $\gamma_{j,j+1} := z_{j+2}\lambda$ ;
    for  $i := j - 1$  to 0 by -1 do
       $\gamma_{i,j+1} := \gamma_{i+1,j+1} + z_{i+2}\lambda$ ;
    end;
     $\rho_{j+1} := \sqrt{\lambda}$ ;           {residual norm}
     $j := j + 1$ ;
  end;
   $\mathbf{s}_j := \mathbf{x}_0 + \mathbf{Q}_j\mathbf{R}_j\mathbf{c}_j$ ;
  RRE :=  $\mathbf{s}_j$ 
end RRE;

```

From (3.19) we know that $\Delta_j^2(\mathbf{x}_0) = \mathbf{A}\Delta_j(\mathbf{x}_0)$. By the proof of Lemma 3.6 we know that $\Delta_j(\mathbf{x}_0)$ is a basis of the Krylov space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$. By Lemma 3.4 we know that it is the correction space. Solving (3.50) is therefore equivalent to minimizing $\|\mathbf{r}_j\|$ over the Krylov space $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ transformed by a left hand side multiplication with the system matrix \mathbf{A} . This is exactly what GMRES does. We have proven the following theorem.

EQUIVALENT
KRYLOV
SPACE
METHOD:
GMRES

Theorem 3.19 *The reduced rank extrapolation method (RRE) is mathematically equivalent to GMRES applied to System (3.2). \square*

NONLINEAR
BEHAVIOUR

Using(3.50) we see by the proof of Lemma 3.4 that RRE obtains \mathbf{s}_k by

$$\mathbf{s}_k = \mathbf{x}_0 + \Delta_{k-1}(\mathbf{x}_0)\Delta_{k-1}^2(\mathbf{x}_0)^+\delta\mathbf{x}_0.$$

Reusing \mathbf{s}_k as a new \mathbf{x}_0 — as we do when dealing with nonlinear systems of equations — we obtain

$$\mathbf{s}_{j+1} = \mathbf{s}_j - \Delta_{k-1}(\mathbf{s}_j)\Delta_{k-1}^2(\mathbf{s}_j)^+\delta\mathbf{s}_j.$$

This is equivalent to Henrici's method (3.33), where the linear system of rank $k < n$ is solved by a least squares approach. Therefore RRE deserves to be called a *natural generalization of Henrici's method*.

3.5 Epsilon Algorithms

Epsilon algorithms are the second big group of algorithms of vector extrapolation methods. The story of these algorithms begins with an important result independently found by Schmidt in 1941 [34] and Shanks in 1955 [35]. They formulated a transformation of a vector sequence for the acceleration of its convergence based on the quotient of two determinants. This transformation is nowadays known as *Schmidt-Shanks transform*. However, computing determinants must be avoided in numerical analysis. But already in 1956 Wynn [40] discovered a recursive scheme to make the computation of the Schmidt-Shanks transform feasible, the so called *epsilon algorithm*. This algorithm was extended by Wynn himself to the *vector epsilon algorithm*, which was observed to perform well on the acceleration of vector sequences. Unfortunately up to day a representation of this algorithm in terms of a determinantal formula

was not found. Thus it is not known, if and what kind of linear system is solved in the background of this algorithm. This problem was addressed by Brezinsky in 1975 [6] when he developed the *topological epsilon algorithm*. This algorithm is known to be mathematically equivalent to the Biconjugate Gradient method (see Section 2.4.6) in the case of a linearly generated vector sequence.

3.5.1 Schmidt-Shanks Transform

The Schmidt-Shanks transform was independently developed by Schmidt [34] and Shanks [35]. It transforms a given (nonlinear) *scalar* sequence $\{x_0, x_1, \dots\}$ into a sequence $\{s_0, s_1, \dots\}$ converging faster to the same limit or anti limit s as the original sequence. It is a generalization of Steffensen's method.

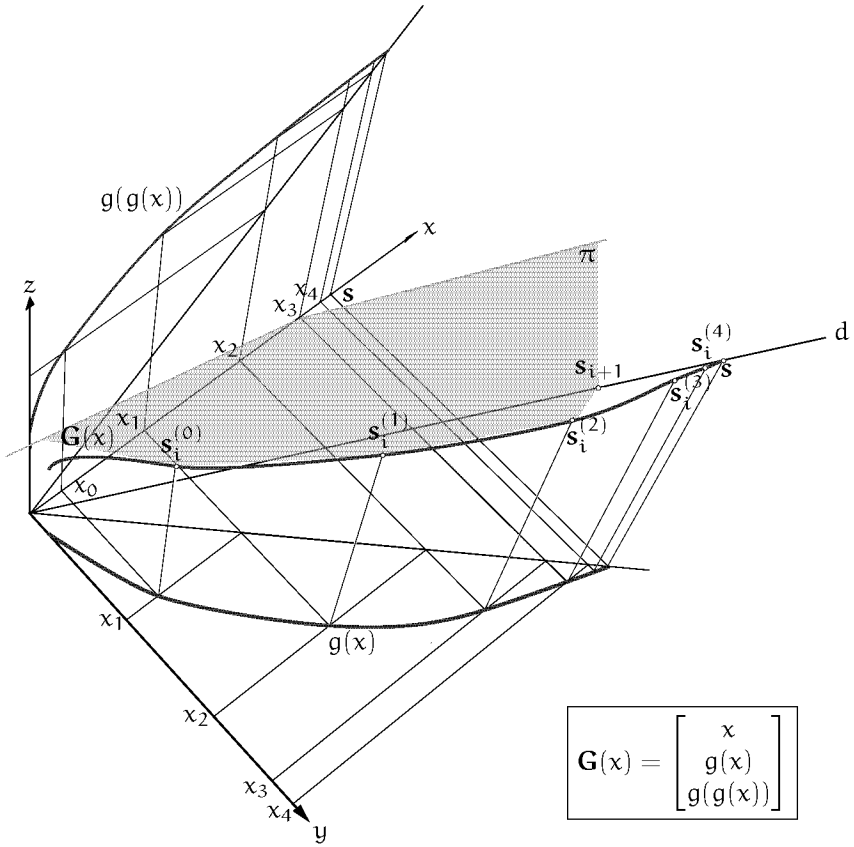
The main idea of the Schmidt-Shanks transform is to interpret the given sequence as an operator in d dimensional vector space as follows: Let g be the unknown generating functional of the sequence $\{x_j\}$, $j = 0, 1, 2, \dots$ given a starting value x_0 and $x_{j+1} := g(x_j)$. We use the notation for the repeated application of a functional introduced by (3.9) on g and define the operator

$$\mathbf{G}(x) := [x, g(x), \dots, g^{d-1}(x)]^T. \quad (3.52)$$

Note that this operator has a *scalar* argument! By construction this operator has a fixed point $s := [s, \dots, s]^T$ if g has a fixed point s .

We start with an initial guess $x_0 := s_i$ for the fixed point s and construct a hyperplane π defined by d successive points of this operator. These points are defined by $s_i^{(0)} := \mathbf{G}(x_0)$, $s_i^{(1)} := \mathbf{G}(x_1)$, \dots , $s_i^{(d-1)} := \mathbf{G}(x_{d-1})$. With

Figure 3.5 Three dimensional plot of figures involved in one step of a Schmidt-Shanks transform for $d = 3$.



these values the hyperplane becomes

$$\pi : \mathbf{x} = \mathbf{s}_i^{(0)} + \sum_{j=0}^{d-2} \lambda_j (\mathbf{s}_i^{(j)} - \mathbf{s}_i^{(j+1)}) \tag{3.53}$$

$$= \begin{bmatrix} x_0 & \delta x_0 & \dots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ x_{d-1} & \delta x_{d-1} & \dots & \delta x_{2d-3} \end{bmatrix} \begin{bmatrix} 1 \\ \lambda_0 \\ \vdots \\ \lambda_{d-2} \end{bmatrix}$$

A step of the Schmidt-Shanks transform is based on the hypothesis that the curve of the operator \mathbf{G} lies in the hyperplane π . Therefore the new approximation s_{i+1} is determined by the intersection of the hyperplane with the space diagonal $d : x = \lambda[1, \dots, 1]^T$, i.e.

$$\begin{bmatrix} s_{i+1} \\ \vdots \\ s_{i+1} \end{bmatrix} \stackrel{!}{=} \begin{bmatrix} x_0 & \delta x_0 & \dots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ x_{d-1} & \delta x_{d-1} & \dots & \delta x_{2d-3} \end{bmatrix} \begin{bmatrix} 1 \\ \lambda_0 \\ \vdots \\ \lambda_{d-2} \end{bmatrix}.$$

The unknowns are $s_{i+1}, \lambda_0, \dots, \lambda_{d-2}$. The rearranged system becomes

$$\begin{bmatrix} 1 & \delta x_0 & \dots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ 1 & \delta x_{d-1} & \dots & \delta x_{2d-3} \end{bmatrix} \begin{bmatrix} s_{i+1} \\ \lambda_0 \\ \vdots \\ \lambda_{d-2} \end{bmatrix} = \begin{bmatrix} x_0 \\ \vdots \\ x_{d-1} \end{bmatrix}.$$

This system is solved for the first unknown by Cramer's rule. We get the original (although transposed) formulation of the Schmidt-Shanks transform

$$s_{i+1} = e_{d-1}(x_0) := \frac{\begin{vmatrix} x_0 & \delta x_0 & \dots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ x_{d-1} & \delta x_{d-1} & \dots & \delta x_{2d-3} \end{vmatrix}}{\begin{vmatrix} 1 & \delta x_0 & \dots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ 1 & \delta x_{d-1} & \dots & \delta x_{2d-3} \end{vmatrix}}. \quad (3.54)$$

For the case $d = 2$ this transform determines a line that intersects with the diagonal in the (x, y) -Plane, in which case the transform becomes

$$s_{i+1} = e_1(x_0) = \frac{\begin{vmatrix} x_0 & \delta x_0 \\ x_1 & \delta x_1 \end{vmatrix}}{\begin{vmatrix} 1 & \delta x_0 \\ 1 & \delta x_1 \end{vmatrix}} = \frac{x_0 \delta x_0 - x_1 \delta x_0}{\delta x_1 - \delta x_0},$$

which is nothing else than Steffensen's method (compare (3.21)). This is why the Schmidt-Shanks transform is a natural generalization of Steffensen's method. This generalization reveals that fixed points of linear functionals are computed exactly by the Schmidt-Shanks transform for $d = 2$.

Figure 3.5 shows a three dimensional plot of all the figures involved in one step of a Schmidt-Shanks transform for $d = 3$.

REPRESENTATION IN TERMS OF HANKEL DETERMINANTS.

Definition 3.20 *A determinant of the following form*

$$H_k(x_n) := \begin{vmatrix} x_n & x_{n+1} & \cdots & x_{n+k-1} \\ x_{n+1} & x_{n+2} & \cdots & x_{n+k-2} \\ \vdots & \vdots & & \vdots \\ x_{n+k-1} & x_{n+k} & \cdots & x_{n+2k-2} \end{vmatrix}$$

is called Hankel determinant.

The Schmidt-Shanks transform can be expressed as a ratio of Hankel determinants. Let us sum up the columns of the numerator of (3.54) subsequently. Let us subtract subsequent rows of the denominator and develop the resulting determinant by the upper left element which remains 1. Thus the Schmidt-Shanks transform becomes

$$e_{d-1}(x_0) := \frac{\begin{vmatrix} x_0 & x_1 & \cdots & x_{d-1} \\ \vdots & \vdots & & \vdots \\ x_{d-1} & x_{d-2} & \cdots & x_{2d-2} \end{vmatrix}}{\begin{vmatrix} \delta^2 x_0 & \cdots & \delta^2 x_{d-2} \\ \vdots & & \vdots \\ \delta^2 x_{d-2} & \cdots & \delta^2 x_{2d-4} \end{vmatrix}} = \frac{H_d(x_0)}{H_{d-1}(\delta^2 x_0)}.$$

KERNEL AND KERNEL FUNCTIONALS

Definition 3.21 *The set of all sequences, for which a sequence transformation finds the exact limit (or anti limit) s within one step, is called kernel of the sequence transformation.*

Brezinski and Redivo Zaglia[7, p. 79] have given the kernel for the Schmidt-Shanks transform using determinants of size k .

Definition 3.22 *We call an iteration functional g kernel functional of an acceleration method, if the fixed point s of that functional is exactly determined by that acceleration method within one step regardless of the choice of the starting value x_0 .*

A sufficient condition for g to be a kernel functional of the Schmidt-Shanks transform using determinants of size k is that the functional has exactly one fixed point and that the graph of the corresponding operator G defined by (3.52) lies in a hyperplane in a vector space of dimension k for all values of x . Due to the help of Prof. Jörg Waldvogel, who had the basic idea of how to find such a functional, the following theorem was formulated. It describes an infinite class of nonlinear scalar functionals g for which the operator given by (3.52) is planar in k dimensional space.

Theorem 3.23 *Let $k \in \mathbb{N}$, $\mathcal{M} \subset \mathbb{R}$ and $\varphi : \mathcal{M} \rightarrow \mathcal{M}, x \mapsto x^{k-1}$ be an invertible functional. Let $h : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto cx + d$ be a linear scalar functional. The coefficients c and d are chosen such that h maps all elements of \mathcal{M} to any subset of \mathcal{M} .*

Then the following expression for $g(x)$ is defined,

$$g(x) := \varphi \circ h \circ \varphi^{-1}(x) = \varphi(h(\varphi^{-1}(x))) = \left(cx^{\frac{1}{k-1}} + d \right)^{k-1}. \quad (3.55)$$

The graph of $G(x) := [x, g(x), \dots, g^{k-1}(x)]^T$ is planar in \mathcal{M}^k .

Proof. To prove the statement, we have to find a hyperplane $\pi: \sum_{i=0}^{k-1} a_i x_i = 1$, containing \mathbf{G} . $\mathbf{G}(x)$ is contained in the hyperplane π if the following equation holds,

$$\sum_{i=0}^{k-1} a_i g^i(x) - 1 = 0. \quad (3.56)$$

Let $y := x^{\frac{1}{k-1}}$. With the given expression for g , we obtain

$$g^i(x) = \left(c^i y + \sum_{j=0}^{i-1} c^j d \right)^{k-1}.$$

Therefore the left hand side of (3.56) is a polynomial in y of degree $k-1$. The coefficients of the left hand side of (3.56) have to vanish. This results in a system of linear equations with k unknowns a_0, \dots, a_{k-1} which has always at least one solution. Thus we have found a hyperplane containing \mathbf{G} , which completes the proof. \square

Thus Theorem 3.23 guarantees the first condition for g defined by (3.55) being a kernel functional. The second condition — g having exactly one fixed point — is established by the following lemma.

Lemma 3.24 *Let $\mathcal{M} \subset \mathbb{R}$ and $\varphi: \mathcal{M} \rightarrow \mathcal{M}$, $\varphi(x) = x^{k-1}$ be an invertible nonlinear scalar functional. Let $\tilde{s} \in \mathcal{M}$ be a fixed point of $h: \mathbb{R} \rightarrow \mathbb{R}$, a linear scalar functional with $h' \neq 1$.*

Then g as defined by (3.55) has exactly one fixed point in \mathcal{M} .

Proof. Let $\tilde{s} \in \mathcal{M}$ be the only fixed point of h . This is only possible if $h' \neq 1$, because a linear functional with $h' = 1$ is parallel to the bisecting line $y = x$. Such a function has either infinitely many fixed points or none.

The following equation holds,

$$\tilde{s} = h(\tilde{s}).$$

Since φ is invertible in \mathcal{M} , a unique value $s \in \mathcal{M}$ can be found such that $\varphi^{-1}(s) = \tilde{s}$. The above given equation can be transformed into

$$\varphi^{-1}(s) = h(\varphi^{-1}(s)),$$

and further into

$$s = \varphi(h(\varphi^{-1}(s))) = g(s).$$

Because φ is invertible, s is the only fixed point of g . \square

By combining Theorem 3.23 and Lemma 3.24 the sufficient condition for g defined by (3.55) being a kernel function of the Schmidt-Shanks transform using determinants of dimension k is satisfied. The following example illustrates the behavior of one instance of the described class of kernel functions for the Schmidt-Shanks transform for $k = 4$.

Example 3.25 Let g be

$$g(x) := \left(\frac{1}{3}x^{\frac{1}{3}} + 2 \right)^3.$$

Using this function the following equation holds regardless of the value of x ,

$$\mathbf{G}(x)^H \begin{bmatrix} 1 \\ -39 \\ 351 \\ -729 \end{bmatrix} = -11232.$$

This means that $\mathbf{G}(x)$ is contained in the four-dimensional hyperplane

$$x_1 - 39x_2 + 351x_3 - 729x_4 = -11232.$$

The function g has exactly one fixed point at 27, which the Schmidt-Shanks transform with $k = 4$ is able to find within one step regardless of the choice of the starting value. For $x_0 = 0$ we get

$$27 \doteq \left| \begin{array}{cccc|cccc} 0 & 8 & 18.96 & 24.11 & 1 & 1 & 1 & 1 \\ 8 & 10.96 & 5.15 & 1.90 & 8 & 10.96 & 5.15 & 1.90 \\ 10.96 & 5.15 & 1.90 & 0.66 & 10.96 & 5.15 & 1.90 & 0.66 \\ 5.15 & 1.90 & 0.66 & 0.22 & 5.15 & 1.90 & 0.66 & 0.22 \end{array} \right|,$$

for $x_0 = 3$ we get

$$27 \doteq \left| \begin{array}{cccc|cccc} 3 & 15.27 & 22.59 & 25.47 & 1 & 1 & 1 & 1 \\ 12.27 & 7.32 & 2.88 & 1.01 & 12.27 & 7.32 & 2.88 & 1.01 \\ 7.32 & 2.88 & 1.01 & 0.34 & 7.32 & 2.88 & 1.01 & 0.34 \\ 2.88 & 1.01 & 0.34 & 0.12 & 2.88 & 1.01 & 0.34 & 0.12 \end{array} \right|.$$

Brezinski [6] notes that instead of $s_i^{(0)}$ any other point of the hyperplane could be used as the anchor of the hyperplane of (3.53). In particular he uses the last vector $s_i^{(d-1)}$. This choice results in the linear system

$$\begin{bmatrix} s_{i+1} \\ \vdots \\ s_{i+1} \end{bmatrix} = \begin{bmatrix} x_{d-1} & \delta x_0 & \cdots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ x_{2d-2} & \delta x_{d-1} & \cdots & \delta x_{2d-3} \end{bmatrix} \begin{bmatrix} 1 \\ \lambda_0 \\ \vdots \\ \lambda_{d-2} \end{bmatrix}$$

and finally a variant of the Schmidt-Shanks transform

$$s_{i+1} = \tilde{e}_{d-1}(x_0) := \frac{\left| \begin{array}{cccc} x_{d-1} & \delta x_0 & \cdots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ x_{2d-2} & \delta x_{d-1} & \cdots & \delta x_{2d-3} \end{array} \right|}{\left| \begin{array}{cccc} 1 & \delta x_0 & \cdots & \delta x_{d-2} \\ \vdots & \vdots & & \vdots \\ 1 & \delta x_{d-1} & \cdots & \delta x_{2d-3} \end{array} \right|}.$$

A similar transformation of the determinant in the numerator as before yields

$$\bar{e}_{d-1}(x_0) = \frac{H_d(x_0)}{H_{d-1}(\delta^2 x_0)}.$$

3.5.2 Scalar Epsilon Algorithm

Already in 1956 Wynn discovered that there is a recursive scheme to determine the quotient of determinants for the Schmidt-Shanks transform. Let $\delta\epsilon_j^{(i)}$ denote the difference $\epsilon_j^{(i+1)} - \epsilon_j^{(i)}$. The following scheme describes his *scalar epsilon algorithm (SEA)*.

$\epsilon_{-1}^{(i)} := 0,$	$\forall i \geq 0,$
$\epsilon_0^{(i)} := x_i,$	$\forall i \geq 0,$
$\epsilon_{j+1}^{(i)} := \epsilon_{j-1}^{(i+1)} + \frac{1}{\delta\epsilon_j^{(i)}},$	$\forall i, j \geq 0.$

The values $e_d()$ of the Schmidt-Shanks transform are related to the epsilon values as follows.

$$\epsilon_{2d}^{(j)} = e_d(x_j), \quad \text{and} \quad \epsilon_{2d+1}^{(j)} = \frac{1}{e_d(\delta x_j)}.$$

Wynn proved these relations in [40] by induction using the expansion of the quotient of two determinants by Schweins [1, pp. 106 – 109].

The epsilon values $\epsilon_j^{(i)}$ are commonly displayed in a table of which the index i denotes the row and the index j denotes the column. However, for a progressive implementation of the algorithm storage of subsequent diagonals is convenient. Baring this consideration in mind a distorted epsilon table is depicted in Figure 3.6 in such a way that the diagonals appear as rows. This way the

EPSILON
TABLE

Figure 3.6 Epsilon Table: Epsilon values involved in the computation of $e_2(x_0)$ and their dependencies. In a progressive computation only the most recent two diagonals (appearing as rows in this figure) have to be stored.

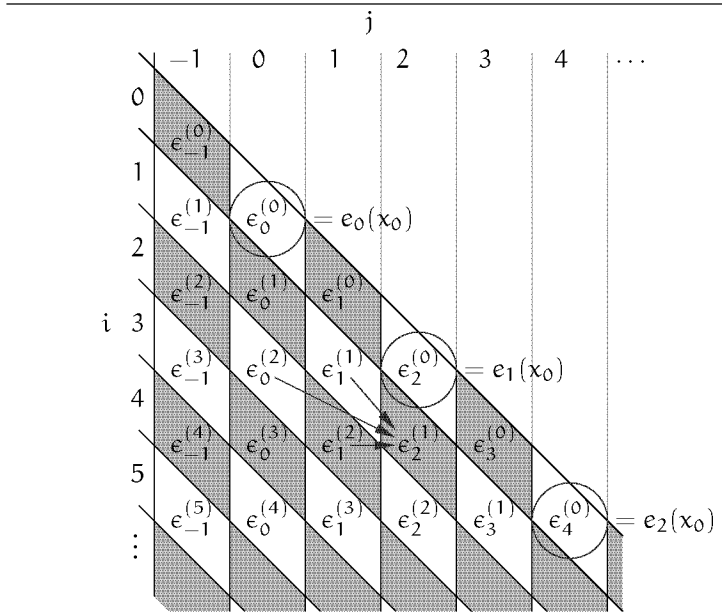


figure clearly reveals the observation that only two diagonals of the epsilon table have to be remembered to compute the next diagonal of the table.

IMPLEMENTATION

A progressive implementation of the scalar epsilon algorithm is given in Algorithm 3.5. In this algorithm we use the notation $\epsilon^{[d]} := [\epsilon_1^{[d]}, \dots, \epsilon_{d+2}^{[d]}]$, $\epsilon_j^{[d]} = \epsilon_{j-2}^{(d-j+2)}$ to represent a diagonal of the epsilon table. Diagonals are numbered by the lower index of the last element of the diagonal.

Algorithm 3.5 Progressive implementation of the scalar epsilon algorithm.

function SEA(g, x_0, ε):**vector**;

$s_0 := x_0$;

$\rho_0 := g(s_0) - s_0$;

$e^{[0]} := [0, x_0]$;

$d := 0$;

while $|\rho_d| > \varepsilon |\rho_0|$ **do**

$x_{d+1} := g(x_d)$;

$e^{[d+1]} := [0, x_{d+1}]$;

for $j := 0$ **to** d **do**

$\hat{\delta} \varepsilon_j^{(d-j)} := e_{j+2}^{[d+1]} - e_{j+2}^{[d]}$;

$e_{j+3}^{[d+1]} := e_{j+1}^{[d]} + [\hat{\delta} \varepsilon_j^{(d-j)}]^{-1}$

end;

if $d + 1 \bmod 2 = 0$ **then**

$s_{d+1} := e_{d+3}^{[d+1]}$;

$\rho_{d+1} := g(s_{d+1}) - s_{d+1}$

end;

$d := d + 1$

end;

SEA := s_d

end SEA;

3.5.3 Generalized Schmidt-Shanks Transform

To extend the Schmidt-Shanks transform to systems of linear equations Brezinski [6] introduced the notation of the *vector determinant*.

Definition 3.26 Let \mathbf{N} be a block matrix whose first block row contains only vectors of the same size, i.e.

$$\mathbf{N} := \begin{bmatrix} \mathbf{a}_{1,1} & \dots & \mathbf{a}_{1,n} \\ \alpha_{2,1} & \dots & \alpha_{2,n} \\ \vdots & & \vdots \\ \alpha_{n,1} & \dots & \alpha_{n,n} \end{bmatrix}.$$

Let $\mathbf{N}_{i,j}$ denote the matrix that arises by deleting the i^{th} row and j^{th} column of \mathbf{N} .

We define the vector determinant of \mathbf{N} as a linear combination of the vectors $\mathbf{a}_{1,1}, \dots, \mathbf{a}_{1,n}$ the coefficients of which are signed cofactors, i.e.

$$\mathbf{det}(\mathbf{N}) := \sum_{k=1}^n (-1)^{k+1} \mathbf{det}(\mathbf{N}_{1,k}) \mathbf{a}_{1,k}.$$

Thus the vector determinant is determined by developing the determinant of the matrix \mathbf{N} in the classical sense with respect to its first row.

When we use $\mathcal{K}_j(\mathbf{G}^H, \mathbf{a})$ as projection space with \mathbf{a} being an arbitrarily chosen vector, System (3.15) becomes

$$\begin{bmatrix} 1 & \dots & 1 \\ \mathbf{a}^H \delta \mathbf{x}_0 & \dots & \mathbf{a}^H \delta \mathbf{x}_j \\ \vdots & & \vdots \\ \mathbf{a}^H \delta \mathbf{x}_{j-1} & \dots & \mathbf{a}^H \delta \mathbf{x}_{2j-1} \end{bmatrix} \mathbf{b}_j = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (3.57)$$

Let the matrix of this system be denoted by \mathbf{M} and let $\mathbf{M}_{i,j}$ denote the submatrix of \mathbf{M} obtained by deleting the i^{th} row and the j^{th} column from \mathbf{M} . Solving (3.57) by Cramer's rule for $\beta_{k,j}$ results in

$$\beta_{k,j} = (-1)^{k+1} \frac{\mathbf{det}(\mathbf{M}_{1,k})}{\mathbf{det}(\mathbf{M})}.$$

Using (3.13) to determine the approximate solution \mathbf{s}_j we obtain the *generalized Schmidt-Shanks transform*,

$$\mathbf{s}_j = \mathbf{e}_j(\mathbf{x}_0) := \frac{1}{\mathbf{det}(\mathbf{M})} \begin{vmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_j \\ \mathbf{a}^H \delta \mathbf{x}_0 & \dots & \mathbf{a}^H \delta \mathbf{x}_j \\ \vdots & & \vdots \\ \mathbf{a}^H \delta \mathbf{x}_{j-1} & \dots & \mathbf{a}^H \delta \mathbf{x}_{2j-1} \end{vmatrix}.$$

We already know that $\det(\mathbf{M})$ is a Hankel determinant, in particular $H_j(\mathbf{a}^H \delta^2 \mathbf{x}_0)$. Let us define

$$\tilde{H}_j(\mathbf{a}, \mathbf{x}_0) := \begin{vmatrix} \mathbf{x}_0 & \cdots & \mathbf{x}_j \\ \mathbf{a}^H \delta \mathbf{x}_0 & \cdots & \mathbf{a}^H \delta \mathbf{x}_j \\ \vdots & & \vdots \\ \mathbf{a}^H \delta \mathbf{x}_{j-1} & \cdots & \mathbf{a}^H \delta \mathbf{x}_{2j-1} \end{vmatrix}.$$

Note that this definition implies $H_j(\mathbf{a}^H \mathbf{x}_0) = \mathbf{a}^H \tilde{H}_j(\mathbf{a}, \mathbf{x}_0)$. With this definition the first variant of the generalized Schmidt-Shanks transform becomes

$$e_j(\mathbf{x}_0) = \frac{\tilde{H}_{j+1}(\mathbf{a}, \mathbf{x}_0)}{H_j(\mathbf{a}^H \delta^2 \mathbf{x}_0)}. \quad (3.58)$$

By a similar idea as for the scalar epsilon algorithm Brezinski defines the second variant of the generalized Schmidt-Shanks transform by

$$\tilde{e}_j(\mathbf{x}_0) = \frac{1}{H_j(\mathbf{a}^H \delta^2 \mathbf{x}_0)} \begin{vmatrix} \mathbf{x}_j & \cdots & \mathbf{x}_{2j} \\ \mathbf{a}^H \delta \mathbf{x}_0 & \cdots & \mathbf{a}^H \delta \mathbf{x}_j \\ \vdots & & \vdots \\ \mathbf{a}^H \delta \mathbf{x}_{j-1} & \cdots & \mathbf{a}^H \delta \mathbf{x}_{2j-1} \end{vmatrix}.$$

This is another vector extrapolation method that corresponds to the projection space $\mathcal{K}_j(\mathbf{G}^H, (\mathbf{G}^{-j})^H \mathbf{a})$.

3.5.4 Topological Epsilon Algorithm

Since determinants are not always easy to compute, a recursive algorithm for computing the quotient of the determinants of the generalized Schmidt-Shanks transform similar to the epsilon algorithm would be desirable. To develop such an algorithm one can rely on the epsilon algorithm, but one has to find a way to invert

vectors. Brezinski [6] uses ordered pairs of vectors (\mathbf{x}, \mathbf{y}) to define

$$\mathbf{y}^{-1} := \frac{\mathbf{x}}{\mathbf{y}^H \mathbf{x}} \quad \text{and} \quad \mathbf{x}^{-1} := \frac{\mathbf{y}}{\mathbf{y}^H \mathbf{x}}.$$

Thus we have

$$(\mathbf{x}^{-1})^H \mathbf{x} = 1 \quad \text{and} \quad \mathbf{y}^H \mathbf{y}^{-1} = 1.$$

Let $\hat{\delta} \epsilon_j^{(i)}$ denote the difference $\epsilon_j^{(i+1)} - \epsilon_j^{(i)}$. Similar to Wynn and his epsilon algorithm Brezinski proved that the sequence transformation defined by (3.58) on the sequence of vectors $\mathbf{x}_0, \dots, \mathbf{x}_j$ can be obtained recursively by the following scheme, which is known as the *Topological Epsilon Algorithm (TEA1)*.

$$\begin{aligned} \epsilon_{-1}^{(i)} &= 0, \epsilon_0^{(i)} = \mathbf{x}_i, & \forall i \geq 0, \\ \epsilon_{2j+1}^{(i)} &= \epsilon_{2j-1}^{(i+1)} + \left[\hat{\delta} \epsilon_{2j}^{(i)} \right]^{-1}, & \forall i, j \geq 0, \\ \epsilon_{2j+2}^{(i)} &= \epsilon_{2j}^{(i+1)} + \left[\hat{\delta} \epsilon_{2j+1}^{(i)} \right]^{-1}, & \forall i, j \geq 0, \\ \text{with } \left[\hat{\delta} \epsilon_{2j}^{(i)} \right]^{-1} &= \frac{\mathbf{a}}{\mathbf{a}^H (\hat{\delta} \epsilon_{2j}^{(i)})}, \quad \mathbf{a}^{-1} = \frac{\hat{\delta} \epsilon_{2j}^{(i)}}{\mathbf{a}^H (\hat{\delta} \epsilon_{2j}^{(i)})}, \\ \left[\hat{\delta} \epsilon_{2j+1}^{(i)} \right]^{-1} &= \frac{\mathbf{a}^{-1}}{(\hat{\delta} \epsilon_{2j+1}^{(i)})^H \mathbf{a}^{-1}} = \frac{\hat{\delta} \epsilon_{2j}^{(i)}}{(\hat{\delta} \epsilon_{2j+1}^{(i)})^H \hat{\delta} \epsilon_{2j}^{(i)}}. \end{aligned}$$

(3.59)

Very similar to the scalar case the corresponding values of the generalized Schmidt-Shanks transform are

$$\epsilon_{2j}^{(i)} = \mathbf{e}_j(\mathbf{x}_i), \quad \text{and} \quad \epsilon_{2j+1}^{(i)} = \frac{\mathbf{a}}{\mathbf{a}^H \mathbf{e}_j(\delta \mathbf{x}_i)}.$$

The second variant of the generalized Schmidt-Shanks transform yields a second topological epsilon algorithm (TEA2).

$$\begin{array}{l}
 \mathbf{e}_{-1}^{(i)} = 0, \mathbf{e}_0^{(i)} = \mathbf{x}_i, \quad \forall i \geq 0, \\
 \mathbf{e}_{2j+1}^{(i)} = \mathbf{e}_{2j-1}^{(i+1)} + \left[\hat{\delta} \mathbf{e}_{2j}^{(i)} \right]^{-1}, \quad \forall i, j \geq 0, \\
 \mathbf{e}_{2j+2}^{(i)} = \mathbf{e}_{2j}^{(i+1)} + \left[\hat{\delta} \mathbf{e}_{2j+1}^{(i)} \right]^{-1}, \quad \forall i, j \geq 0, \\
 \text{with } \left[\hat{\delta} \mathbf{e}_{2j}^{(i)} \right]^{-1} = \frac{\mathbf{a}}{\mathbf{a}^H (\hat{\delta} \mathbf{e}_{2j}^{(i)})}, \quad \mathbf{a}^{-1} = \frac{\hat{\delta} \mathbf{e}_{2j}^{(i+1)}}{\mathbf{a}^H (\hat{\delta} \mathbf{e}_{2j}^{(i+1)})}, \\
 \left[\hat{\delta} \mathbf{e}_{2j+1}^{(i)} \right]^{-1} = \frac{\mathbf{a}^{-1}}{(\hat{\delta} \mathbf{e}_{2j+1}^{(i)})^H \mathbf{a}^{-1}} = \frac{\hat{\delta} \mathbf{e}_{2j}^{(i+1)}}{(\hat{\delta} \mathbf{e}_{2j+1}^{(i)})^H \hat{\delta} \mathbf{e}_{2j}^{(i+1)}}.
 \end{array}$$

The corresponding values of the second variant of the generalized Schmidt-Shanks transform are

$$\mathbf{e}_{2j}^{(i)} = \tilde{\mathbf{e}}_j(\mathbf{x}_i), \quad \text{and} \quad \mathbf{e}_{2j+1}^{(i)} = \frac{\mathbf{a}}{\mathbf{a}^H \tilde{\mathbf{e}}_j(\delta \mathbf{x}_i)}.$$

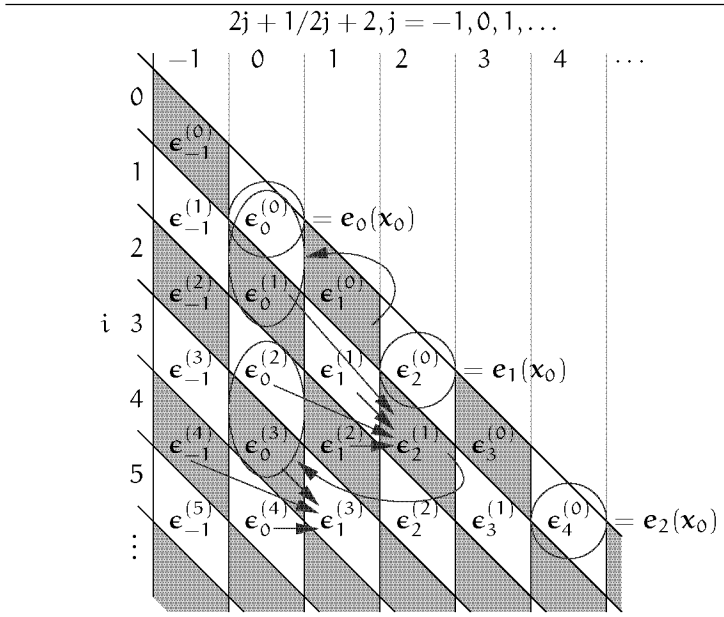
Since TEA1 fits perfectly into our framework of vector extrapolation methods, we know already by Lemma 3.4 that its correction space is $\mathcal{K}_j(\mathbf{G}, \delta \mathbf{x}_0)$. By Lemma 3.6 we know that this space is equal to $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$. If we choose $\mathbf{a} = \mathbf{r}_0 = \delta \mathbf{x}_0$ the assumptions of (3.57) show that the projection space of TEA1 is $\mathcal{K}_j(\mathbf{G}^H, \mathbf{r}_0)$, which is equal to $\mathcal{K}_j(\mathbf{A}^H, \mathbf{r}_0)$ by very similar considerations as in the proof of Lemma 3.6. However, a Krylov space method using these two spaces satisfies Definition 2.25, which is the definition of the biconjugate gradient method. Thus we have proven the following theorem.

EQUIVALENT
 KRYLOV
 SPACE
 METHOD:
 BiCG

Theorem 3.27 *Under the assumptions (3.1) the topological epsilon algorithm as defined by Scheme (3.59) is equivalent to BiCG applied to System (3.2) if $\mathbf{a} = \delta \mathbf{x}_0$.* □

The implementation of the topological epsilon algorithm is similar to the implementation of the scalar epsilon algorithm. However, even and odd values of j have to be distinguished. The epsilon table for TEA1 in Figure 3.7

Figure 3.7 Epsilon Table for TEA1: Epsilon vectors involved in the computation of $e_2(x_0)$ and their dependencies. Although $e_2^{(1)}$ is dependent on epsilon vectors from three different diagonals, it is possible to store only the most recent two diagonals.



shows that epsilon vectors with even column index are dependent on values from two previous diagonals. However, a clever implementation will realize that the only information needed from the first of these are differences of even columns. Storing these difference vectors separately saves one half of the storage of that diagonal. Difference vectors $\delta e_j^{(i+1)}$ can replace $\delta e_j^{(i)}$ if they are computed after $e_{j+2}^{(i)}$ has been stored. At the end of an odd diagonal d the difference vector $\delta e_{d-1}^{(0)}$ has to be computed.

Algorithm 3.6 Progressive implementation of the first topological epsilon algorithm.

```

function TEA1( $g, x_0, \alpha, \varepsilon$ ):vector;
   $s_0 := x_0$ ;
   $r_0 := g(s_0) - s_0$ ;
   $E^{[0]} := [0, x_0]$ ;
   $d := 0$ ;
  while  $\|r_d\| > \varepsilon \|r_0\|$  do
     $x_{d+1} := g(x_d)$ ;
     $E^{[d+1]} := [0, x_{d+1}]$ ;
    for  $j := 0$  to  $d$  do
       $\hat{\delta}e_j^{(d-j)} := e_{j+2}^{[d+1]} - e_{j+2}^{[d]}$ ;
      if  $j \bmod 2 = 0$  then
        
$$\left[ \hat{\delta}e_j^{(d-j)} \right]^{-1} := \frac{\alpha}{\alpha^H \hat{\delta}e_j^{(d-j)}}$$

      else
        
$$\left[ \hat{\delta}e_j^{(d-j)} \right]^{-1} := \frac{\hat{\delta}e_{j-1}^{(d-j)}}{(\hat{\delta}e_j^{(d-j)})^H \hat{\delta}e_{j-1}^{(d-j)}}$$

      end;
       $e_{j+3}^{[d+1]} := e_{j+1}^{[d]} + \left[ \hat{\delta}e_j^{(d-j)} \right]^{-1}$ ;
    end;
    if  $d + 1 \bmod 2 = 0$  then
       $s_{d+1} := e_{d+3}^{[d+1]}$ ;
       $r_{d+1} := g(s_{d+1}) - s_{d+1}$ 
    end;
     $d := d + 1$ 
  end;
  TEA1 :=  $s_d$ 
end TEA1;

```

These dependencies are shown in Figure 3.7 by curved arrows. However, they have been neglected in the implementation of Algorithm 3.6 in order to provide a clear progressive version of the topological epsilon algorithm.

Gander, Golub and Gruntz [14] remark that — although the algorithm generates the same residual vectors as

the non symmetric Lanczos algorithm — it only needs to compute matrix-vector multiplications with \mathbf{G} and not with \mathbf{G}^H . However, we need to point out, that this advantage is compensated by a much higher memory requirement: To solve a $d \times d$ system of equations the storage of $4(d-1) - 1$ epsilon vectors and $\lfloor (2(d-1) - 1)/2 \rfloor$ difference vectors is needed, so a total of $5d - O(1)$ difference vectors. This is nearly 5 times the size of the system matrix, if it is a dense matrix. Furthermore as we will see in our conclusion, the numerical performance is rather bad, compared to a linear solver. This is why the topological epsilon algorithm is not a good choice, if the system matrix is known.

3.6 Conclusion

To conclude this chapter we present two examples to illustrate the numerical performance of vector extrapolation methods.

Example 3.28 The first example is a numerical experiment on a linear problem taken from (14). We compare the performance of linear Krylov space methods FOM, GMRES, BiCG and QMR with the performance of MPE, GMRES and TEA1. The problem is an elliptic partial differential equation

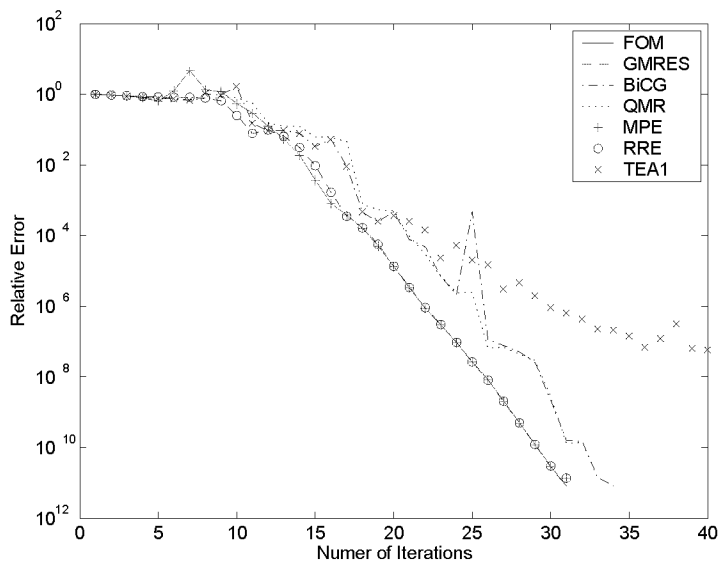
$$-\Delta u + \gamma \left(x \frac{\partial u}{\partial x} + y \frac{\partial u}{\partial y} \right) + \beta u = g \quad \text{with } \Omega = (0, 1) \times (0, 1).$$

with Dirichlet boundary conditions on $\partial\Omega$. We discretize this system using centered differences for both the first and second order derivatives on a uniform mesh. The mesh size $h = 1/32$ is used which corresponds to 961 unknowns. The boundary conditions are chosen such that the solution u to the discrete system is one everywhere.

Gander, Golub and Gruntz point out that “the parameters γ and β can be varied to make the problem more or less difficult to solve. The greater γ is chosen, the more unsymmetrical the system becomes, and the value of β makes the system more

or less positive definite.” (14). The basic iteration we use in our experiment is Gauss-Seidel, a stationary method generating a vector sequence according to (3.1). The Krylov methods solve the corresponding preconditioned system given by (3.2). The initial approximation $x_0 = s_0$ is the zero vector.

Figure 3.8 Linear Krylov space methods versus vector extrapolation methods for $n = 961$, $\gamma = 96$, $\beta = 0$. The basic iteration is Gauss-Seidel.



The development of the relative error of the numerical experiments are shown in Figure 3.8. This figure indicates the mathematical equivalence of FOM and MPE, GRMES and RRE, BiCG and TEA1.

It also shows that Krylov space methods are numerically more stable than the vector extrapolation methods. The most clear difference can be seen when comparing BiCG to TEA1. The convergence curve of TEA1 branches off the one of BiCG after about 20 iterations. TEA1 takes about twice as much iterations as BiCG to converge to the same tolerance (60 compared to 34). Thus it also consumes much more memory — as already mentioned before. Fortunately, Gander, Golub and Gruntz dis-

cuss in their paper how these drawbacks can be overcome by restarting the algorithms. Brezinski also has tried to solve these numerical problems by bordering methods. However, not even TEA1 shows a worse numerical behavior than its Krylov space counterpart, but also MPE and RRE branch off in the last two iterations. Even if these differences can be neglected in this example, cases can easily be found, in which the worse numerical behavior of MPE and RRE is apparent. On the other hand this example shows that there are linear cases for which MPE and RRE are able to compete with their Krylov space counterparts.

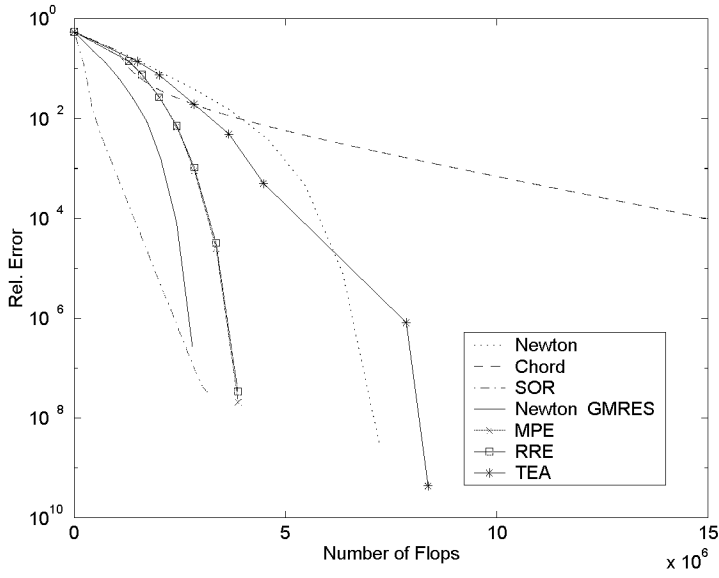
Brezinski and Redivo Zaglia point out that “the main interest of these algorithms mostly lies in the possibility of obtaining quadratic *derivative-free* methods for systems of nonlinear equations”[7, p. 307]. Let us therefore conclude with another example demonstrating the numerical behavior of the presented vector extrapolation methods for a nonlinear problem.

Example 3.29 Let us perform a numerical experiment on the Chandrasekhar H-equation. We use the same parameters $n = 100$, $c = 0.9999$ as in the concluding example of the introduction for this nonlinear problem. Newton’s method, the chord method, Newton-GMRES and SOR are compared with the vector extrapolation methods MPE, RRE and TEA1.

The vector extrapolation algorithms are wrapped by the error control framework developed for inexact Newton methods, given in Algorithm 1.8. However, neither the operator φ , a zero of which has to be found, nor the derivative $\mathbf{D}\varphi$ are provided as an argument. But they are replaced by an operator g , a fixed point of which has to be found. In our case one step of the chord method is used for g . In Algorithm 1.8 the assignment used to obtain the residual in lines 2 and 9 is replaced by $r_i := g(s_i) - s_i$. Lines 6 to 8 are replaced by the call of the particular vector extrapolation method. The same parameters for controlling the convergence behavior of Newton-GMRES are also used for these nonlinear vector extrapolation methods: $\theta_{\max} := 0.9999$, $\alpha := 2$, $\gamma := 0.9$. SOR uses $\omega := 1.39647$ as the relaxation parameter.

The development of the relative error of the several methods

Figure 3.9 The Chandrasekhar H-equation with $n = 100$ and $c = 0.9999$.



operating on the given problem is shown in Figure 3.9. Times and Flops measured on a Sun Sparc 336 Mhz with 3 GB main memory are shown in Table 3.3.

Table 3.3 Flops and execution times of several methods for solving the Chandrasekhar H-equation. Flops were counted with the `flops` command of Matlab 5.3. Times were measured in Matlab on a Sun SPARC 336 MHz with 3 GB main memory.

Method	Iterations	Mflops	Time(s)
Newton-GMRES	8	2.8	0.22
Newton	8	7.2	0.30
RRE	7	3.9	0.39
MPE	7	3.9	0.46
TEA1	7	8.4	1.21
Chord	262	27.8	3.23
SOR	17	3.2	3.55

As expected, both figure and table show that the vector extrapolation methods behave in a similar way as the Newton-GMRES method. TEA1 is the worst of the compared vector extrapolation methods. The drift off from Newton-GMRES can be explained by the startup time of the chord method, which has to perform a complete Gaussian elimination in the first step.

From the point of view of sequence acceleration we read from the table that MPE and RRE accelerate the chord method by a factor 8.2 (i.e. MPE and RRE need about 12 percent of the time the chord method takes), TEA1 accelerates the chord method by a factor 2.7 (which is 37.5 percent of the chord method - a similar value as published by Brezinski in (7) for another problem).

The performance of the vector extrapolation methods is dependent on the basic method for the fixed point iteration. For example the nonlinear Jacobi method results in worse convergence behavior. However, using SOR only saves the startup time and does not yield in a better convergence behavior than the chord method elsewhere.

Newton's method is by far the best method with respect to the Megaflop rate. However, Newton-GMRES and the vector extrapolation methods compete very well. Using larger problem sizes reveals that all of them compete even better and beat Newton's method with respect to the execution time.

We have to mention that the main advantage of the vector extrapolation methods consists in their not requiring the derivative of φ . In this example the amount of work for evaluating $\mathbf{D}\varphi$ is small (120817 flops). This is why this advantage does not count in the obtained results. However, there are larger problems where this advantage becomes important.

Chapter 4

Nonlinear Krylov Space Methods

4.1 Introduction

As has been shown in the last chapter, Krylov space methods are mathematically equivalent to vector extrapolation methods, when they are applied to solve linear systems of equations.

Moreover, we have found that vector extrapolation methods are generalizations of Henrici's method, when they are applied to systems of nonlinear equations. Henrici's method is a nonlinear solver comparable to Newton's method. It can be viewed as an accelerator of an existing iterative method, or it can be viewed as an effective solver of a system of nonlinear equations requiring the help of a nonlinear preconditioner. These two points of views are depicted in Figure 4.1. Our point of view is the second one.

Definition 4.1 *Given a system of nonlinear equations*

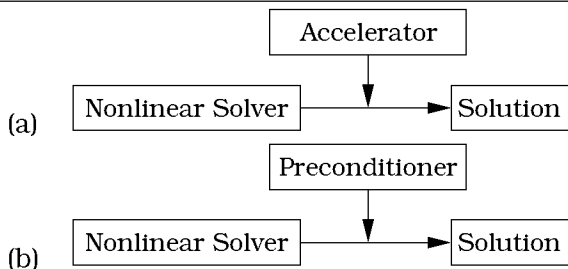
$$\varphi(x) = 0. \tag{4.1}$$

We call a nonlinear operator \mathbf{g} nonlinear preconditioner, if a solution s of (4.1) is a fixed point of \mathbf{g} and the Jacobian of \mathbf{g} has no eigenvalues equal to 1.

Note that a preconditioner usually should not worsen the convergence of a solver, when it is applied. This is why one would expect that the fixed point s of \mathbf{g} should also be a point of attraction. However, our methods do not require the preconditioner to converge to the fixed point, although the quality of a preconditioner might be measured with respect to this property.

A more important property is that the iteration $\mathbf{x}_{i+1} := \mathbf{g}(\mathbf{x}_i)$ should not stagnate in any direction. Therefore the Jacobian of \mathbf{g} is not allowed to have eigenvalues equal to 1.

Figure 4.1 Different points of view: (a) Henrici's method as accelerator for an ineffective nonlinear solver, (b) Henrici's method as nonlinear solver requiring the help of a nonlinear preconditioner.



In this chapter we develop the framework to combine vector extrapolation methods and Krylov space methods to *nonlinear* Krylov space methods. These methods are capable of solving nonlinear systems of equations by the help of a nonlinear preconditioner. They remain mathematically equivalent to their vector extrapolation counterparts and thus Henrici's method.

4.2 Arnoldi Type Methods

Arnoldi type methods make use of the Arnoldi process. In step j this process generates a new basis vector of the Krylov space by multiplying the system matrix \mathbf{A} with the last basis vector \mathbf{v}_j . Our problem is, that we do not know \mathbf{A} . We do not even know $\mathbf{G} = \mathbf{1} - \mathbf{A}$. Therefore the matrix vector multiplication $\mathbf{A}\mathbf{v}_j$ has to be replaced.

Our first aim is to develop a recursive scheme expressing the matrix vector multiplication $\mathbf{A}\mathbf{v}_j$ by known objects. The following lemma gives such a recursive scheme.

Lemma 4.2 *Given an arbitrary linearly generated vector sequence $\{\mathbf{x}_i\}$, for which $\mathbf{x}_{i+1} = \mathbf{G}\mathbf{x}_i + \mathbf{d}$ holds, where $\mathbf{G} \in \mathbb{R}^{n \times n}$ is a matrix the eigenvalues of which do not equal 1. We set $\mathbf{A} := \mathbf{1} - \mathbf{G}$ and $\delta\mathbf{x}_i := \mathbf{x}_{i+1} - \mathbf{x}_i$.*

Let $\mathbf{V}_{j+1} := [\mathbf{v}_0, \dots, \mathbf{v}_j] \in \mathbb{R}^{n \times j}$ be the orthonormal Arnoldi basis of the Krylov space $\mathcal{K}_j(\mathbf{A}, \delta\mathbf{x}_0)$. Let $\tilde{\mathbf{H}}_j \in \mathbb{R}^{(j+1) \times j}$ be the extended upper Hessenberg matrix for which the Arnoldi equation $\mathbf{V}_{j+1}\tilde{\mathbf{H}}_j = \mathbf{A}\mathbf{V}_j$ holds. Let \mathbf{h}_j denote the last column vector of $\tilde{\mathbf{H}}_j$.

Then using the following scheme

$$\begin{aligned}
 q_0 &:= -\frac{1}{\|\delta\mathbf{x}_0\|}, \\
 q_j &:= -\frac{1}{h_{j+1,j}}q_{j-1}, \\
 \mathbf{p}_0 &:= [1], \\
 \mathbf{p}_j &:= \frac{1}{h_{j+1,j}} \left(\mathbf{1}^{(j+1) \times (j+1)} - [\tilde{\mathbf{H}}_j | \mathbf{0}] \right) \left(\mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \right),
 \end{aligned} \tag{4.2}$$

the following equation holds

$$\mathbf{A}\mathbf{v}_j = q_j\delta\mathbf{x}_{j+1} + \mathbf{V}_{j+1}\mathbf{p}_j.$$

Proof. We prove the statement of the theorem by induction.

Initialization Since by (3.8) and (3.2) $\mathbf{A}\delta\mathbf{x}_0 = \delta\mathbf{x}_0 - \delta\mathbf{x}_1$ we have

$$\mathbf{A}\mathbf{v}_0 = \mathbf{A} \frac{\delta\mathbf{x}_0}{\|\delta\mathbf{x}_0\|} = \frac{1}{\|\delta\mathbf{x}_0\|} (\delta\mathbf{x}_0 - \delta\mathbf{x}_1) = -\frac{1}{\|\delta\mathbf{x}_0\|} \delta\mathbf{x}_1 + \mathbf{v}_0.$$

Induction Step The Arnoldi process guarantees

$$\mathbf{V}_{j+1}\mathbf{h}_j = \mathbf{A}\mathbf{v}_{j-1} = q_{j-1}\delta\mathbf{x}_j + \mathbf{V}_j\mathbf{p}_{j-1}.$$

The second equality holds by the induction hypothesis. Reordering the vectors and matrices in this equation we get

$$\mathbf{h}_{j+1,j}\mathbf{v}_j = q_{j-1}\delta\mathbf{x}_j - [\mathbf{V}_j|0] \begin{pmatrix} \mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \end{pmatrix}. \quad (4.3)$$

When multiplying the last equation by \mathbf{A} , we use again $\mathbf{A}\delta\mathbf{x}_j = \delta\mathbf{x}_j - \delta\mathbf{x}_{j+1}$. We obtain

$$\begin{aligned} \mathbf{h}_{j+1,j}\mathbf{A}\mathbf{v}_j &= \\ q_{j-1}\delta\mathbf{x}_j - q_{j-1}\delta\mathbf{x}_{j+1} - \mathbf{A}[\mathbf{V}_j|0] &\begin{pmatrix} \mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \end{pmatrix}. \end{aligned}$$

Solving (4.3) for $q_{j-1}\delta\mathbf{x}_j$ and using this expression as well as the Arnoldi equation $\mathbf{A}\mathbf{V}_j = \mathbf{V}_{j+1}\tilde{\mathbf{H}}_j$ in the last equation, we finally obtain

$$\begin{aligned} \mathbf{h}_{j+1,j}\mathbf{A}\mathbf{v}_j &= \\ -q_{j-1}\delta\mathbf{x}_{j+1} + \mathbf{V}_{j+1} (1 - [\tilde{\mathbf{H}}_j|0]) &\begin{pmatrix} \mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \end{pmatrix}. \end{aligned}$$

Replacing the given expressions for q_j and \mathbf{p}_j in this equation completes the proof. \square

The recursive scheme given in Lemma 4.2 can be used to compute the matrix vector multiplication $\mathbf{A}v_j$. With the result of this operation an Arnoldi step can be performed to compute v_{j+1} and \mathbf{H}_{j+1} . These are the only objects required to compute the next matrix vector multiplication. Thus, by Lemma 4.2 the matrix vector multiplication has been successfully replaced by a recursive scheme.

However, the scheme described in this lemma has a numerical drawback, which we like to illustrate by the following *linear* example.

Example 4.3 We apply the recursive scheme from Lemma 4.2 to the linearly generated sequence

$$x_0 := \begin{bmatrix} 2 \\ 3 \\ 0 \\ -3 \end{bmatrix}, x_{i+1} := \frac{1}{36} \begin{bmatrix} 0 & 6 & -9 & -3 \\ 4 & 0 & 0 & -4 \\ -9 & -3 & 0 & 3 \\ 16 & -4 & 12 & 0 \end{bmatrix} x_i + \begin{bmatrix} -1 \\ 2 \\ -1 \\ -1 \end{bmatrix}.$$

Table 4.1 shows the values of q_j and p_j obtained by the recursive scheme.

Table 4.1 Values of q_j and p_j for the given example.

j	q_j	p_j
0	-0.2516	$[1]^T$
1	0.8683	$[-0.4981, 0.6200]^T$
2	-7.3578	$[-0.8602, -0.5972, 0.7557]^T$
3	296.2253	$[-11.4420, -8.4234, -3.1968, 0.8004]^T$

We see that the absolute values of q_j grow exponentially. They change the sign in every step.

The observed exponential growth of the values of q_j is not an accident: The better the preconditioner works, the faster the norm of the difference vectors decreases. The norm of the difference vectors affects $h_{j+1,j}$, and the values of q_j reflect the product of these norms with

changing signs. Therefore in the computation of $\mathbf{A}\mathbf{v}_j$ the product $q_j\delta\mathbf{x}_{j+1}$ is unbalanced: If q_j becomes large, one can be sure that the norm of $\delta\mathbf{x}_{j+1}$ is small.

To solve this problem, we use the fact that the difference vectors $\delta\mathbf{x}_0, \dots, \delta\mathbf{x}_{j-1}$ span the Krylov space $\mathcal{K}_j(\mathbf{A}, \delta\mathbf{x}_0)$ (see Theorem 3.6). The following lemma shows how to construct the orthonormal basis of the Arnoldi process.

Lemma 4.4 *Given all conditions of Lemma 4.2.*

For $i = 0 \dots j$ the Gram-Schmidt process applied to $(-1)^i\delta\mathbf{x}_i$ yields the same orthonormal vectors \mathbf{v}_i as the Arnoldi process applied to $\mathbf{A}^i\delta\mathbf{x}_0$.

Proof. The statement is proved by induction.

Initialization The initialization is trivial, since $\mathbf{v}_0 = \delta\mathbf{x}_0/\|\delta\mathbf{x}_0\|$.

Induction Step The induction hypothesis is

$$[\delta\mathbf{x}_0, -\delta\mathbf{x}_1, \dots, \pm\delta\mathbf{x}_{j-1}] = \mathbf{V}_j\mathbf{K}_j.$$

Let \mathbf{w}_j denote the normalized vector that arises by orthogonalizing $\delta\mathbf{x}_j$ against \mathbf{V}_j . By the induction hypothesis we have

$$\begin{aligned} k_{j+1,j+1}\mathbf{w}_j &= (-1)^j\delta\mathbf{x}_j - \sum_{i=0}^{j-1} k_{i+1,j}\mathbf{v}_i \\ &= \left(\mathbf{I} - \mathbf{V}_j\mathbf{V}_j^H\right)(-1)^j\delta\mathbf{x}_j. \end{aligned} \quad (4.4)$$

Equation (4.3) implies

$$\delta\mathbf{x}_j = (-1)^j \frac{1}{|q_{j-1}|} \left(\mathbf{h}_{j+1,j}\mathbf{v}_j + [\mathbf{V}_j|0] \left(\mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \right) \right). \quad (4.5)$$

Replacing this expression for δx_j in (4.4) we get

$$k_{j+1,j+1} \mathbf{w}_j = \frac{h_{j+1,j}}{|q_{j-1}|} \mathbf{v}_j.$$

Both vectors \mathbf{w}_j and \mathbf{v}_j have the same length 1 and the same sign. Therefore they must be the same. Moreover we obtain for free a direct connection between $k_{j+1,j+1}$ and q_j ,

$$k_{j+1,j+1} = \frac{h_{j+1,j}}{|q_{j-1}|} = \frac{1}{|q_j|}. \tag{4.6}$$

□

The last lemma shows that the orthonormal basis of the Gram-Schmidt process applied to the signed difference vectors $(-1)^j \delta x_j$ is the same as the orthonormal basis generated by an Arnoldi process constructing $\mathcal{K}_j(\mathbf{A}, \delta \mathbf{x})$. However, the extended Hessenberg matrix $\tilde{\mathbf{H}}_j$ and the upper triangular matrix \mathbf{K}_j are different.

By the help of the last two lemmas the following theorem solves this problem. It provides a recursive scheme for obtaining the next column of the extended upper Hessenberg matrix $\tilde{\mathbf{H}}_j$ using the last two columns of the upper triangular matrix obtained by the Gram-Schmidt process and the actual extended upper Hessenberg matrix $\tilde{\mathbf{H}}_{j-1}$.

Theorem 4.5 Given $[\delta x_0, -\delta x_1, \dots, (-1)^{j-1} \delta x_{j-1}] = \mathbf{V}_j \mathbf{K}_j$ — a QR-decomposition determined by a Gram-Schmidt process, let $\mathbf{A} \mathbf{V}_j = \mathbf{V}_{j+1} \tilde{\mathbf{H}}_j$ be the result of an Arnoldi process on \mathbf{A} and δx_0 .

Then the column vectors of $\tilde{\mathbf{H}}_j$ can be obtained by the following recursive scheme

$$\begin{aligned} \tilde{\mathbf{H}}_0 &= [\] \in \mathbb{R}^{1 \times 0}, \\ \mathbf{h}_j &= \frac{1}{k_{j,j}} \left(\mathbf{k}_{j+1} + \begin{bmatrix} (1 - [\tilde{\mathbf{H}}_{j-1}|0]) \mathbf{k}_j \\ 0 \end{bmatrix} \right) \quad \forall j \geq 1. \end{aligned}$$

Proof. The initial value for $\tilde{\mathbf{H}}_0$ is trivial.

The recursive part of the given scheme remains to be proven. From (4.5) we know

$$(-1)^j \delta \mathbf{x}_j = \mathbf{V}_{j+1} \left(\mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \right) / |q_{j-1}|.$$

Thus we observe that the last column vector of $\tilde{\mathbf{K}}_j$ takes the following form

$$\mathbf{k}_{j+1} = \left(\mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} \right) / |q_{j-1}|. \quad (4.7)$$

Using (4.6) we can rewrite that as

$$\mathbf{k}_{j+1} = k_{j,j} \mathbf{h}_j - \begin{bmatrix} \mathbf{p}_{j-1} \\ 0 \end{bmatrix} / |q_{j-1}|.$$

Our aim is to eliminate q_{j-1} and \mathbf{p}_{j-1} . Using their definitions (4.2) we get

$$\mathbf{k}_{j+1} = k_{j,j} \mathbf{h}_j - \left[(1 - [\tilde{\mathbf{H}}_{j-1}|0]) \begin{pmatrix} \mathbf{h}_{j-1} - [\mathbf{p}_{j-1}^T, 0] \\ 0 \end{pmatrix} / |q_{j-2}| \right].$$

The recursive sensation happens by the help of (4.7): the expression $(\mathbf{h}_{j-1} - [\mathbf{p}_{j-1}^T, 0]) / |q_{j-2}|$ can be replaced by \mathbf{k}_j . Thus we get

$$\mathbf{k}_{j+1} = k_{j,j} \mathbf{h}_j - \begin{bmatrix} (1 - [\tilde{\mathbf{H}}_{j-1}|0]) \mathbf{k}_j \\ 0 \end{bmatrix}.$$

Solving this equation for \mathbf{h}_j yields the proposed result. \square

Theorem 4.5 implies a procedure for obtaining the Arnoldi basis \mathbf{V}_j and the extended Hessenberg matrix $\tilde{\mathbf{H}}_j$, if the signed difference vectors $(-1)^j \delta \mathbf{x}_j$ are orthogonalized against each other. This results in the *nonlinear Arnoldi process*, which is implemented in Algorithm 4.1.

Algorithm 4.1 Step j of the nonlinear Arnoldi process.

function nlArnoldi($j, \mathbf{v}_j, \mathbf{V}_j, \mathbf{k}_j, \tilde{\mathbf{H}}_{j-1}$)
:[vector, vector, vector];

{Modified Gram–Schmidt orthogonalization}

for $i := 0$ **to** $j - 1$ **do**
 $k_{i+1, j+1} := \mathbf{v}_j^H \mathbf{v}_i$;
 $\mathbf{v}_j := \mathbf{v}_j - k_{i+1, j+1} \mathbf{v}_i$;
end;

$k_{j+1, j+1} := \|\mathbf{v}_j\|$;
 $\mathbf{v}_j := \mathbf{v}_j / k_{j+1, j+1}$;

{Determining the Hessenberg matrix}

$\mathbf{h}_j := \left(\mathbf{k}_{j+1} + \begin{bmatrix} (1 - [\tilde{\mathbf{H}}_{j-1} \mathbf{0}]) \mathbf{k}_j \\ 0 \end{bmatrix} \right) / k_{j, j}$;

nlArnoldi := $[\mathbf{k}_j, \mathbf{h}_j, \mathbf{v}_j]$;
end nlArnoldi;

The algorithm performs one step of the Arnoldi process from $j - 1$ to j . Its arguments are the step number j , the signed difference vector $\mathbf{v}_j := (-1)^j \delta \mathbf{x}_j$, the already obtained Arnoldi basis \mathbf{V}_j , the last column vector of the actual upper triangular matrix of the Gram–Schmidt process \mathbf{k}_j and the already obtained extended Hessenberg matrix $\tilde{\mathbf{H}}_{j-1}$ from the last step. It returns the next column of both the upper triangular matrix and the extended Hessenberg matrix and the new vector of the Arnoldi basis.

The nonlinear Arnoldi process can be used to reformulate all Arnoldi based linear Krylov space methods as nonlinear Krylov space methods. Thus resulting in a huge family of nonlinear Krylov space methods, for which a vector extrapolation counterpart might not even exist. In the following we present the nonlinear versions of the two Arnoldi based algorithms from Chapter 2.

4.2.1 Full Orthogonalization Method

Algorithm 4.2 Nonlinear FOM with constant number (n) of steps.

```

function nlFOM( $g(\cdot)$ ,  $s_0$ ,  $\varepsilon$ ,  $n$ ): vector;
   $\mathbf{b} := g(0)$ ;
  repeat
     $\mathbf{x}_0 := s_0$ ;  $\mathbf{x}_1 := g(\mathbf{x}_0)$ ;
     $\delta\mathbf{x}_0 := \mathbf{x}_1 - \mathbf{x}_0$ ;  $\rho_0 := \|\delta\mathbf{x}_0\|$ ;
     $k_{1,1} := \rho_0$ ;  $\mathbf{v}_0 := \delta\mathbf{x}_0/\rho_0$ ;   {Gram–Schmidt setup}
     $\tilde{\mathbf{H}}_0 := []$ ;                         {Arnoldi setup}
    for  $j := 1$  to  $n$  do
       $\mathbf{x}_{j+1} := g(\mathbf{x}_j)$ ;  $\delta\mathbf{x}_j := \mathbf{x}_{j+1} - \mathbf{x}_j$ ;

      [ $k_j$ ,  $\mathbf{h}_j$ ,  $\mathbf{v}_j$ ] := nlArnoldi( $(-1)^j \delta\mathbf{x}_j$ ,  $\mathbf{V}_j$ ,  $\mathbf{k}_{j-1}$ ,  $\tilde{\mathbf{H}}_{j-1}$ );

       $\mathbf{z}_j := \mathbf{H}_j^{-1} \mathbf{e}_1 \rho_0$ ;           {common FOM}
       $\mathbf{s}_j := s_0 + \mathbf{V}_j \mathbf{z}_j$ ;
       $\rho_j := \mathbf{h}_{j+1,j} |e_j^H \mathbf{z}_j|$ 
    end;
     $s_0 := s_n$                                {restart}
  until  $\rho_n \leq \varepsilon \|\mathbf{b}\|$ ;
  nlFOM :=  $s_n$ 
end nlFOM;

```

The nonlinear Arnoldi process is everything needed to transform linear FOM into nonlinear FOM. An algorithm combining linear FOM with the nonlinear Arnoldi process is given in Algorithm 4.2.

This algorithm takes as arguments the nonlinear preconditioner g , an initial guess of the solution s_0 and a desired tolerance ε . For the ease of understanding we formulated the algorithm such that it performs a constant number of steps until a restart takes place. This is the last argument n , the algorithm takes. However, a fancier implementation will use the framework of inexact Newton methods for error control described in Chap-

ter 1 to have a variable number of steps (less steps far from the solution, more steps near the solution).

4.2.2 General Minimum Residual Method

Replacing the linear Arnoldi process by the nonlinear Arnoldi process in GMRES results in the nonlinear version of GMRES shown in Algorithm 4.3.

This algorithm takes the same arguments as nonlinear FOM. Instead of the constant number of steps used in this algorithm, one would also like to use the error control framework for inexact Newton methods described in Chapter 1. For the ease of understanding this framework has not been implemented in Algorithm 4.3.

4.3 Lanczos Type Methods

In analogy to the nonlinear Arnoldi methods, the aim of nonlinear Lanczos type methods is to replace the matrix vector multiplications $\mathbf{A}\mathbf{v}_j$ and $\mathbf{A}^H\mathbf{w}_j$ by using information from the topological epsilon algorithm or its relatives. There are strong indications that such a replacement can be found: *First* Theorem 3.27 shows the mathematical equivalence of the topological epsilon algorithm and the biconjugate gradient method, *secondly* Brezinsky [7, p. 236] mentions how the direction vectors of the conjugate gradient method can be constructed from the topological epsilon algorithm if it is applied to a linear system with a symmetric positive definite matrix.

The benefit of such a replacement would be a nonlinear Lanczos biorthogonalization resulting in a nonlinear BiCG algorithm and a nonlinear QMR algorithm. As far as we know there is no vector extrapolation counterpart known so far for the latter algorithm.

However, up to now we have not been able to develop

Algorithm 4.3 Nonlinear GMRES with constant number n of steps before restart.

function nlGMRES($g(\cdot)$, s_0 , ε , n): **vector**;
 $\mathbf{b} := g(0)$;
repeat
 $\mathbf{x}_0 := s_0$; $\mathbf{x}_1 := g(\mathbf{x}_0)$;
 $\delta\mathbf{x}_0 := \mathbf{x}_1 - \mathbf{x}_0$; $\rho_0 := \|\delta\mathbf{x}_0\|$;
 $\mathbf{k}_{1,1} := \rho_0$; $\mathbf{v}_0 := \delta\mathbf{x}_0/\rho_0$;
 $\bar{\mathbf{H}}_0 := []$;
for $j := 1$ **to** n **do**
 $\mathbf{x}_{j+1} := g(\mathbf{x}_j)$; $\delta\mathbf{x}_j := \mathbf{x}_{j+1} - \mathbf{x}_j$;
 $[\mathbf{k}_j, \mathbf{h}_j, \mathbf{v}_j] := \text{nlArnoldi}(\delta\mathbf{x}_j, \mathbf{V}_j, \mathbf{k}_{j-1}, \bar{\mathbf{H}}_{j-1})$;
{Update previous Givens Rotations}
 $r_{1,j} := h_{1,j}$;
for $i := 1$ **to** $j-1$,

$$\begin{bmatrix} r_{i,j} \\ r_{i+1,j} \end{bmatrix} := \begin{bmatrix} \cos_i & \sin_i \\ -\sin_i & \cos_i \end{bmatrix} \begin{bmatrix} r_{i,j} \\ h_{i+1,j} \end{bmatrix}$$

end;
{Actual Givens Rotation}
 $\tan := h_{j+1,j}/r_{j,j}$;
 $\cos_j := 1/\sqrt{1 + \tan^2}$; $\sin_j := \cos_j \tan$;
 $r_{j,j} := r_{j,j} \cos_j + h_{j+1,j} \sin_j$;
{Right hand side and component vector}
 $\rho_j := -\rho_{j-1} \sin_j$; $g_j^{(j)} := \rho_{j-1} \cos_j$;
 $\mathbf{z}_j := \mathbf{R}_j^{-1} g^{(j)}$;
end;
 $s_0 := s_n$
until $\rho_n \leq \varepsilon \|\mathbf{b}\|$;
nlGMRES := s_n
end nlGMRES;

a recursive scheme that can replace the matrix vector multiplications mentioned above. Therefore nonlinear Lanczos type methods remain an interesting field to be investigated.

4.4 Conclusion

Let us begin our conclusion with a numerical example.

Example 4.6 We illustrate our nonlinear Krylov space methods by the Chandrasekhar H-equation introduced at the end of Chapter 1. To show that nIFOM is numerically more stable than MPE we use a larger problem with $n = 400$. We compare the Newton method, the chord method, Newton-GMRES, MPE, RRE, nIFOM and nIGMRES. All algorithms had to achieve a required relative residual norm of 10^{-10} . For Newton-GMRES, MPE, RRE, nIFOM and nIGMRES the error control mechanism of inexact Newton methods is used as described in Chapter 1. MPE, RRE, nIFOM and nIGMRES use the chord method as the nonlinear preconditioner.

Both algorithms nIGMRES as well as nIFOM use the update procedure with Givens rotations discussed in Chapter 2 for GMRES to decompose the upper Hessenberg matrix into a sequence of rotation matrices and an upper triangular matrix.

Figure 4.2 shows the error curves of the particular Algorithms. We see that RRE and nIGMRES and MPE and nIFOM show a comparable behavior.

Table 4.2 shows the number of (outer) iterations, floating point operations and time measured on a Sun SPARC 336 MHz with 3 GB main memory for the compared algorithms applied to the given problem. RRE and nIGMRES have comparable results, nIFOM is slightly better than MPE although it takes more outer iterations.

The Krylov space paradigm is a field very well researched by numerical analysts. Therefore our hope is that transferring these vector extrapolation algorithms to the Krylov space paradigm gives rise for further ideas for numerical improvement of these algorithms.

CONCLUSION

The new algorithms are able to solve nonlinear problems without the knowledge of the Jacobian. However, they rely on the application of a nonlinear preconditioner.

Further work should include research on Lanczos type methods and epsilon algorithms. In this field the trans-

Figure 4.2 The Chandrasekhar H-equation with $n = 400$ and $c = 0.9999$.

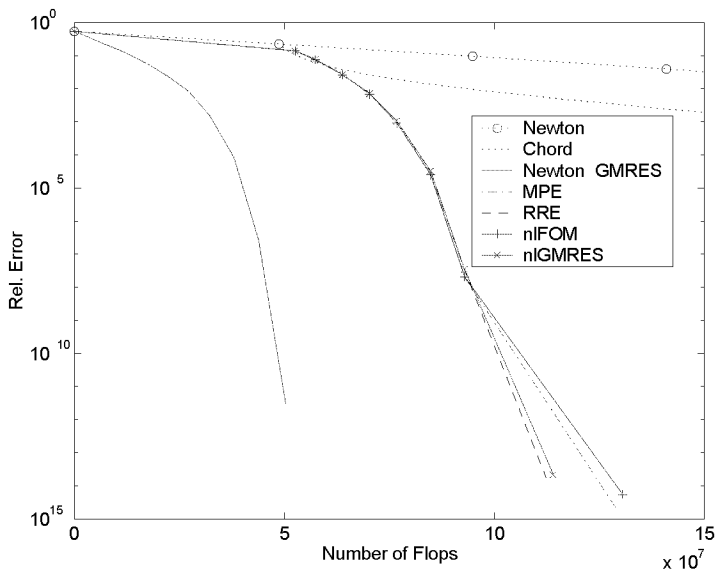


Table 4.2 Flops and execution times of several methods for solving the Chandrasekhar H-equation. Flops were counted with the `flops` command of Matlab 5.3. Times were measured in Matlab on a Sun SPARC 336 MHz with 3 GB main memory.

Method	(Outer) Iterations	Mflops	Time(s)
Newton-GMRES	9	50.3	53.63
Newton	9	417.6	117.85
RRE	8	112.3	123.24
nIGMRES	8	114.0	125.00
nIFOM	11	118.7	132.35
MPE	8	129.2	148.63
Chord	437	750.5	1323.41


fer to the Krylov space paradigm promises completely new algorithms such as nonlinear QMR, for which no vector extrapolation counterpart is known as of today.

This could also have an impact on new vector extrapolation methods. Moreover, we think that it is very likely, that the bad behavior of TEA1 algorithm shown at the end of Chapter 3 can be cured by such a transfer with respect to memory requirements, amount of work and numerical stability.

Chapter 5

Application

5.1 Introduction

 In this chapter we apply the new algorithms to a real world problem. We describe the simulation of an ash melting oven used in the Deglor project of the Asea Brown Boveri Company (ABB).

The simulation consists of a heat transfer problem: The distribution of the temperature and the heat fluxes at the surface of the oven are determined in the stationary case. The physical model of the oven and its discretization lead to a nonlinear system of equations, which can be solved with the algorithms that have been described in this thesis.

In our diploma thesis [21] and in a conference paper [22] we have dealt with the parallelization of the simulation. The parallel simulation was used to examine the behavior of a very large Deglor oven that was build by the ABB in Japan. The parallelized code used nonlinear Jacobi, Gauss-Seidel and SOR as a solver.

In this chapter we explain the purpose and the geometry of the oven. Next we describe the problem, the physical model and its discretization. Then we apply our new algorithms to the simulation of a medium sized oven that was built in the research center of the ABB company in Dättwil/Switzerland for test purposes. We compare the results to competing existing algorithms. Finally we conclude on the results made in this thesis.

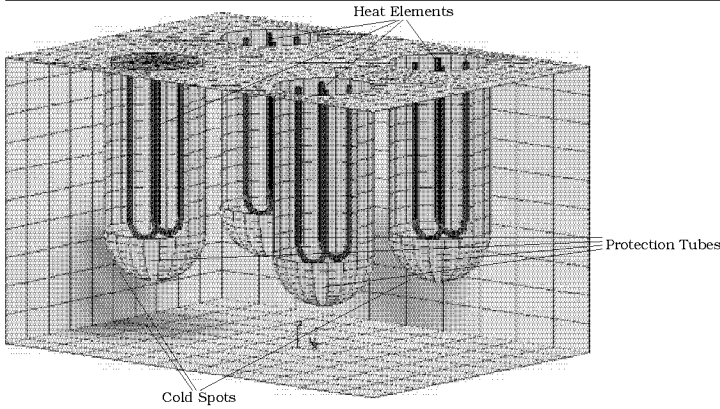
5.2 ABB's Deglor Oven

PURPOSE When waste is incinerated, an ash is produced in the filters containing toxic heavy metals at a level of approximately 10 percent. Melting the ash allows for the evaporation of the heavy metals. The heavy metals in gaseous form are conducted into another filter, where they condensate. The condensed heavy metals are hazardous waste and have to be disposed carefully.

About 90 percent of the ash do not evaporate. If the melt is cooled down, it becomes glass, which can be recycled for further industrial processes.

SHAPE Figure 5.1 shows a picture of the upper part of a Deglor ash melting furnace. The floor represents the upper surface of the melt. Heat elements powered by electricity are hanging from the ceiling. They are surrounded by protection tubes. Three cold spots are on the walls and on the floor. These are locations where the temperature is considered to be constantly *cold* (i.e. 600 K) throughout the simulation. The first cold spot on the left side models a hole in the left wall, where new ash is pushed into the oven. The second cold spot is the place on the upper surface of the melt below that hole, where the new ash is placed when its pushed into the oven through the hole. The third cold place is at the back wall. It models a hole, where the evaporated heavy metals may leave the furnace.

Figure 5.1 Picture of a Deglor ash melting furnace consisting of 3 cold spots and 4 protection tubes each of which contains 2 heat elements.



The modeling of the oven's geometry and the graphical output of the results were done by *ASTRID/B2000*, a modular finite element analysis system developed at EPFL Lausanne[4]. To store the data this system uses a data management system called *MEMCOM*. Both programming environments are commercial products and can be obtained from SMR Engineering & Development, a consulting company specialized in scientific computing and engineering computer simulation.

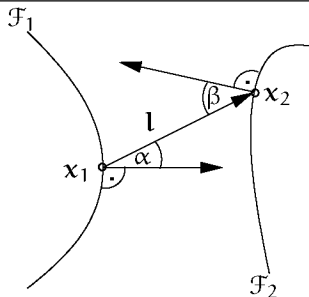
TOOLS

5.2.1 Physical Model

Definition 5.1 Given two points x_1 and x_2 on given surfaces, a view factor describes the fraction of the intensity of radiation between these two points x_1 and x_2 . Since the points are located on surfaces, we can define angles α , β between the distance vector $\mathbf{l} := x_2 - x_1$ and the normal vectors of the surface at the given points (see Figure 5.2).

SURFACE
VIEW
FACTORS

Figure 5.2 The angles α and β and the length of the distance vector l are quantities for computing the view factor $v(x_1, x_2)$ between the two points x_1 and x_2 on the surfaces \mathcal{F}_1 and \mathcal{F}_2 .



The view factor $v(x_1, x_2)$ depends on the distance $l := \|x_2 - x_1\|$ of the two points and their angles, i.e.

$$v(x_1, x_2) := \frac{\cos \alpha \cos \beta}{l^2 \pi}.$$

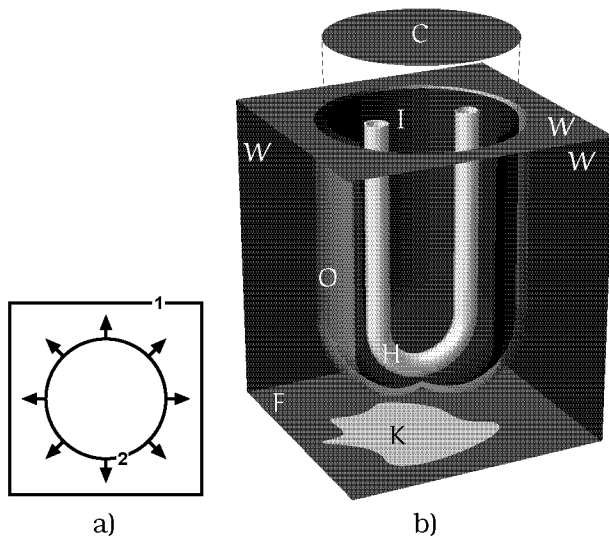
If there is an obstacle between the two points, the view factor becomes zero. We also define the view factor to be zero if x_1 equals x_2 .

The walls of the oven together with the outside walls of the protection tubes form a closed space (*exterior region*). The same is true for the protection tubes together with the heat elements (*interior regions*). Both types of regions can be viewed to have a cold outside wall (1) and a hot emission part (2) (see Figure 5.3). Inside these regions *heat radiation* takes place. The interior regions are connected to the exterior region by *heat conduction*.

HEAT
RADIATION

Let $p(x)$ denote the heat flux at a point x . Let $T(x)$ be the temperature and $\varepsilon(x)$ the heat radiation coefficient at this point. The ratio of power emitted by the black body for a range of wavelengths to the whole spectrum at temperature $T(x)$ shall be represented by $\gamma(x)$, σ denotes the Boltzmann constant. To determine the distri-

Figure 5.3 a) Every region can be divided into a cold outside wall (1) and a hot temperature emission part (2). b) The diagram denotes our notation for these parts of the oven.



tribution of temperatures and heat fluxes on the surface of the Deglor oven the following equation has to be solved for T p for all points x of the surface \mathcal{F} of a closed region of the oven (see [9]):

$$\begin{aligned} & \frac{p(x)}{\varepsilon(x)} - \int_{\mathcal{F}} v(x, \mathbf{y}) \frac{1 - \varepsilon(\mathbf{y})}{\varepsilon(\mathbf{y})} p(\mathbf{y}) d\mathcal{F}(\mathbf{y}) \\ &= \gamma(x) \sigma T(x)^4 - \int_{\mathcal{F}} v(x, \mathbf{y}) \gamma(\mathbf{y}) \sigma T(\mathbf{y})^4 d\mathcal{F}(\mathbf{y}). \quad (5.1) \end{aligned}$$

In this equation the heat flux is dependent on the 4th power of the temperature.

On the outside walls and the tube walls heat conduction happens. Let $T_O(x)$ denote the temperature on the

outside of the wall and $T_I(x)$ the temperature on the inside at a point x . The heat conduction is modeled by a linear dependence of temperature and heat flux, i.e.

$$p(x) = \alpha(x)(T_O(x) - T_I(x)). \quad (5.2)$$

In this equation $\alpha(x)$ denotes the heat conduction coefficient at the point x .

5.2.2 Discretization

MODEL OF THE FURNACE

The aim of the simulation is to determine physical values on the surfaces of the oven. Therefore the model of the furnace has only two dimensions.

An ash melting furnace consists of the following geometric solids: planes (the floor, the ceiling, the walls, covers of the protection tubes), cylinders and semi spheres (protection tubes), cylinders and half toruses (heat elements). The ceiling of the oven is a plane with holes. The ceiling of the oven is a plane with holes. Each protection tube causes one. Therefore the ceiling is partitioned into subplanes. Four subplanes form a rectangular area containing one hole.

SUBDOMAINS

The solids of which the oven is composed, are called *subdomains*. Every furnace has five large plane subdomains, four for the walls and one for the floor. The ceiling is formed by small planar subdomains, the number of which is four times the number of protection tubes. Every protection tube consists of a cylindrical and a semi spherical subdomain. Every heat element consists of two cylindrical subdomains and a half torus shaped ones. The oven in Figure 5.1 consists of 65 subdomains.

THE MESH AND SURFACE ELEMENTS

Every subdomain has only two dimensions. Therefore we can specify a two dimensional grid for each subdomain. Cylinders and spheres are approximated by prisms. This way every subdomain is partitioned into a finite number of quadrilateral *surface elements*.

Definition 5.2 The view factor $v^{\mathcal{F}}(\mathcal{F}_1, \mathcal{F}_2)$ between two surface elements \mathcal{F}_1 and \mathcal{F}_2 is the average over all view factors between these surface elements, i.e.

$$\begin{aligned} v^{\mathcal{F}}(\mathcal{F}_1, \mathcal{F}_2) &:= \frac{1}{\|\mathcal{F}_1\| \|\mathcal{F}_2\|} \int_{\mathcal{F}_1} \int_{\mathcal{F}_2} v(\mathbf{x}, \mathbf{y}) d\mathcal{F}(\mathbf{y}) d\mathcal{F}(\mathbf{x}) \\ &= \frac{1}{\pi \|\mathcal{F}_1\| \|\mathcal{F}_2\|} \int_{\mathcal{F}_1} \int_{\mathcal{F}_2} \frac{\cos \alpha \cos \beta}{l^2} d\mathcal{F}(\mathbf{y}) d\mathcal{F}(\mathbf{x}). \end{aligned}$$

To simplify the linear system of equations which we are developing for the problem, we will use the following definition from [9, p. 2].

Definition 5.3 The standard modification $w(\mathcal{F}_1, \mathcal{F}_2)$ of the view factor between two surface elements \mathcal{F}_1 and \mathcal{F}_2 is defined by

$$\begin{aligned} w(\mathcal{F}_1, \mathcal{F}_2) &:= v^{\mathcal{F}}(\mathcal{F}_1, \mathcal{F}_2) \|\mathcal{F}_2\| \\ &= \frac{1}{\pi \|\mathcal{F}_1\|} \int_{\mathcal{F}_1} \int_{\mathcal{F}_2} \frac{\cos \alpha \cos \beta}{l^2} d\mathcal{F}(\mathbf{y}) d\mathcal{F}(\mathbf{x}). \end{aligned} \quad (5.3)$$

Note that view factors of surface elements are symmetric, i.e. $v^{\mathcal{F}}(\mathcal{F}_1, \mathcal{F}_2) = v^{\mathcal{F}}(\mathcal{F}_2, \mathcal{F}_1)$ whereas the standard modification is not. Instead the standard modification satisfies $w(\mathcal{F}_1, \mathcal{F}_2) = \|\mathcal{F}_2\| / \|\mathcal{F}_1\| w(\mathcal{F}_2, \mathcal{F}_1)$. However, the integral in (5.3) is still symmetric.

We put all the view factors of the surface elements into a matrix \mathbf{V} and their standard modifications into a matrix \mathbf{W} with $v_{i,j} = v^{\mathcal{F}}(\mathcal{F}_1, \mathcal{F}_2)$ and $w_{i,j} = w(\mathcal{F}_1, \mathcal{F}_2)$.

The integral in (5.3) is approximated by partitioning the source and the target surface elements \mathcal{F}_1 and \mathcal{F}_2 into k subareas $\mathcal{F}_{1,1\dots k}$ and $\mathcal{F}_{2,1\dots k}$. Of these subareas the angles $\alpha_{1\dots k,1\dots k}$ and $\beta_{1\dots k,1\dots k}$ and the distance $l_{1\dots k,1\dots k}$ of the points in the center are computed. These values are considered to be constant on the whole subarea.

The integral is approximated by the symmetric sum

$$\int_{\mathcal{F}_1} \int_{\mathcal{F}_2} \frac{\cos \alpha \cos \beta}{l^2} d\mathcal{F}(\mathbf{y}) d\mathcal{F}(\mathbf{x}) = \sum_{i=1}^k \sum_{j=1}^k \frac{\cos \alpha_{i,j} \cos \beta_{i,j}}{l_{i,j}^2} \|\mathcal{F}_{2,j}\| \|\mathcal{F}_{1,i}\|.$$

A LINEAR SYSTEM

Let us define an order for the surface elements and assign an index to every surface element. We consider the physical quantities of (5.1) to be constant on a surface element. Therefore let the physical quantities be indexed according to the surface elements. Let n denote the total number of surface elements. With these assumptions the discretization of (5.1) becomes

$$\frac{p_i}{\varepsilon_i} - \sum_{j=1}^n v_{i,j} \frac{1 - \varepsilon_j}{\varepsilon_j} p_j \|\mathcal{F}_j\| = \gamma_i \sigma T_i^4 - \sum_{j=1}^n v_{i,j} \gamma_j \sigma T_j^4 \|\mathcal{F}_j\| \quad (i = 1 \dots n).$$

This set of equations defines a linear system in the values p_i and $\gamma_i \sigma T_i^4$. Let us put heat fluxes into a vector \mathbf{p} , temperatures into a vector \mathbf{t} and the values $\gamma_i \sigma T_i^4$, ($i = 1 \dots n$) into a vector \mathbf{h} .

As we have seen in Figure 5.3, every region can be divided into a cold wall and a hot temperature emission part. On the surface of the cold wall we suppose that the temperature is known, on the surface of the hot part we suppose that a distribution of the heat fluxes is known.

Therefore let the surface elements be ordered such that those belonging to the cold outside wall hold indexes in the range $1 \dots m - 1$, whereas those belonging to the hot temperature emission part hold indexes in the range $m \dots n$.

According to the partition of the surface elements the matrix containing the standard modification of the

view factors can be partitioned in to four submatrices $W_{1,1} \in \mathbb{R}^{(m-1) \times (m-1)}$, $W_{1,2} \in \mathbb{R}^{(m-1) \times (n-m)}$, $W_{2,1} \in \mathbb{R}^{(n-m) \times (m-1)}$ and $W_{2,2} \in \mathbb{R}^{(n-m) \times (n-m)}$ such that W becomes

$$W = \left[\begin{array}{c|c} W_{1,1} & W_{1,2} \\ \hline W_{2,1} & W_{2,2} \end{array} \right].$$

Likewise let all vectors be partitioned into a subvectors the first of which belongs to \mathbb{R}^{m-1} and the second to \mathbb{R}^{n-m} .

Let us also define two diagonal matrices E and H the diagonal elements of which are defined by

$$e_{i,i} = \varepsilon_i \quad \text{and} \quad h_{i,i} = \frac{1 - \varepsilon_i}{\varepsilon_i}.$$

These matrices are partitioned in the same way as W .

With these definitions and the assumption that the emission coefficients ε_i do not depend on the temperature T_i the discretized linear relation between p and T^4 for a whole region can be rewritten in matrix vector notation

$$\underbrace{\left[\begin{array}{c|c} 1 - W_{1,1} & -W_{1,2} \\ \hline -W_{2,1} & 1 - W_{2,2} \end{array} \right]}_{=: A} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \tag{5.4}$$

$$\underbrace{\left[\begin{array}{c|c} E_{1,1}^{-1} - W_{1,1}H_{1,1} & -W_{1,2}H_{2,2} \\ \hline -W_{2,1}H_{1,1} & E_{2,2}^{-1} - W_{2,2}H_{2,2} \end{array} \right]}_{=: B} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}.$$

It is true that the heat fluxes on the surface of the hot temperature emission part is known. However, on the walls only the outside temperature is known. The heat flux on the walls is related to the outside temperature by (5.2), which is in discretized form

$$p_W = \alpha_W(t_O - t_W). \tag{5.5}$$

In this equation the vectors \mathbf{p}_W and \mathbf{t}_W hold the unknown heat fluxes and temperature on the inside of the wall, \mathbf{t}_O holds the known temperature on the outside of the wall. The constant α_W denotes the heat conduction coefficient at the wall. The equation describes a linear relation between temperature and the heat fluxes. The combination of (5.5) with (5.4) results in a nonlinear system in heat fluxes and temperature.

Performing the simulation means finding the temperatures and heatfluxes in the stationary case of the furnace. The engineers at ABB have considered two models of the oven: A simple model describing the exterior region only and a complex model describing exterior and interior regions.

5.2.3 Simple Model: Exterior Region Only

Let us partition the exterior region into *walls and ceiling (W)*, *floor (F)*, *cold places (K)* and *outer sides of the protection tubes (O)*, see Figure 5.3.

Let the number of protection tubes be t . According to this name scheme the matrices \mathbf{A} , \mathbf{B} and the vectors \mathbf{t} and \mathbf{p} can be partitioned as

$$\begin{aligned}\mathbf{A} &= \left[\mathbf{A}_W | \mathbf{A}_F | \mathbf{A}_K | \mathbf{A}_O^{(1)} | \dots | \mathbf{A}_O^{(t)} \right], \\ \mathbf{B} &= \left[\mathbf{B}_W | \mathbf{B}_F | \mathbf{B}_K | \mathbf{B}_O^{(1)} | \dots | \mathbf{B}_O^{(t)} \right], \\ \mathbf{t} &= \left[\mathbf{t}_W, \mathbf{t}_F, \mathbf{t}_K, \mathbf{t}_O^{(1)}, \dots, \mathbf{t}_O^{(t)} \right]^T, \\ \mathbf{p} &= \left[\mathbf{p}_W, \mathbf{p}_F, \mathbf{p}_K, \mathbf{p}_O^{(1)}, \dots, \mathbf{p}_O^{(t)} \right]^T.\end{aligned}$$

Let the vector containing the outside temperature be denoted by \mathbf{t}_O . Using (5.5), (5.4) the definition of \mathbf{h} and the

partition of the matrices and vectors the system to be solved for the outer region turns into

$$\gamma\sigma \left(\mathbf{A}_W \underline{\mathbf{t}}_W^4 + \mathbf{A}_F \underline{\mathbf{t}}_F^4 + \mathbf{A}_K \underline{\mathbf{t}}_K^4 + \sum_{i=1}^t \mathbf{A}_O^{(i)} \underline{\mathbf{t}}_O^{(i)4} \right) =$$

$$\alpha_W \mathbf{B}_W (\underline{\mathbf{t}}_O - \underline{\mathbf{t}}_W) + \mathbf{B}_F \underline{\mathbf{p}}_F + \mathbf{B}_K \underline{\mathbf{p}}_K + \sum_{i=1}^t \mathbf{B}_O^{(i)} \underline{\mathbf{p}}_O^{(i)}. \quad (5.6)$$

In this equation we supposed that γ_i is constant. We also introduced the notation of the power of a vector. It means that the power has to be applied to every component of that vector.

To get an impression of what is unknown and what is known, the known vectors have been underlined. The other values are unknown. Let us put them into the vector of unknowns $\mathbf{x} := [\underline{\mathbf{t}}_W, \underline{\mathbf{t}}_F, \underline{\mathbf{p}}_K, \underline{\mathbf{t}}_O^{(1)}, \dots, \underline{\mathbf{t}}_O^{(t)}]^T$.

Reordering (5.6) we define

$$\tilde{\mathbf{A}} := \gamma\sigma \left[\mathbf{A}_W | \mathbf{A}_F | \mathbf{0}_K | \mathbf{A}_O^{(1)} | \dots | \mathbf{A}_O^{(t)} \right],$$

$$\tilde{\mathbf{B}} := \left[\alpha_W \mathbf{B}_W | \mathbf{0}_F | - \mathbf{B}_K | \mathbf{0}_O^{(1)} | \dots | \mathbf{0}_O^{(t)} \right],$$

$$\mathbf{c} := \left[-\alpha_W \mathbf{B}_W | - \mathbf{B}_F | \gamma\sigma \mathbf{A}_K | - \mathbf{B}_O^{(1)} | \dots | - \mathbf{B}_O^{(t)} \right]$$

$$[\underline{\mathbf{t}}_O, \underline{\mathbf{p}}_F, \underline{\mathbf{t}}_K^4, \underline{\mathbf{p}}_O^{(1)}, \dots, \underline{\mathbf{p}}_O^{(t)}]^T.$$

With these definitions the nonlinear system (5.6) can be rewritten in a simple form as

$$\boldsymbol{\varphi}(\mathbf{x}) := \tilde{\mathbf{A}}\mathbf{x}^4 + \tilde{\mathbf{B}}\mathbf{x} + \mathbf{c} \stackrel{!}{=} \mathbf{0} \quad (5.7)$$

The operator $\boldsymbol{\varphi}$ holds a polynomial in every component. We are interested in finding a zero of $\boldsymbol{\varphi}$.

5.2.4 Complex Model: Exterior and Interior Regions

THE SYSTEM
IN THE
INTERIOR
REGION

An interior region consists of the *inner side of the tubes (I)*, a *cover (C)* and *heat elements (H)* — see Figure 5.3. Since all values we deal with concern the i^{th} interior region, we omit the (i) for the ease of reading.

In every interior region the linear system (5.4) holds. At the walls of the tubes the heat conduction equation (5.5) holds, which in the case of the interior regions becomes

$$\mathbf{p}_I = \alpha_T(\mathbf{t}_O - \mathbf{t}_I).$$

The constant α_T denotes the heat conduction coefficient on the tube walls. At the cover we have

$$\mathbf{p}_C = \alpha_W(\mathbf{t}_O - \mathbf{t}_C).$$

Similarly as before the nonlinear system inside the interior regions turns into

$$\begin{aligned} \gamma\sigma(\mathbf{A}_I \mathbf{t}_I^4 + \mathbf{A}_C \mathbf{t}_C^4 + \mathbf{A}_H \mathbf{t}_H^4) = \\ \alpha_T \mathbf{B}_I(\underline{\mathbf{t}_O} - \mathbf{t}_I) + \alpha_W \mathbf{B}_C(\mathbf{t}_O - \mathbf{t}_C) + \mathbf{B}_H \underline{\mathbf{p}_H}. \end{aligned}$$

Again known values have been underlined in this equation.

COMBINING
THE REGIONS

The nonlinear system (5.6) for the exterior region still applies. However, the values $\mathbf{p}_O^{(i)}$ are now longer known values. Instead the exterior region is now connected to the interior regions by the relation

$$\mathbf{p}_O^{(i)} = \alpha_T(\mathbf{t}_I^{(i)} - \mathbf{t}_O^{(i)}).$$

With this alteration (5.6) turns into

$$\gamma\sigma \left(\mathbf{A}_W \mathbf{t}_W^4 + \mathbf{A}_F \mathbf{t}_F^4 + \mathbf{A}_K \mathbf{t}_K^4 + \sum_{i=1}^t \mathbf{A}_O^{(i)} \mathbf{t}_O^{(i)4} \right) = \\ \alpha_W \mathbf{B}_W (\mathbf{t}_O - \mathbf{t}_W) + \mathbf{B}_F \mathbf{p}_F + \mathbf{B}_K \mathbf{p}_K + \alpha_T \sum_{i=1}^t \mathbf{B}_O^{(i)} (\mathbf{t}_I^{(i)} - \mathbf{t}_O^{(i)}).$$

Let us define the vector of unknown values

$\mathbf{x} :=$

$$\left[\mathbf{t}_I^{(1)}, \mathbf{t}_C^{(1)}, \mathbf{t}_H^{(1)}, \dots, \mathbf{t}_I^{(t)}, \mathbf{t}_C^{(t)}, \mathbf{t}_H^{(t)}, \mathbf{t}_W, \mathbf{t}_F, \mathbf{p}_K, \mathbf{t}_O^{(1)}, \dots, \mathbf{t}_O^{(t)} \right]^T.$$

Again we define matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ and the vector \mathbf{c} .

$\tilde{\mathbf{A}}$ is a block diagonal matrix whose diagonal blocks are

$$\tilde{\mathbf{A}}_D^{(i)} = \gamma\sigma \left[\mathbf{A}_I^{(i)} | \mathbf{A}_C^{(i)} | \mathbf{A}_H^{(i)} \right] \text{ for } i = 1, \dots, t \text{ and} \\ \tilde{\mathbf{A}}_D^{(t+1)} = \gamma\sigma \left[\mathbf{A}_W | \mathbf{A}_F | \mathbf{0}_K | \mathbf{A}_O^{(1)} | \dots | \mathbf{A}_O^{(t)} \right].$$

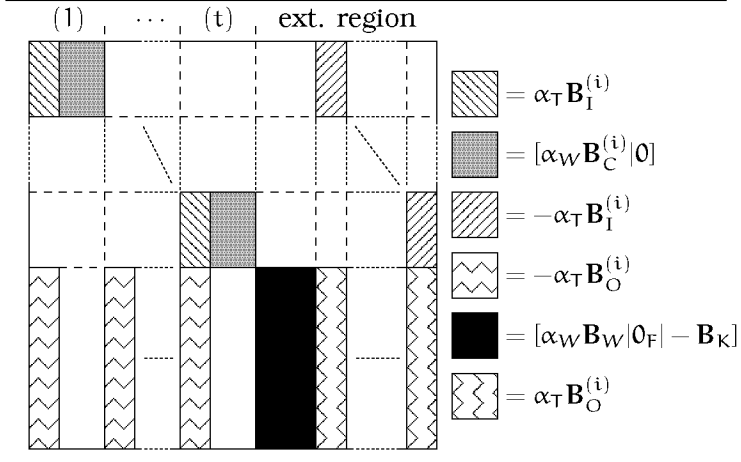
The structure of $\tilde{\mathbf{B}}$ and the meaning of its blocks is shown in Figure 5.4.

The constant \mathbf{c} becomes

$$\mathbf{c} := \left[-\alpha_W \mathbf{B}_C^{(1)} \mathbf{t}_O - \mathbf{B}_H^{(1)} \mathbf{p}_H, \dots, -\alpha_W \mathbf{B}_C^{(t)} \mathbf{t}_O - \mathbf{B}_H^{(t)} \mathbf{p}_H, \right. \\ \left. -\alpha_W \mathbf{B}_W \mathbf{t}_O - \mathbf{B}_F \mathbf{p}_F + \gamma\sigma \mathbf{A}_K \mathbf{t}_K^4 \right]^T.$$

With these definitions we obtain again a nonlinear operator φ as described by (5.7), of which we are interested in finding a zero.

Figure 5.4 Structure of matrix $\tilde{\mathbf{B}}$.



5.3 Numerical Results

OVEN FROM
ABB RESEARCH
CENTER

We perform the simulation on the *complex model* of the oven shown in Figure 5.1. This oven was built by the research center of the ABB company in Dättwil/Switzerland for test purposes. This oven consists of 4 protection tubes each holding to 2 heat elements. Each interior region contains 1080 surface elements, the exterior region has 2420 surface elements. The whole discretized model consists of 6740 surface elements.

Figure 5.5 shows the structure of the two matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$. Each of them is of size 6740×6740 . Matrix $\tilde{\mathbf{A}}$ shows the block diagonal structure described before. $\tilde{\mathbf{B}}$ has the special structure shown in Figure 5.4.

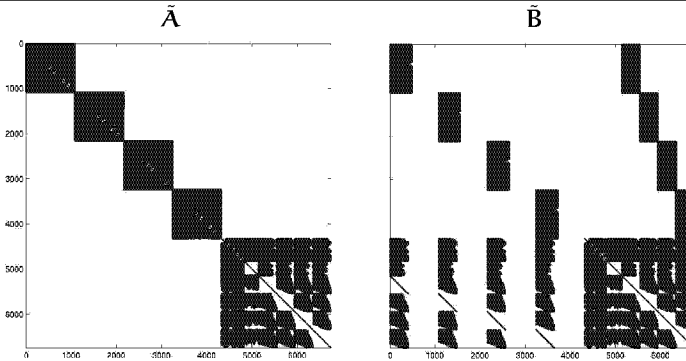
PHYSICAL
VALUES

The total power radiation which is equally distributed over the heat elements is 18 kW. The outside temperature is set to 300 K, the temperature of the cold regions is 600 K. The heat flux on the floor is set to 0 W. Initial values for the temperatures on the walls and the ceiling are 1670 K, 1700 K on the floor, 1800 K on the tube walls and 2000 K on the heat elements.

ALGORITHMS

The engineers at ABB were happy with results containing 6 correct digits. This is why every tested algorithm had to achieve

Figure 5.5 Structure of matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ for the given problem.



a relative residual norm of 10^{-6} . We tested the following algorithms to perform the simulation

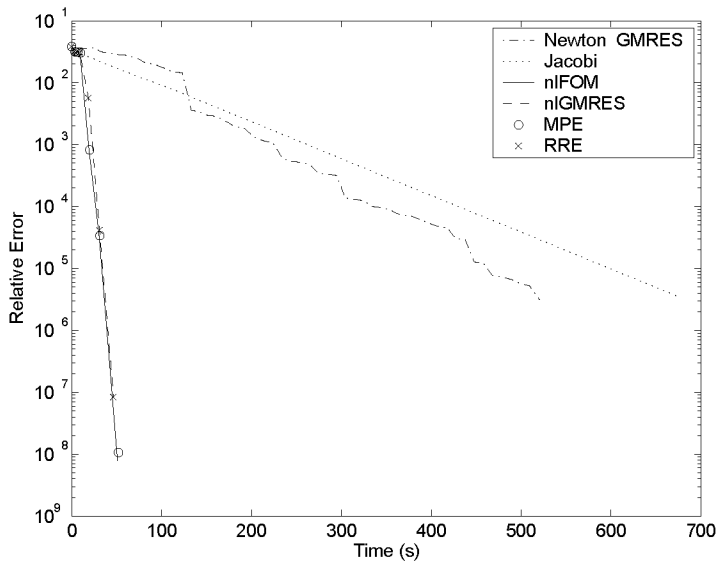
Newton-GMRES This method has been proven to be very effective on the Chandrasekhar problem in previous chapters. We also chose it, since it is related to the new algorithms: These can be viewed as incomplete Henrici methods. Newton-GMRES is an incomplete Newton method. For both Newton's method and the Henrici method the same convergence statement exist. The same parameters were used as in the introduction to control the convergence behavior.

Nonlinear Jacobi Up to now this method has been used in the parallel implementation we dealt with in (21) and (22). Nonlinear Jacobi consists of an inner iteration performing a local Newton iteration on every vector component. The outer iteration is the actual Jacobi process. For this numerical experiment Jacobi was implemented in Matlab. The number of steps for the inner iteration was chosen to be constantly 15. With fewer Newton steps Jacobi did converge to the desired tolerance.

Nonlinear FOM and GMRES, MPE and RRE Of course the new algorithms nonlinear FOM and GMRES had to be tested, but also their mathematically equivalent vector extrapolation companions MPE and RRE to see if the new algo-

rithms make an improvement. All these algorithms used nonlinear Jacobi as the preconditioner (the base iteration respectively). However for these algorithms we were able to use a version of Jacobi that performed only one step of the Newton iteration on each component — an algorithm that would not have converged to the desired tolerance. For all these algorithms the same mechanism was used to control the convergence as for Newton-GMRES.

Figure 5.6 Development of the relative error for different algorithms computing the simulation of the Deglor oven of the ABB research center in Dättwil/Switzerland. The times were taken on an Intel Pentium IV machine running at 1.7 GHz rate equipped with 512 MByte Memory, SuSE Linux 8.0 and Matlab 6.5.



RESULTS

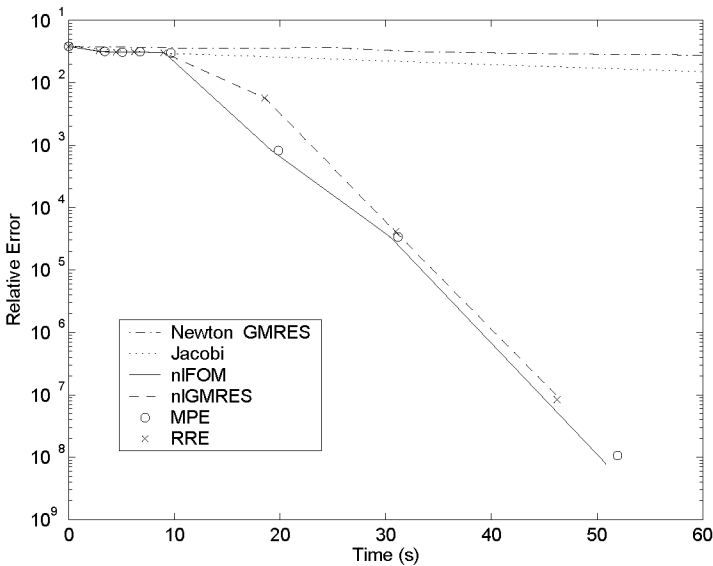
The results of the numerical experiment are shown in Figure 5.6. We were not able to get an old Matlab version for our Linux machine that is able to count flops. However, we have seen in the concluding section of Chapter 4 that time and flops corre-

late very well for the tested algorithms. Therefore we measured time instead of counting flops.

The surprise is that Newton-GMRES did not perform very well on the given problem although we used the same algorithm as in the previous chapters and we used the same convergence control mechanism. Newton-GMRES took 520 seconds to converge to the requested tolerance.

We never expected the nonlinear Jacobi algorithm to converge very well. It converged very linearly and delivered a solution at the requested tolerance after 677 seconds.

Figure 5.7 A closer look on the results for the methods nonlinear FOM, nonlinear GMRES, MPE and RRE.



A closer look on the performance of the other algorithms is shown in Figure 5.7.

These algorithms were about ten times faster than Newton-GMRES and nonlinear Jacobi. They roughly performed equally well on the given problem. The fastest algorithm was nonlinear GMRES. It computed the result to the requested tolerance after 47 seconds. Nonlinear FOM needed 51 seconds. However,

it returned a result about one magnitude better than GMRES. The time differences to their mathematical equivalent vector extrapolation companions MPE and RRE are neglectable.

5.4 Conclusion

In this thesis we have combined linear Krylov space methods with the vector extrapolation approach in order to derive nonlinear Krylov space methods that are numerically more stable than the vector extrapolation methods are.

To do this we have presented the theory of Krylov space methods. We have categorized Krylov space methods with respect to three properties: matrix type, projection method and Krylov space generation. Since nonlinear systems of equations do not consist of a single matrix, the first property does not count. We have described and discussed in detail the two projection approaches (orthogonalization, minimization) and the two Krylov space generation methods (Arnoldi, Lanczos bi-orthogonalization) used today for linear systems with general matrices. Thus we have presented four linear Krylov space methods:

FOM an Arnoldi method using orthogonalization,

GMRES an Arnoldi method using minimization,

BiCG the basic Lanczos method using orthogonalization,

QMR the basic Lanczos method using quasi-minimization.

We have shown that all vector extrapolation methods can be regarded as incomplete Henrici methods. We have proven a Kantorovich like theorem for their convergence, which did not exist for Henrici methods. Together with existing results for Henrici's method such

as the proven local quadratic behavior by Ortega and Rheinboldt [28] and by Jbilou and Sadok [23] we have extended the perception of Henrici's method (and Steffensen's method as well) as a method equally powerful as Newton's method. Moreover, Henrici's method *does not require* the knowledge of the Jacobian!

While combining the linear Krylov space methods with the vector extrapolation methods, we have concentrated on Arnoldi methods and developed a *nonlinear Arnoldi process* consisting of a *nonlinear preconditioner*, a *Gram-Schmidt process* and basically a *matrix vector multiplication*. The replacement of the Arnoldi process in linear Arnoldi methods by the nonlinear Arnoldi process results in nonlinear Arnoldi methods. We have done this replacement in the two presented linear algorithms FOM and GMRES and thus developed a nonlinear FOM and a nonlinear GMRES algorithm. These algorithms have been tested on one theoretical problem, the Chandrasekhar H-equation, published by Kelley [25], and on a real world problem of the ABB Company [9, 22, 21]. In the first problem the algorithms have shown a performance comparable to that of incomplete Newton methods, as we have expected. In the second problem the algorithms have shown a much better performance.

We conclude the discussion of the new algorithms by pointing out some aspects:

ASPECTS OF
THE NEW
ALGORITHMS

Numerical performance Our new methods, that use the Krylov space approach, do not improve the numerical performance of vector extrapolation methods. But the numerical performance of our algorithms is comparable to vector extrapolation methods. Therefore they are not suitable for solving linear systems of equations. In fact, we believe that using the difference approach of another basic iteration is the limiting factor for their numerical performance. Therefore, we think that our methods do numerically the best they can do using this approach. However, since the Krylov space

paradigm is a topic, which many researchers are working on studying linear systems of equations, our work is a step towards connecting these two classes of algorithms in the nonlinear contact. We expect more such relations like we found. But this is left to future research.

Memory requirements The new algorithms use about the same amount of memory as their mathematically equivalent vector extrapolation companions and they do not require any more memory than their linear companions.

Newton's method One aim of this thesis was that the new algorithms beat Newton's method with respect to large dense systems of equations. Not only they are able to beat Newton's method in this respect. The last problem has also shown that there are cases, where incomplete Newton methods perform poorly whereas the new algorithms perform very well.

Parallelism The parallelism of the new algorithms is dependent on the parallelism of the preconditioner and the parallelism of the linear method it generalizes. The Gram-Schmidt process and the matrix vector multiplication that it uses, can be parallelized very well. If the preconditioner used is nonlinear Jacobi or Gauss-Seidel or SOR, the resulting method has good chances to parallelize very well.

Lanczos Methods As has been mentioned in Chapter 4 chances are good that similar considerations on the topological epsilon algorithm (or similar methods) like we did on Arnoldi methods lead to a nonlinear version of the Lanczos biorthogonalization. Such an approach has the possibility to improve the numerical performance and the memory requirements of the topological epsilon algorithm in an enormous way. In its basic version the TEA is neither numerically stable nor is it efficient in

memory usage. However, its linear Krylov space companion BiCG is very well known for using little memory, variants of the algorithm promise good numerical performance. Therefore the development of a nonlinear Lanczos biorthogonalization could improve the performance of the TEA a lot. Moreover, this approach would also result *for free* in a nonlinear QMR algorithm — a completely new solution method, since so far no equivalent vector extrapolation method is known for this Lanczos method.

... **until** {*comprehension*}
end PhDthesis.

Bibliography

- [1] A. C. Aitken. *Determinanten und Matrizen*. Bibliographisches Institut, Hochschultaschenbücher Verlag, Mannheim etc., 1969.
- [2] H. A. Antosiewicz and W. C. Rheinboldt. Numerical analysis and functional analysis. In J. Todd, editor, *Survey of Numerical Analysis*, pages 485 – 517. McGraw-Hill, New York, 1962.
- [3] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Science*. SIAM, Philadelphia, 1994.
- [4] E. Bonomi et al. Astrid: A programming environment for scientific applications on parallel vectorcomputers. Rapport GASOV No. 23, EPF Lausanne, January 1990.
- [5] R. Brent. Some efficient algorithms for solving systems of nonlinear equations. *SIAM Journal on Numerical Analysis*, 10:327 – 344, 1973.
- [6] C. Brezinski. Généralisation de la transformation de Shanks, de la table de Padé et de l' ϵ -algorithme. *Calcolo*, 12:317 – 360, 1975.
- [7] C. Brezinski and M. Redivo Zaglia. *Extrapolation Methods: theory and practice*. North-Holland, Amsterdam etc., 1991.

-
- [8] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400 – 408, 1982.
 - [9] W. Egli. Appendix 3: Simulation of deglor ovens above the melt — numerical model and solution method. *Deglor Handbook R&D*, 1994.
 - [10] M. Eiermann and O. G. Ernst. Geometric aspects in the theory of Krylov subspace methods. *Acta Numerica*, pages 251 – 312, 2001.
 - [11] S. C. Eisenstadt and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17:16 – 32, 1996.
 - [12] R. Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Proceedings of the Dundee Biennial Conference on Numerical Analysis 1974*, pages 73 – 89, New York, 1975. Springer.
 - [13] R. W. Freund and N. M. Nachtigal. QMR: A quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik*, 60:315–339, 1991.
 - [14] W. Gander, G. H. Golub, and D. Gruntz. Solving linear equations by extrapolation. In *Supercomputing, Trondheim 1989*, number 62 in *Comp. Systems Science*, pages 279–293, Berlin, 1990. Springer.
 - [15] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.
 - [16] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Number 17 in *Frontier in applied mathematics*. SIAM, Philadelphia, 1997.
 - [17] P. Henrici. *Elements of Numerical Analysis*. John Wiley & Sons, New York etc., 1964.

- [18] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.
- [19] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 4th edition, 1990.
- [20] I. C. Ipsen and C. D. Meyer. The idea behind Krylov methods. *American Mathematical Monthly*, 105(10):889 – 99, 1998.
- [21] L. Jaschke. Parallelization of Programs Simulating ABB's DEGLOR Ovens Above the Melt. Master's thesis, ETH Zürich, 1995.
- [22] L. Jaschke. Parallel simulation of an ash melting furnace. In B. Hertzberger and P. Sloot, editors, *High-Performance Computing and Networking*, volume 1225 of *Lecture Notes in Computer Science*, pages 282 – 292, Vienna, April 1997. Springer.
- [23] K. Jbilou and H. Sadok. Some results about vector extrapolation methods and related fixed-point iterations. *Journal of Computational and Applied Mathematics*, 36:385 – 398, 1991.
- [24] L. W. Johnson and D. R. Scholz. On Steffensen's method. *SIAM Journal of Numerical Analysis*, 5(2):296 – 302, June 1968.
- [25] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.
- [26] B. Noble and J. W. Daniel. *Applied Linear Algebra*. Prentice-Hall Inc., Englewood Cliffs (N.J.), 2nd edition, 1977.
- [27] D. P. O'Leary. Conjugate gradients and related KMP algorithms: The beginnings. In *Linear and Nonlinear Conjugate Gradient-related Methods*, pages 1 – 8. SIAM, 1996.

- [28] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York etc., 1970.
- [29] A. Ostrowski. Über Verfahren von Steffensen und Householder zur Konvergenzverbesserung von Iterationen. *Zeitschrift für Angewandte Mathematik und Physik*, 7:218 – 229, 19556.
- [30] V. A. Rios. Immer diese Rechnungen. In *Micky Maus*, number 23, pages 2–8. Ehapa Verlag, Stuttgart, 1983.
- [31] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comp.*, 37:105 – 126, 1981.
- [32] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1995.
- [33] J. W. Schmidt. Eine Übertragung der Regula Falsi auf Gleichungen in Banachräumen. *Zeitschrift für angewandte Mathematik und Mechanik*, 43:1–8, 97–110, 1963.
- [34] R. J. Schmidt. On the numerical solution of linear simultaneous equations by an iterative method. *Philosophical Magazine, Ser. 7*, 32(214):369 – 383, 1941.
- [35] D. Shanks. Non-linear transformations of divergent and slowly convergent sequences. *Journal of Mathematics and Physics*, 34:1 – 42, 1955.
- [36] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1994.
- [37] A. Sidi. Efficient implementation of minimal polynomial and reduced rank extrapolation methods.

Technical Report 620, Technion — Israel Institute of Technology, Department of Computer Science, April 1990.

- [38] J. Stoer. *Einführung in die Numerische Mathematik I*. Springer, Berlin etc., 4th edition, 1983.
- [39] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford Science Publications, 1988.
- [40] P. Wynn. On a device for computing the $e_m(s_n)$ transformation. *Mathematical Tables and Other Aids to Computation*, 10:91 – 96, 1956.

About the Author



Person

Name Leonhard Jaschke
Date of Birth March 24, 1971
Place of Birth Innsbruck, Austria
Nationality Austrian

Education

1995 – 2003 Ph.D. thesis at the Institute of Scientific Computing, ETH Zürich
1990 – 1995 Study in computer science (Informatik) at ETH Zürich; graduated as Dipl. Informatik-Ing. ETH
1981 – 1989 Akademisches Gymnasium, Innsbruck, Austria; AHS-Matura on Mai 31, 1989

Practical Work

since 2003 Software developer at Rola AG, Schlieren
1995 – 2002 Teaching and research assistant at the Institute of Scientific Computing, ETH Zürich
1994 Development of a website for the Database Research Group at ETH Zürich
1993 Development of a prototype of a product database for System-12 components at Alcatel STR

Interests and Skills

- *Information Systems* Databases, WWW
- *Theoretic Computer Science* Cryptology, theory of graphs
- *Numeric Computing* Nonlinear systems of equations, parallel programming
- *Computer Graphics, Programming, Teaching*
- *Culture* Music, opera, dancing, literature
- *Sports* Skiing, snowboarding, rowing, biking, swimming