Apollo 11 Reentry with the Ultraspherical Spectral Method

A Numerically Robust Spectral Method for Non-Linear Boundary Value Problems

Lucas Aschenbach

Supervisor: Prof. Dr. Folkmar Bornemann

Submission: September 15, 2023

A thesis presented for the degree of
Bachelor of Science

Department of Mathematics
TUM School of Computation, Information and Technology
Technical University of Munich

Eigenständigkeitserklärung
Declaration of Authorship

Ich erkläre hiermit, dass ich diese Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

München, September 18, 2023

Lucas Aschenbach

Contents

Introduction

Motivation

Solving linear boundary value problems (BVPs) stands as a foundational problem in numerical mathematics, holding significant relevance across a multitude of scientific and engineering domains. Historically, solution approaches for linear BVPs have been dominated by finite difference methods, finite element methods, and pseudo-spectral methods which all discretize the infinite dimensional operator, defining the differential equation, a priori and then solve the problem as a finite dimensional linear system which Olver and Townsend later termed the *discretize-then-solve* paradigm [1]. In 2012, Olver and Townsend proposed an alternative to this approach with the Ultraspherical Spectral Method [2] which defers discretization of the problem and instead works with an exact representation of the differential operator as a sparse and well-conditioned infinite dimensional matrix. Here, the discretization step is combined with the solution of the linear system made possible by an iterative QR decomposition algorithm which enables an a priori error estimator for the discretization error to dynamically determine the optimal truncation for the infinite linear system before solving it. This method was shown to be in some cases competitive with or even superior to the state of the art in terms of accuracy and efficiency and reopened the discussion about the viability of coefficient spectral methods for solving linear BVPs.

In 2014, the authors of [2] released an implementation of the Ultraspherical Spectral Method for the Julia programming language under the name *ApproxFun.jl* [3] which provides a versatile framework for solving linear BVPs as infinite matrices.

Aim of this Thesis

In this thesis, we build on the work in [2] and [1] and extend the Ultraspherical Spectral Method to non-linear BVPs using the Newton-Raphson method. The goal of this thesis is to develop a robust and efficient implementation of Newton's method as a Julia package on top of *ApproxFun.jl* and demonstrate its viability on a number of non-linear BVPs. To this end, we explore in particular two globalization strategies for the Newton-Raphson method according to Deuflhard [4], relaxation and continuation, to improve the robustness of the method for bad starting values.

Outline

The thesis is structured into three main sections, each with a specific focus, as outlined below:

Chapter 1. The first chapter is a brief primer into infinite dimensional linear algebra. The concept of infinite dimensional matrices will be introduced along with a description of the iterative QR decomposition algorithm and a priori error estimator used by the Ultraspherical Spectral Method. The linear solver will be demonstrated on the example of the Bessel functions which will be computed using their 3-term recurrence relation, represented as an infinite dimensional matrix.

Chapter 2. In the second chapter, the Ultraspherical Spectral Method is briefly introduced. This will be followed by a revision of local convergence results for Newton's method on Banach spaces in Deuflhard's affine covariant setting. On this basis several refinements of newtons method will be discussed including a damped Newton method and a continuation method exhibit a more robust convergence behavior, even for starting values farther away from the solution.

Chapter 3. In the third chapter, the Algorithms developed in chapter 2 will be tested on common test cases for non-linear BVP solvers to demonstrate the methods strengths and weaknesses. The final example here will be the notoriously sensitive Apollo 11 reentry problem which will provide important insights into the limitations of the here presented methods.

CHAPTER 1

Infinite Dimensional Linear Algebra

Solving infinite linear systems of equations is a fundamental part of the Ultraspherical Spectral Method. In this chapter, we will introduce the necessary concepts from infinite dimensional linear algebra and describe the iterative QR decomposition algorithm as well as the a priori error estimator used by the Ultraspherical Spectral Method.

1.1. Preliminaries

The primary objects of interest for this chapter will be infinite dimensional matrices. More specifically, semi-infinite matrices which are elements of the space $\mathbb{K}^{\mathbb{N}\times\mathbb{N}}$. Since only semi-infinite matrices occur in the context of the Ultraspherical Spectral Method, we will refer to them just as infinite matrices going forward. Futhermore, for simplicity we restrict ourselves to the case $\mathbb{K} = \mathbb{R}$. Infinite matrices can naturally be interpreted as linear operators between two sequence spaces. For an infinite matrix $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ we obtain $A : X \to Y$ by matrix multiplication where X and Y are normed sequence spaces. Of course, for the matrix multiplication to be well-defined each row a_n of A must satisfy $\sum_{i=0}^{\infty} \alpha_i^{(n)} \xi_i < \infty$ for all $(\xi_i) \in X$, $n \in \mathbb{N}$. In the future, when speaking of A as an operator on a space X, this requirement is always assumed to be satisfied.

We are interested in solving problems of the form

$$Ax = b \tag{1.1}$$

where $A : X \to Y$ is an infinite matrix and $b \in Y$ is a vector of the image space of A. The solution of this problem differs from its finite dimensional counterpart in several significant ways. First of all, it is important to note that in the infinite case, linear independence of the columns of A is no longer a sufficient criterion for bijectivity. For example, the matrix

$$A = \begin{pmatrix} 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & \ddots \\ & & & \ddots \end{pmatrix}$$

will produce the same output $Ax_1 = Ax_2 \equiv 1$ for $x_1 := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & \cdots \end{pmatrix}$ and $x_2 := \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & \cdots \end{pmatrix}$. One must therefore understand the underlying operator represented by A and check its invertibility to guarantee existence and uniqueness of a solution. Furthermore, the solution of (1.1) will be an infinite dimensional vector which means it can only ever be numerically computed in a truncated form. Which truncation is appropriate depends on the space the solution is viewed in. For example, if the solution is in ℓ^p with $1 \leq p < \infty$, we are guaranteed to find an n for any $\varepsilon > 0$ such that $\|S_n x - x\|_p < \varepsilon$ where S_n is the natural projection to the first n sequence elements while this is not possible for ℓ^∞. Generally, we are interested in minimizing the error $\|\hat{x} - x\|$, where $\hat{x}$ is the truncated solution vector and x the true solution vector. Coefficient spectral methods are operating on the coefficient space for some Schauder-basis, usually a polynomial- or the Fourier basis, which exhibits the property $\|S_n x - x\| \to 0$ for $n \to \infty$. This motivates an algorithm which truncates the solution after n components, ideally in an iterative fashion to allow for early termination if the error is sufficiently small or post correction steps otherwise.

Before discussing solution algorithms we define a few important classes of infinite dimensional matrices which will be of interest later on.

Definition 1.1. A Matrix $M \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ is called:

(1) *column finite* if for all $j \in \mathbb{N}$, there exists an $n_j \in \mathbb{N}$ such that $a_{ij} = 0 \ \forall i > n_j$.
(2) *row finite* if for all $i \in \mathbb{N}$, there exists an $m_i \in \mathbb{N}$ such that $a_{ij} = 0 \ \forall j > m_i$.

The sequences (n_j) and (m_i) are called *truncation indices* of M.

As we will see later, the Ultraspherical Spectral Method will always produce a column finite matrix to represent linear differential operators. Specifically, the matrices will have one of the two following structures.

DEFINITION 1.2 (Banded Matrices). Let $M \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ be a matrix.

(1) If there exists $k \in \mathbb{N}$ such that $M_{ij} = 0$ for all $i, j \in \mathbb{N}$ with $|i - j| > k$, then M is called *banded*.
(2) If there exists $k, l \in \mathbb{N}$ such that $M_{ij} = 0$ for all $i, j \in \mathbb{N}$ with $j - i > k$ and $i > l$, then M is called *almost banded matrix*.

Since we will extensively work with blocks of infinite matrices, we will use an index notation, familiar from programming languages such as Julia, for succinctly defining these blocks. For an infinite matrix A, we denote

$$(A)_{(a:b)(x:y)}$$

as the block ranging from rows a to b and columns x to y. For $a = b$ we simply write $(A)_{(a),(x:y)}$. Furthermore, the following operators will turn out handy for rewriting block extractions as matrix products

$$P_n = \left(\begin{array}{c|c} I_n & 0 \end{array}\right) \in \mathbb{R}^{n\times\mathbb{N}}, \quad E_n = \left(\begin{array}{c|c} \underbrace{0 \cdots 0}_{n} & I \end{array}\right) \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$$

Here P_n is the canonical projection to $\mathbb{R}^n$ and E_n the shift operator which shifts the components of a sequence to the left by one. The matrices I_n and I are understood to be the identity matrices of size n and $\mathbb{N}$, respectively.

1.2. Iterative QR Decomposition

We will now describe an algorithm for computing a QR-decomposition of a column finite matrix. This decomposition will produce the unique solution if A is invertible but will continue to produce a solution as long as the columns of A are linearly independent. This algorithm has a long history with its first appearance already in 1967 where it was described in a more specialized form by Olver [5]. In 2012, Olver and Townsend generalized the algorithm to the class of almost banded matrices for the Ultraspherical Spectral Method in [2]. Here, we describe the algorithm in its most general implementable form for column finite matrices.

QR Decomposition. Let $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ be a column finite matrix with linearly independent columns representing an operator $A : X \to Y$ with X, Y sequence spaces and Banach. Futhermore, let $(n_j) \in \mathbb{N}^{\mathbb{N}}$ be a monotonically increasing sequence of truncation indices with $n_j \geq j$ for the columns of A, i.e. $a_{ij} = 0$ for $i > n_j$. Let $k \in \mathbb{N}$ and A_k the k-th iteration in the algorithm. We assume that $(A_k \cdot P_k')_{ij} = 0$ for $i > j$, i.e. the first k columns of A_k are in upper triangular structure. The left upper $k \times k$ block will be denoted with $R_k := (A_k)_{(1:k)(1:k)}$. Using this upper triangular structure and the fact that the $(k+1)$-th column of A_k satisfies $a_{(i)(k+1)} = 0$ for $i > n_{k+1}$, computing a QR decomposition for the block $\tilde{A}_k := (A_k)_{(k+1:n_k)(k+1)}$

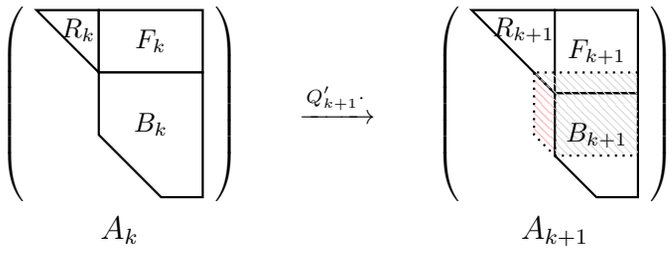
$$\tilde{A}_{k+1} = \tilde{Q}_{k+1} \cdot \underbrace{\begin{pmatrix} \rho_{k+1} \\ 0 \\ \vdots \end{pmatrix}}_{\tilde{R}_{k+1}}$$

gives all necessary building blocks for generating a QR decomposition of the first $k+1$ columns of A_k.

$$A_k \cdot P_{k+1}' = \underbrace{\begin{pmatrix} I_k & & \\ & \tilde{Q}_{k+1} & \\ & & I \end{pmatrix}}_{Q_{k+1}} \cdot \left(\begin{array}{cc} R_k & * \\ & \rho_{k+1} \\ \hline & 0 \end{array}\right)$$

Applying Q_{k+1}' to A_k then gives the next iteration A_{k+1} with $(A_{k+1}P_{k+1})_{ij} = 0$ for $i > j$, i.e. the first $k+1$ columns are in upper triangular structure. Also note that the truncation

indices (n_j) for A_k continue to be truncation indices of A_{k+1} since Q'_{k+1} only modifies rows with an index $i \leq n_{k+1}$ (precisely, the rows $k+1 : n_{k+1}$). By the monotonicity of (n_j) follows $(A_{k+1})_{ij} = (A_k)_{ij} = 0$ for all $i > n_j \geq n_{k+1}$ with $i \geq k+1$. In summary, the $(k+1)$-th iteration transforms A_k as follows:



Where the blocks R_{k+1} and F_{k+1} are given by

$$R_{k+1} = \left(\begin{array}{c|c} R_k & F_k e_1 \\ \hline 0 & \rho_{k+1} \end{array}\right), \quad F_{k+1} = \left(\begin{array}{c} F_k E' \\ \hline f_{k+1} \end{array}\right)$$

In passing, we also showed the feasibility of the algorithm by strong induction. Also, notice that we did not yet use the linear independence of the columns of A this step.

With this framing, we also managed to completely abstract away the inner proceedings of the local QR decomposition. Normally, either *Householder Reflection* or *Given's Rotations* are used for the reduction step which will produce an orthogonal matrix $Q_{k+1} = I - 2\omega\omega'$ with $\|\omega\| = 1$ and $Q_{k+1} = Q_{k+1}^{(1)} \cdots Q_{k+1}^{(l+1)}$ with $Q_{k+1}^{(i)}$ being a Given's Rotation, eliminating the $(k+1+i)$-th component of the $(k+1)$-th column in A_k, respectively. However, with this additional layer of abstraction it is also possible to dispatch specialized algorithms for specific local structures of the matrix without affecting the feasibility of the algorithm. This is particularly interesting when expanding $\tilde{A}_k$ to a batch of columns instead of applying the local QR decomposition to one column at a time. Adapting the algorithm to accommodate for batching is an easy task and does not affect its feasibility.

Remark. The requirement of column finiteness may also be dropped under certain conditions since Householder reflections can be generalized as infinite rank operators. This was explored by Trefethen in [6] for the case where the columns of A are elements of $\ell^2(\mathbb{N})$. However, in practice this requires knowledge of the indices at which the column tail can be bounded by a certain tolerance, essentially reducing the problem back to decomposing a column finite matrix.

Backwards Substitution. After the k-th iteration, we have a QR decomposition for the first k columns of A, i.e.

$$A \cdot P'_k = Q_1 \cdots Q_k \cdot \left(\frac{R_k}{0}\right)$$

By applying the inverses of the Q_i in reverse order to the right-hand side $b_k := Q'_k \cdots Q'_1 b$ and truncating after the first k rows $y_k := P_k \cdot b_k$, we can apply backwards substitution to the resulting system of equations to obtain an estimate for the first k components of x.

$$R_k x_k = y_k \tag{1.2}$$

Optimal Truncation. Now that we can compute an arbitrarily large QR decomposition of A, we need to answer the question of when to stop the algorithm such that solving the truncated system (1.2) gives an approximation of the solution to the equation $Ax = b$ within a predefined tolerance TOL. To this end, we will use the following theorem:

Theorem 1.3. *Let $x_k \in \mathbb{R}^{\mathbb{N}}$ be the vector obtained from (1.2) after the k-th iteration for the problem $Ax = b$ with $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$, $A : X \to Y$ a column finite matrix with linearly independent columns. Then*

$$\|\hat{x}_k - x\|_X \leq \left\|R^{-1}\right\| \|\hat{r}_k\|_Y \tag{1.3}$$

where $\hat{x}_k = P'_k x_k$ is the truncated solution, $R^{-1} = R_\infty^{-1}$ the limit of the inverse upper triangular blocks R_k^{-1}, and $\hat{r}_k = \begin{pmatrix} 0 & \cdots & 0 & r_k \end{pmatrix}^T$ the truncated part of b_k with k leading zeros and $r_k = (b_k)_{(k+1:\infty)}$

PROOF. It is clear that because of the triangle structure, R_k^{-1} inherits the property that the first k columns remain unchanged after the k-th iteration from R_k.

$$R_{k+1}^{-1} = \left(\begin{array}{c|c} R_k^{-1} & * \\ \hline 0 & * \end{array}\right)$$

Formally, this implies $P_k' R_k^{-1} = R^{-1} P_k'$. We can therefore write

$$\begin{aligned}
\|\hat{x}_k - x\|_X &= \left\|P_k' R_k^{-1} y_k - R^{-1} b_k\right\|_X \\
&= \left\|P_k' R_k^{-1} y_k - R^{-1}(P_k' y_k + \hat{r}_k)\right\|_X \\
&= \left\|(P_k' R_k^{-1} - R^{-1} P_k') y_k - R^{-1}\hat{r}_k\right\|_X \\
&= \left\|(P_k' R_k^{-1} - P_k' R_k^{-1}) y_k - R^{-1}\hat{r}_k\right\|_X \\
&= \left\|R^{-1}\hat{r}_k\right\|_X \\
&\leq \left\|R^{-1}\right\| \|\hat{r}_k\|_Y
\end{aligned}$$

□

COROLLARY 1.4. *Let $x_k \in \mathbb{R}^{\mathbb{N}}$ be the vector obtained from (1.2) after the k-th iteration for the problem $Ax = b$ with $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$, $A : \ell^2 \to \ell^2$ a column finite invertible matrix with a bounded inverse. Then*

$$\|\hat{x}_k - x\|_2 \leq \left\|A^{-1}\right\|_2 \|\hat{r}_k\|_2 \tag{1.4}$$

where $\hat{x}_k$, $\hat{r}_k$ as in Theorem 1.3.

For a linear problem where $\left\|A^{-1}\right\|_2 < \infty$, this provides a cheaply computable error bound for the discretization error. Most importantly, the error bound enables the QR decomposition algorithm to determine a priori, in each iteration before truncating and solving the linear system, a worst case error estimate at the current truncation. Hence, once the value of $\left\|A^{-1}\right\|_2 \|\hat{r}_k\|$ satisfies a given tolerance we may terminate the QR-decomposition and compute x_k by backwards substitution.

Implementation. For the implementation, there is one significant aspect left to be discussed. Since A and b are of infinite dimension, we can only ever have a finite subset of their entries stored in memory. The same is true for operations performed on infinite objects which cannot be reduced to finitely many arithmetic operations. The solution is to lazily evaluate such operations. Instead of immediately evaluating an operation, each operation is stored and separately performed for individual components of the output, once that component is explicitly required. For example the infinite QR decomposition $QR = A$ will be represented symbolically but the actual Q-factors will only be constructed once the transpose of the symbolic Q is multiplied to a right-hand-side b and will only be constructed as far as needed to meet the given tolerance. As a consequence, the algorithm and implementation have a "symbolic feel" to them as the operator equation is never truncated right up until the point when the error bound reaches the desired tolerance and x_k is computed.

Algorithm 1.1 is the algorithm for solving a column finite linear system of equations described in this section. It also explicitly articulates the lazy operations as they would be required on a finite memory machine.

Algorithm 1.1 Column Finite Linear Solver

$A_0 \leftarrow A$
$n \leftarrow$ truncation indices of A_0
$k \leftarrow 0$
while $\varepsilon >$ TOL **do**
    $k \leftarrow k + 1$
    $A_k \leftarrow A_{k-1}$
    if $k > 1$ **then**
        $A_k[1 : n_k, k] \leftarrow Q_{k-1}' \cdots Q_1' \cdot A_k[1 : n_k, k]$ ▷ Lazyly apply Qs to A
    end if
    $\tilde{A}_k \leftarrow A_{k-1}[k : k + n_k, k + 1]$
    $\tilde{Q}_k, \tilde{R}_k \leftarrow$ QR decomposition of $\tilde{A}_k$

12: $A_k[k : k + n_k, k + 1] \leftarrow \tilde{R}_k$
13: $b[k : k + n_k] \leftarrow \tilde{Q}'_k \cdot b[k : k + n_k]$
14: $\varepsilon \leftarrow \left\|A^{-1}\right\|_2 \|b[k + 1 : \infty]\|_2$
15: **end while**
16: Solve $R_k x_k = b[1 : k]$ for x_k
17: **return** x_k

REMARK. For ease of notation, we have omitted the reshaping of the Q'_i in line 8.

Practical implementations of Algorithm 1.1 such as *ApproxFun.jl* and *InfiniteLinearAlgebra.jl* which implement a specialized variation for almost banded matrices neglect the prefactor $\left\|A^{-1}\right\|_2$ and use just $\|\hat{r}_k\|$ to estimate the error. In a practical experiment, as we will also see in the next section, we observed that $\|\hat{r}_k\|$ actually approaches the true error as the error bound approaches zero. This could be explained by the fact that the error, which is given exactly by $\left\|R^{-1}\hat{r}_k\right\|$ as seen in the proof of Theorem 1.3, only depends on the columns of R^{-1} after the k-th column since $\hat{r}_k$ has k leading zeros. This can be written as

$$\|\hat{x}_k - x\| \leq \left\|R^{-1}E'_k\right\| \|r_k\|$$

Therefore, if $\left\|R^{-1}E'_k\right\| \to 1$ for $k \to \infty$ were to hold, then $\|\hat{r}_k\|$ would indeed be a good approximation of the error. However, this requires some more investigation and is left for future work.

1.3. Bessel 3-Term Recurrence

Historically, Algorithm 1.1 was first developed for the solution of linear difference equations:

$$x_{i+k} + \alpha_{k-1}x_{i+k-1} + \cdots + \alpha_0 x_i = b_i, \quad i \in \mathbb{N}$$

While in some cases this type of problem may be solved by simple forward extrapolation from k starting values, this approach will be subject to numerical instability in cases where the solution values are rapidly decreasing for higher orders due to *elimination of leading digits*. Initially, *Miller's recurrence algorithm* [8] was developed to combat this problem by filling the recurrence constraints backwards, starting at a chosen maximum order M. The obvious drawback of Miller's algorithm is that the truncation M must be chosen a priori and may therefore be non-optimal. This motivated the development of the described algorithm. The first version of the algorithm used Gaussian Elimination without Pivoting and was developed by Olver in [5] specifically for 2nd order difference equations ($k = 2$). This algorithm served as the foundation for the more general algorithm described in [2] and here.

We demonstrate Algorithm 1.1 on the example of the Bessel functions of the first kind whose rapidly decreasing function values for higher orders make them unsuitable for forward extrapolation. They arise as separable solutions to Laplace's equation and the Helmholtz equation in cylindrical or spherical coordinates and are commonly defined as solutions to the ordinary differential equation

$$x^2 \frac{d^2}{dx^2} J_\alpha(x) + x \frac{d}{dx} J_\alpha(x) + (x^2 - \alpha^2) J_\alpha(x) = 0$$

where $J_\alpha(0) = 0$ and $\alpha \in \mathbb{C}$ is the order of the function. Using the Frobenius Ansatz, one can obtain an explicit power series representation for the Bessel functions of integer order and derive the generating function

$$\exp\left(\frac{x}{2}\left(t - \frac{1}{t}\right)\right) = \sum_{n=-\infty}^{\infty} J_n(x) t^n \tag{1.5}$$

From (1.5) it is easy to derive the following 3-term recurrence relation

$$J_{n-1} - \frac{2n}{x} J_n + J_{n+1} = 0, \quad n \in \mathbb{N}$$

For a unique classification of $J_n(x)$, we need to impose additional constraints. Since the linear difference equation is ill conditioned as an initial value problem but not as a boundary

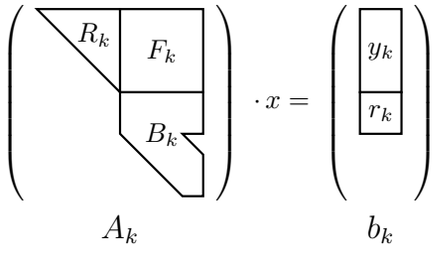
value problem, we employ the following normalizing condition

$$J_0(x) + 2\sum_{k=1}^{\infty} J_{2k}(x) = 1$$

which too is derived from (1.5) by setting $t = 1$. This boundary condition uniquely classifies the Bessel functions of the first kind together with the 3-term recurrence relation. Additionally, it has the advantage that it connects Bessel functions up to an arbitrarily large order which means the resulting problem is well conditioned regardless of the optimal truncation order n_{opt}. This gives us the linear system

$$\underbrace{\begin{pmatrix} 1 & 0 & 2 & 0 & 2 & 0 & \cdots \\ 1 & -\frac{2}{x} & 1 & & & & \\ & 1 & -\frac{4}{x} & 1 & & & \\ & & 1 & -\frac{6}{x} & 1 & & \\ & & & 1 & -\frac{8}{x} & 1 & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}}_{A} \cdot \begin{pmatrix} J_0(x) \\ J_1(x) \\ J_2(x) \\ J_3(x) \\ J_4(x) \\ J_5(x) \\ \vdots \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}}_{b}$$

The matrix A represents an operator $A : \ell^1 \to c_0$ which is well defined since all columns of A are in ℓ^∞ and for all $x_n \in \ell^1 : nx_n \to 0$. Since the 3-term recurrence together with the normalizer uniquely identify $J_n(x)$, the operator is injective. Furthermore, the matrix A is almost banded and by extension column finite, hence we may apply Algorithm 1.1 to obtain the unique solution. For this specific example, the k-th step has the form



Since all but the first value of b are zero such that $\operatorname{supp} b_n \subset \{1, \cdots, n+1\}$ the error bound from Corollary 1.4 simplifies to

$$\|\hat{x}_k - x\|_2 \leq \left\|A^{-1}\right\|_2 \|r_k\|_2 = \left\|A^{-1}\right\|_2 |(r_k)_1|$$

An empirical analysis shows that for all $x > 0$, there exists a $d_x > 0$ such that the smallest singular value $\sigma_{\min}(A) \geq d_x$ and equivalently $\left\|A^{-1}\right\|_2 = \sigma_{\max}(A^{-1}) \leq 1/d_x < \infty$, i.e. A^{-1} bounded on ℓ^2.

Figure 1 shows the ℓ^2-error of the Bessel approximation at $x = 100$ for different truncation orders n. One immediately sees that the stopping criterion stops the algorithm shortly after the error plateaus at near machine precision $\left\|A^{-1}\right\|_2 \cdot \varepsilon_{\text{mach}}$, determining a near optimal truncation order, in this case $n = 150$.

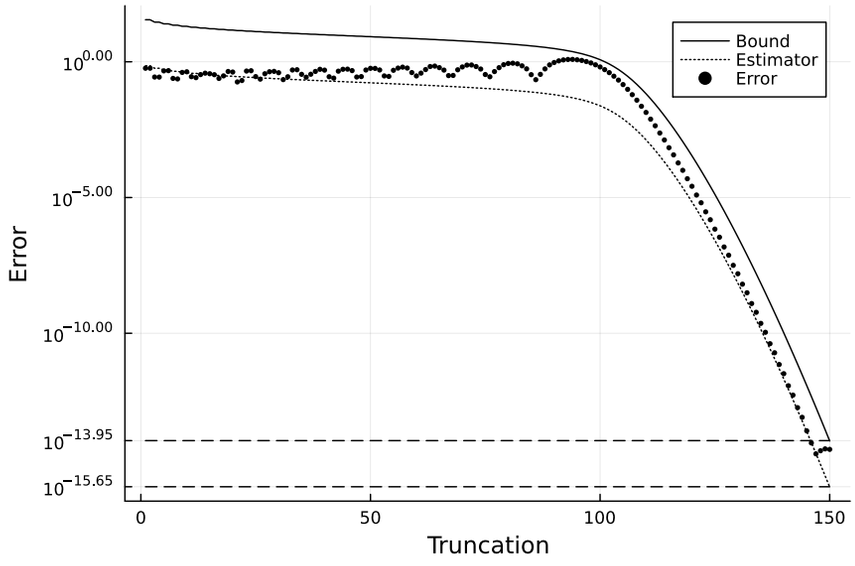


FIGURE 1. Error of the Bessel function approximation at $x = 100$ for increasing truncation orders n. The iteration stops once the error bound reaches $\left\|A^{-1}\right\|_2 \cdot \varepsilon_{\text{mach}}$ with $\left\|A^{-1}\right\|_2 \approx 50.0$

CHAPTER 2

Spectral Methods for Non-Linear Boundary Value Problems

Spectral methods solve differential equations by approximating the solution with a linear combination of global basis functions

$$u(x) \approx \sum_{n=0}^{N} \alpha_n \varphi_n(x).$$

This structure makes them particularly suitable for solving linear differential equations as the problem must only be solved for the basis functions φ_n which is often much easier for common choices like a polynomial or the Fourier basis. In particular, the basis is chosen to be closed with respect to the operator such that the problem can be represented as a linear system of equations. Additionally, the method inherits the convergence behavior of the interpolation which is super-algebraic for a sufficiently smooth solution u and some polynomial or the Fourier basis. Spectral methods are further divided into *collocation methods* and *coefficient methods*. Collocation methods enforce the differential equation at a set of collocation points $\{x_i\}_{i=0}^{N}$ and operate on the *value space*, while coefficient methods enforce the differential equation in the space spanned by the basis functions $\{\varphi_n\}_{n=0}^{N}$ and operate on the *coefficient space*. It was a long held belief that collocation methods are superior to coefficient methods for linear differential equations with variable coefficients. However, this belief was challenged by Olver and Townsend with the development of the *Ultraspherical Spectral Method* in [2] which in some cases is competitive or even superior to collocation methods.

2.1. The Ultraspherical Spectral Method

The Ultraspherical Spectral Method is a coefficient method which uses multiple bases selected from the family of ultraspherical polynomial bases $C^{(\lambda)} := \{C_i^{(\lambda)}\}_{i=0}^{\infty}$. The fundamental idea is to exploit the sparse recurrence relation

$$\frac{d^\lambda}{dx^\lambda} T_n = \begin{cases} 2^{\lambda-1} n(\lambda-1)!\, C_{n-\lambda}^{(\lambda)} & , \lambda \leq n \\ 0 & , \lambda \geq n+1 \end{cases}$$

to allow for the solution to be interpolated using Chebyshev polynomials $T_n = C_n^0$ which have a near optimal Lebesgue constant while maintaining a sparse and well-conditioned differentiation matrix by performing a change of basis to $C^{(\lambda)}$ as part of applying the λ-th derivative. The resulting differentiation operator for the λ-th derivative is given by

$$\mathcal{D}_\lambda = 2^{\lambda-1} n(\lambda-1)! \begin{pmatrix} \overbrace{0 \;\cdots\; 0}^{\lambda \text{ times}} & \lambda & & & \\ & & \lambda+1 & & \\ & & & \lambda+2 & \\ & & & & \ddots \end{pmatrix}, \quad \lambda \in \mathbb{N}$$

with the special case $\mathcal{D}_0 := I$ and maps between the spaces $\mathcal{D}_\lambda : C^{(0)} \to C^{(\lambda)}$. When mixing terms which map to different spaces, all terms are elevated to the space with the highest parameter λ using the recurrence

$$T_n = \begin{cases} \frac{1}{2}(C_n^{(1)} - C_{n-2}^{(1)}) & , n \geq 2 \\ \frac{1}{2} C_1^{(1)} & , n = 1 \\ C_0^{(1)} & , n = 0 \end{cases}, \quad C_n^{(\lambda)} = \begin{cases} \frac{\lambda}{\lambda+1}(C_n^{(\lambda+1)} - C_{n-2}^{(\lambda+1)}) & , n \geq 2 \\ \frac{\lambda}{\lambda+1} C_1^{(\lambda+1)} & , n = 1 \\ C_0^{(\lambda+1)} & , n = 0 \end{cases}$$

The resulting basis transformation operators are also sparse and given by

$$\mathcal{S}_0 = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & \\ & & \frac{1}{2} & 0 & \ddots \\ & & & \frac{1}{2} & \ddots \\ & & & & \ddots \end{pmatrix}, \quad \mathcal{S}_\lambda = \begin{pmatrix} 1 & 0 & -\frac{\lambda}{\lambda+1} & & \\ & \frac{\lambda}{\lambda+1} & 0 & -\frac{\lambda}{\lambda+1} & \\ & & \frac{\lambda}{\lambda+1} & 0 & \ddots \\ & & & \frac{\lambda}{\lambda+1} & \ddots \\ & & & & \ddots \end{pmatrix}, \quad \lambda \in \mathbb{N}$$

where $\mathcal{S}_\lambda : C^{(\lambda)} \to C^{(\lambda+1)}$ for $\lambda \in \mathbb{N}_0$. Lastly, variable coefficients $a^{(n)}(x)$ are handled by interpolating the coefficient function using ultraspherical polynomials and embedding the coefficients into a linear multiplication operator. Through laborious transformations, one arrives at the operator definition

$$\left(\mathcal{M}_\lambda[a^{(n)}]\right)_{ij} = \sum_{k=\max(0,j-i)}^{j} a^{(n)}_{2k+i-j} c^\lambda_k(j, 2k+i-j)$$

where

$$c^\lambda_k(i,j) = \frac{i+j+\lambda-2k}{i+j+\lambda-k} \frac{(\lambda)_k (\lambda)_{i-k} (\lambda)_{j-k}}{k!(i-k)!(j-k)!} \frac{(2\lambda)_{i+j-k}}{(\lambda)_{i+j-k}} \frac{(i+j-2k)!}{(2\lambda)_{i+j-2k}}$$

and $(\lambda)_k = \frac{(\lambda+k-1)!}{(\lambda-1)!}$ is the *Pochhammer symbol*. As noted by Olver and Townsend [2], the given formula for c^λ_k cannot be used directly due to overflow issues for $i, j \geq 70$ and is in practice replaced by an expression matching factors in the numerator and denominator of similar magnitude for a numerically stable evaluation. The operator $\mathcal{M}_\lambda[a^{(n)}]$ operates on the coefficient space $C^{(\lambda)}$ and satisfies $\left(\mathcal{M}_\lambda[v] \cdot u\right)(x) = v(x) \cdot u(x)$. This operator is banded with bandwidth N when $a^{(n)}$ is sufficiently well approximated by the given ultraspherical polynomials up to degree N. In practice, the coefficients are calculated for the Chebyshev basis which are cheaply available in $\mathcal{O}(n \log(n))$ where n is the chosen coefficient cutoff using the discrete cosine transform and then elevated to the desired ultraspherical basis with parameter λ using $\mathcal{S}_{\lambda-1} \cdots \mathcal{S}_0$. The boundary conditions are embedded into the final operator by means of *boundary bordering* where N functionals, representing N boundary conditions, are added as rows to the top of the operator. Dirichlet boundary conditions for example may be realized using the evaluation functionals for each boundary value a, b

$$\begin{pmatrix} T_0(a) & T_1(a) & T_2(a) & \cdots \\ T_0(b) & T_1(b) & T_2(b) & \cdots \end{pmatrix} u = \begin{pmatrix} g(a) \\ g(b) \end{pmatrix}$$

With the described operators, a boundary value problem of the form

$$\mathcal{L}u = f, \quad \mathcal{B}u = g$$

where $\mathcal{L}$ is a normalized linear differential operator

$$\mathcal{L} = \sum_{n=0}^{N} a^{(n)}(x) \frac{d^k}{dx^k}, \quad a^{(N)} \equiv 1$$

and $\mathcal{B}$ a linear boundary operator with N rows can be transformed into the infinite linear system of equations

$$\begin{pmatrix} \mathcal{B} \\ \mathcal{D}_N + \sum_{n=0}^{N-1} \left(\mathcal{S}_{N-1} \cdots \mathcal{S}_n\right) \cdot \mathcal{M}_n[a^{(n)}] \cdot \mathcal{D}_n \end{pmatrix} u = \begin{pmatrix} g \\ \left(\mathcal{S}_{N-1} \cdots \mathcal{S}_0\right) \cdot f \end{pmatrix} \tag{2.1}$$

This infinite linear system is almost banded such that, using Algorithm 1.1, the problem can be solved in $\mathcal{O}(m^2 n_{\text{opt}})$ operations where m is the total bandwidth which is in most cases determined by the largest number of Chebyshev points needed to resolve the coefficient functions of $\mathcal{L}$ and n_{opt} is the optimal truncation or the number of Chebyshev coefficients needed to resolve the solution of the problem. Using a diagonal preconditioner, the condition number of the almost banded matrix in (2.1) can be bounded with respect to the ℓ^2_λ norm from 2.1. As a result, the infinite QR decomposition algorithm in 1.1 whose stability is invariant under column scaling when using Householder reflections or Given's rotations for the local QR decomposition, is also stable for this problem. One should note that, while the error bound of Algorithm 1.1 does bound the discretization error for the infinite matrix, this does not include the error introduced when interpolating variable coefficients of the

differential operator. However, for sufficiently smooth coefficient functions, this error is negligible.

For the following chapter it is also important to understand, which space the resulting operators are defined on. Since the underlying problem is involves N-th order differential equation, a sufficient degree of regularity must be assumed of the elements in the solution space. Since the operator is operating on the coefficient space for an ulteraspherical basis, the space ℓ^2_λ is a natural choice for the solution space.

DEFINITION 2.1. The space $\ell^2_\lambda \subset \mathbb{R}^{\mathbb{N}}$ is defined as the Banach space with norm

$$\|u\|_{\ell^2_\lambda} := \left(\sum_{n=0}^{\infty} |u_n|^2 (k+1)^{2\lambda}\right)^{1/2}$$

for $u = (u_n)_n \in \ell^2_\lambda$.

The space ℓ^2_λ is also a Hilbert space with the inner product

$$\langle u, v\rangle_{\ell^2_\lambda} := \sum_{n=0}^{\infty} u_n v_n (k+1)^{2\lambda}$$

This will be useful in the next section.

A Julia implementation of the Ultraspherical Spectral Method is available in the *ApproxFun.jl* [3] package. The package hides most of the heavy machinery discussed here behind a simple backslash-solve interface and leverages Julia's multiple dispatch to allow for a simple and readable definition of differential operators and boundary conditions.

2.2. Local Newton Methods on Banach Spaces

The Ultraspherical Spectral method, as seen in the previous section, provides a robust method of solving linear boundary value problems by reducing the problem to a well-conditioned infinite system of linear equations which can be solved stably to optimal precision using Algorithm 1.1. In this section we build on the Ultraspherical Spectral Method as a base solver for linear problems to implement a Newton algorithm on the ultraspherical coefficient space ℓ^2_λ.

Newton's method is a method for solving non-linear equations of the form $f(x) = 0$ by repeated linearization and solution of the resulting linear system of equations. Simpson was the first to describe the method in 1740 for continuously differentiable scalar valued functions f as the iterative scheme

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Although the method was initially developed for finite dimensional problems, it can be generalized to a broad class of infinite dimensional operators between Banach spaces. For this, we must first generalize the concept of a derivative to infinite dimensional spaces.

DEFINITION 2.2 (Fréchet Derivative). Let X, Y be normed spaces, $U \subset X$ open, and $F : U \to Y$ an operator. F is called *Fréchet differentiable* at $x \in U$ if there exists a bounded linear operator $F' : U \to Y$ such that

$$\lim_{h\to 0} \frac{\|F(x+h) - F(x) - F'(x)(h)\|_Y}{\|h\|_X} = 0$$

$F'(x)$ is called the *Fréchet derivative* of F at x.

If F is Fréchet differentiable at all $x \in U$, then F is called *Fréchet differentiable* on U.

If a Fréchet derivative exists it is unique, hence we will speak of *the* Fréchet derivative for Fréchet differentiable functions. For an n-th order continuously Fréchet differentiable operator, we use the notation $F \in \mathcal{C}^n$. One can easily verify that the Fréchet derivative satisfies the usual properties of a derivative such as the sum, product, and chain rule. As a result, the Fréchet derivative of an operator can be computed using standard automatic differentiation techniques. Since in boundary value problems the number of equations is usually the same as the number of unknown functions, it is practical to employ the much simpler to implement forward mode differentiation. In *ApproxFun.jl* [3], this is realized with the `DualFun` structure which is a generalization of Clifford's dual numbers for operators between function spaces. A `DualFun` instance $\texttt{DualFun}(u, T)$ is an element of a function

space $X \times L(X_0, X)$ where $u \in X$ represents the manipulation of some (unknown) initial value $u_0 \in X_0$ by a chain of (unknown) operators $F_n \circ \cdots \circ F_1$ and T the Fréchet derivative for those manipulations. Similar to dual numbers, `DualFun` exploits the chain rule

$$\frac{dF}{du_0}(u) = F'(u) \circ \frac{du}{du_0} = F'(u) \circ T$$

to iteratively update the Fréchet derivative for new manipulations

$$u_0 \xrightarrow{F_1\circ} \cdots \xrightarrow{F_n\circ} u \xrightarrow{F\circ} F(u)$$
$$I \xrightarrow{F_1'\circ} \cdots \xrightarrow{F_n'\circ} T \xrightarrow{F'\circ} F'(u) \circ T$$

This translates to $F(\texttt{DualFun}(u, T)) = \texttt{DualFun}(F(u), F'(u) \circ T)$. By overriding atomic operations on `DualFun` such as addition and multiplication with an easy to determine Fréchet derivative, the Fréchet derivative for a more complicated F may by constructed by interpreting F itself as a chain of atomic manipulations and iteratively updating the derivative with the atomic Fréchet derivatives using the update rule. Since `DualFun` is designed to behave as a regular functional, *ApproxFun.jl*'s implementation of lazyly evaluated operators carries over to automatic differentiation such that the evaluation and Fréchet derivative are also computed lazily.

The generalization of Newton's method for Banach spaces was first studied by Kantorovich in 1948. For a Fréchet differentiable operator $F : U \to Y$ with $U \subset X$ open where X, Y are Banach spaces the Newton update is given by

$$x_{k+1} = x_k - \Delta x_k, \quad \Delta x_k := F'(x_k)^{-1} F(x_k)$$

where $F'(x_k)$ is the inverse of the Fréchet derivative of F at x_k. Algorithm 2.1 describes the simplest version of Newton's method for solving $F(x) = 0$ for a Fréchet differentiable operator F between Banach spaces.

Algorithm 2.1 Newton's Method

1: $x_0 \leftarrow$ initial guess
2: $k \leftarrow 0$
3: **while** $\|F(x_k)\| >$ TOL **do**
4: Compute Fréchet derivative $F'(x_k)$
5: Solve $F'(x_k)\Delta x_k = -F(x_k)$ for Δx_k
6: $x_{k+1} \leftarrow x_k + \Delta x_k$
7: $k \leftarrow k + 1$
8: **end while**
9: **return** x_k

For the Ultraspherical Spectral Method, F' will be an almost banded matrix of the same shape as F such that line 5 can be solved efficiently to optimal precision with Algorithm 1.1. As discussed by Deuflhard in [4], when dealing with infinite dimensional problems, inexact Newton methods are generally the correct framework due to the discretization error introduced when truncating the problem to a finite dimensional subspace. For the Ultraspherical Spectral Method, this effect is at the very least less pronounced since the discretization error is bounded by the error bound introduced in Corollary 1.4 which is required to satisfy a given tolerance. Remember that even in practical implementations as in [3], the method models the problem exactly, disregarding the approximation error of the coefficient functions, until the point of solving the linear system thanks to lazy evaluation. For this reason, we will neglect the discretization error in the following analysis and remain in the framework of exact Newton methods.

Local Convergence. We will now discuss the local convergence properties of Newton's method. For this, we choose the framework of *affine covariance* which was developed by Deuflhard in [4]. Newthon's method has the special property of being invariant under linear transformations of the image space and to commute with affine transformations of the domain space. Specifically, let X, Y be Banach spaces, $U \subset X$ open, $F : U \to Y$ Fréchet differentiable, and $x^* \in U$ a solution of $F(x) = 0$. Then for $F_A(x) := AF(x)$ with A an invertible linear operator

$$-F_A'(x_k)^{-1} F_A(x_k) = -F'(x_k)^{-1} F(x_k)$$

This property is called *affine covariance*. Futhermore, for $y_k = T(x_k)$, $F_T(x) := F(T(x))$ with $T(x) := Bx + b$ with B an invertible linear operator and $b \in X$

$$\begin{aligned} y_{k+1} &= y_k - F'(y_k)^{-1} F(y_k) \\ &= Bx_k - B\left(F' \circ T(x_k) B\right)^{-1} F \circ T(x_k) + b \\ &= Bx_k - BF_T'(x_k)^{-1} F_T(x_k) + b = T(x_{k+1}) \end{aligned}$$

This property is called *affine contravariance*. As a consequence, neither linear transformations of the image space, nor affine transformations of the domain space affect the convergence of Newton's method. Therefore, any convergence criteria should also be invariant under these transformations. This was Deuflhard's key insight which led to the development of convergence theories invariant under these transformations. Achieving invariance with respect to both transformations was shown to be impossible. Therefore, separate theories were developed for different types and classes of transformations, two of which are the *affine convariance* theory and the *affine contravariance* theory. The *affine covariance* theory formulates any criteria to be invariant under transformations of the image space and is the theory we will be most interested in here as it delivers results for the error in the domain space. The *affine contravariance* theory on the other hand formulates criteria to be invariant under linear transformations of the domain space and naturally delivers results for the residual error.

The following theorem is a refined version of the affine covariant Newton-Mysovskikh theorem, generalized for Banach spaces and gives important insight into the local convergence behavior of Newton's method.

THEOREM 2.3 (Refined affine covariant Newton-Mysovskikh, [9, Thm. 1.5]). *Let $F : U \in X \to Y$ be continuously Fréchet differentiable, with $F'(x)$ invertible for all $x \in U$, $U \subset X$ open and convex. Assume there exists $x_0 \in U$ and $\alpha_0, \omega \geq 0$ such that*

$$\left\| F'(x_0)^{-1} F(x_0) \right\| \leq \alpha_0$$

$$\left\| F'(z)^{-1} \left(F'(y) - F'(x)\right)(y - x) \right\| \leq \omega \left\| y - x \right\|^2 \quad \text{for all collinear } x, y, z \in U \tag{2.2}$$

$$h_0 := \alpha_0 \frac{\omega}{2} < 1$$

$$\overline{B(x_0, \rho)} \subset U \text{ with } \rho := \alpha_0 \sum_{j=0}^{\infty} h_0^{2^j - 1} \leq \frac{\alpha_0}{1 - h_0}$$

Then

1. *The sequence (x_k) of Newton iterates remains in $B(x_0, \rho)$ and converges to a solution $x^* \in \overline{B(x_0, \rho)}$.*
2. *x^* is the unique solution of $F(x) = 0$ in the region $U^* := \bigcup_{k=0}^{\infty} B(x_k, 2/\omega) \cap U$.*
3. *The following error estimates hold:*

$$\left\| x_{k+1} - x_k \right\| \leq \frac{\omega}{2} \left\| x_k - x_{k-1} \right\|^2 \tag{2.3}$$

$$\left\| x_{k+1} - x^* \right\| \leq \frac{\omega}{2} \left\| x_k - x^* \right\|^2 \tag{2.4}$$

$$\left\| x_k - x^* \right\| \leq \frac{\left\| x_{k+1} - x_k \right\|}{1 - \omega/2 \left\| x_{k+1} - x_k \right\|} \tag{2.5}$$

For a close enough starting value x_0, Theorem 2.3 guarantees *existence* and *uniqueness* of a solution x^* inside the neighborhood U^* and that the sequence of Newton iterates converges q-quadratically to that solution. A formal proof of the theorem may be found in [9].

Termination Criteria. In Algorithm 2.1 we used the simple residual termination criterion $F(x_k) \approx 0$. Using the affine covariant Mysovskikh theorem, we can derive more sophisticated termination criteria that are based on the error in the domain space. The inequality (2.5) from the theorem already gives us an upper bound for the error independent of the unknown solution x^*. The only caveat is that the bound depends on the Lipschitz constant ω which is generally difficult to compute.

A way to avoid having to work with ω at all is to take advantage of the fact that for a stopping criteria, we are only interested in the error bound when x_k is already very close to the solution x^*. In this case, the error bound simplifies to

$$\|x_k - x^*\| \leq \frac{\|\Delta x_k\|}{1 - \omega/2 \underbrace{\|\Delta x_k\|}_{\to 0}} \doteq \|\Delta x_k\|, \qquad \text{for } k \to \infty$$

which means that the error bound is approximately equal to the magnitude of the next Newton update. The error also bounds the magnitude of the Newton update from below which precludes premature termination. Therefore, for $0 < \text{TOL} \ll \frac{2}{\omega}$ we can use the criterion

$$\|\Delta x_k\| < \text{TOL} \tag{2.6}$$

A more sophisticated approach is to additionally use result (2.3) from the Mysovskikh theorem to predict the Lipschitz constant sufficiently well to use the error bound (2.5) directly. Following Deuflhard in [4], we define $e_k := \frac{\omega}{2} \|\Delta x_k\|$. From (2.3) follows $e_k \leq e_{k-1}^2$ such that we obtain the predictor

$$\|x_k - x^*\| \leq \frac{\|\Delta x_k\|}{1 - [e_k]}, \qquad \text{with} \qquad e_k \geq e_k \cdot \frac{e_k}{e_{k-1}^2} = \frac{e_k^2}{h_{k-1}^2} = \frac{\|\Delta x_k\|^2}{\|\Delta x_{k-1}\|^2} := [e_k]$$

and the resulting termination criterion

$$\frac{\|\Delta x_k\|}{1 - [e_k]} < \text{TOL} \tag{2.7}$$

In our experiments, (2.7) did not reliably provide better error estimates than (2.6) and both criteria started to agree on the same error bounds long before the error reached machine precision. Therefore, we will use the simpler (2.6) for the remainder of this work.

There is still a little room for improvement in these two termination criteria. The expressions (2.6) and (2.7) provide an estimate for the error of the current iterate while the next iterate has already been computed. The local q-quadratic convergence of Newton's method from Mysovskikh's theorem gives the relation

$$\|x_{k+1} - x\| \leq \frac{\omega}{2} \|x_k - x\|^2 \leq \frac{\omega}{2} \|\Delta x_k\|^2 < \frac{\omega}{2} \text{TOL}^2$$

Therefore, the criterion $\|\Delta x_k\| \ll \sqrt{\text{TOL}}$ is already sufficient to guarantee $\|x_{k+1} - x\| < \text{TOL}$. This will only save one iteration in the best case, but may still make a noticeable difference when each iteration is expensive as is the case here where every iteration requires the solution of a linear boundary value problem.

2.3. Global Newton Methods

It is well known that the convergence of Newton's method strongly depends on the chosen starting value x_0. In this section, we will no longer assume, that the starting value x_0 is *sufficiently close* to the solution x^* and explore several globalization concepts to still guarantee convergence for Newton's method.

Monotonicity. An important foundational step for globalizing Newton's method is to develop computable metrics that indicate whether the algorithm is progressing towards the solution. Naturally, progress in the k-th iteration would be characterized as moving closer to the solution x^*, i.e.

$$\|x_{k+1} - x^*\| < \|x_k - x^*\|$$

Mysovskikh's theorem holds for a local domain U around a solution x^* such that the error bounds (2.6) and (2.7) are only valid for x_k sufficiently close to x^*. To monitor the progress of the algorithm for general starting values, we need to define a global criterion which does not rely on the local convergence properties from Mysovskikh and is affine covariant. Since $F(x^*) = 0$ it is clear that $\|F(x_{k+1})\| < \|F(x_k)\|$ indicates at least progress with respect to

the residual error. In the spirit of affine covariance, we therefore introduce the *general level function*

$$T(x|A) := \frac{1}{2}\left\|AF(x)\right\|^2 \tag{2.8}$$

where A is any bounded bijective linear operator whose domain includes the range of F. Ideally, each Newton iterate should satisfy the *general monotonicity condition*

$$T(x_{k+1}|A) < T(x_k|A) \quad \text{for all } A \in L(Y) \text{ bijective} \tag{2.9}$$

It is not clear that a successor satisfying (2.9) even exists in general. However, if the image space Y is a Hilbert space such as ℓ^2_λ with $\|\cdot\|$ induced by the inner product $\langle\cdot,\cdot\rangle$, then we can show the following lemma.

LEMMA 2.4. *Let $F'(x_k)$ be non-singular, $F(x_k) \neq 0$, and $A \in L(H)$ bijective with H a Hilbert space. With $\Delta x_k = -F'(x_k)^{-1}F(x_k)$, there exists some $\mu > 0$ such that*

$$T(x_k + \lambda\Delta x_k|A) < T(x_k|A) \quad \textit{for all } 0 < \lambda < \mu \tag{2.10}$$

A proof of the lemma was given by Birkisson in [10, Lem. 2.5.1]. The lemma shows that for any A, a small enough step in the Newton direction will satisfy the associated monotonicity condition. As we will see, some choices of A are more useful as a convergence monitor than others. For a global monotonicity condition we want to choose A_k in each iteration such that the resulting condition is as relaxed as possible while still being a good indicator of progress. To this end, we consider the following theorem which provides a *monotonicity bound*, meaning a bound for the contraction quotient $\frac{T(x_{k+1}|A)}{T(x_k|A)}$, for arbitrary general level functions as a function of A and a damping factor λ.

THEOREM 2.5. *Let $F : U \to Y$ be continuously Fréchet differentiable, with $F'(x)$ invertible for all $x \in U$, $U \subset X$ open and convex. Assume the sublevel set of $T(x_k|A)$ is contained in U for a fixed A and*

$$\left\|F'(x)^{-1}\left(F'(y) - F'(x)\right)(y - x)\right\| \leq \omega\left\|y - x\right\|^2$$

Then, for $\lambda \in [0, \min\left\{1, 2/\overline{h}_k\right\}]$

$$\left\|AF(x_k + \lambda\Delta x_k)\right\| \leq (1 - \lambda + \frac{1}{2}\lambda^2\overline{h}_k)\left\|AF(x_k)\right\| \tag{2.11}$$

where

$$\overline{h}_k := h_k \cdot \kappa\left(AF'(x_k)\right), \quad h_k := \omega\left\|\Delta x_k\right\|$$

is the Kantorovich quantity.

Here, $\kappa(A) := \|A\|\left\|A^{-1}\right\|$ is the condition number of A. The following proof is a slightly altered version of the proof in [10, pp. 65-66].

PROOF. To shorten the notation, we drop the k subscript for the duration of the proof. We start by inflating the inner term of the left hand side.

$$\begin{aligned} AF(x + \lambda\Delta x) &= AF(x + \lambda\Delta x) - F(x) - F'(x)\Delta x \\ &= A\left(\int_0^\lambda F'(x + s\Delta x) - F'(x)\Delta x\, ds - (1 - \lambda)F'(x)\Delta x\right) \\ &= AF'(x)\int_0^\lambda F'(x)^{-1}(F'(x + s\Delta x) - F'(x))\Delta x\, ds \\ &\quad - (1 - \lambda)AF(x) \end{aligned} \tag{2.12}$$

Applying the norm to both sides and using the local Lipschitz property of F' yields the result

$$\begin{aligned}\|AF(x+\lambda\Delta x)\| &\leq \|AF'(x)\| \int_0^\lambda \left\|F'(x)^{-1}(F'(x+s\Delta x)-F'(x))\Delta x\right\| ds \\ &\quad + (1-\lambda)\|AF(x)\| \\ &\leq \|AF'(x)\| \frac{1}{2}\lambda^2\omega \|\Delta x\|^2 + (1-\lambda)\|AF(x)\| \\ &= \|AF'(x)\| \frac{1}{2}\lambda^2 h \left\|(AF'(x))^{-1}AF(x)\right\| + (1-\lambda)\|AF(x)\| \\ &\leq \left(1-\lambda+\frac{1}{2}\lambda^2\overline{h}\right)\|AF(x)\|\end{aligned}$$

□

The Lipschitz condition here is a special case of the Lipschitz condition (2.2) from Mysovskikh's Theorem 2.3. However, still no proximity to the solution x^* was assumed. The derived bound (2.11) is useful for understanding the local behavior of the level functions depending on the choice of A and λ. Here, we consider the regular Newton algorithm which corresponds to $\lambda = 1$. In this case, the bound simplifies to

$$\|A_kF(x_k+\Delta x_k)\| \leq \frac{\overline{h}_k}{2}\|A_kF(x_k)\|$$

This bound is proportional to the condition number $\kappa(A_kF(x_k)) \geq 1$. Hence, to obtain a minimal monotonicity bound, the natural choice is $A_k = F'(x_k)^{-1}$ which yields a minimal condition number of 1, leading to the final bound

$$\left\|\overline{\Delta x}_{k+1}\right\| \leq \frac{\omega}{2}\|\Delta x_k\|^2 \tag{2.13}$$

where $\overline{\Delta x}_{k+1} := -F'(x_k)^{-1}F(x_k+\Delta x_k)$ is the simplified Newton update. The resulting level function for $A_k = F'(x_k)^{-1}$ is called the *natural level function* and is associated with the *natural monotonicity condition*

$$T(x_{k+1}|F'(x_k)^{-1}) \leq T(x_k|F'(x_k)^{-1})$$

which is equivalent to

$$\left\|\overline{\Delta x}_{k+1}\right\| \leq \|\Delta x_k\| \tag{2.14}$$

The natural level function also gives a good indicator of progress. For $F \in \mathcal{C}^2(U)$, follows by Taylor expansion

$$F'(x^*)^{-1}F(x_k) \doteq x_k - x^*$$

such that for $x_k \to x^*$ the natural monotonicity condition becomes

$$\|x_{k+1}-x^*\| \dot{\leq} \|x_k-x^*\|$$

The natural monotonicity test was first introduced by Deuflhard in 1972 and has become a popular convergence monitoring tool in Newton methods. In practice, it proved useful to introduce a monotonicity bound $\bar{\Theta}$ which either relaxes or tightens the natural monotonicity condition (2.14) to

$$\left\|\overline{\Delta x}^\lambda_{k+1}\right\| \leq \bar{\Theta}\|\Delta x_k\|$$

Commonly used choices for $\overline{\Theta}$ are given in Table 1.

$\overline{\Theta}$	Description
2	Derived as a necessary condition for local convergence in Mysovkikh's Theorem
1	Natural monotonicity condition
1/2	Strict monotonicity condition
1/4	Derived from simplified Newton method [4, p. 55] which is equivalent for the first step; commonly enforced in continuation methods

TABLE 1. Commonly used Monotonicity Bounds

In the finite dimensional case, this test is also known to be very cost effective in its computational complexity as the matrix factorization of $F'(x_k)$ is already available from computing the Newton step. This transfers to the infinite dimensional case only in part as the optimal truncation index n_{opt} may increase for the changed right-hand-side $-F(x_{k+1})$ and may therefore require additional iterations of Algorithm 1.1. This can and will happen in practice and may as much as double the cost of the iteration. This phenomenon may for example be observed in the nerve pulse problem discussed in section 3.2. An important option in the implementation should therefore be to disable the monotonicity test. Algorithm 2.2 is a refined version of the initial Algorithm 2.1 and includes the natural monotonicity test with adjustable monotonicity bound and error based stopping criteria.

Algorithm 2.2 Refined Newton's Method

$\bar{\Theta} \leftarrow$ monotonicity bound
$k_{\max} \leftarrow$ maximum number of iterations
$x_0 \leftarrow$ initial guess
for $k = 0, \dots, k_{\max}$ **do**
  Compute Fréchet derivative $F'(x_k)$
  Solve $F'(x_k)\Delta x_k = -F(x_k)$ for Δx_k ▷ Newton Update
  $x_{k+1} \leftarrow x_k + \Delta x_k$
  if $\|\Delta x_k\| < \text{TOL}$ **then**
    return x_{k+1}
  end if
  if $\bar{\Theta} = \infty$ **then**
    continue
  end if
  Solve $F'(x_k)\overline{\Delta x}_{k+1} = -F(x_{k+1})$ for $\overline{\Delta x}_{k+1}$ ▷ Simplified Newton Update
  if $\left\|\overline{\Delta x}_{k+1}\right\| < \text{TOL} \wedge \|\Delta x_k\| < \sqrt{10 \cdot \text{TOL}}$ **then**
    return $x_{k+1} + \overline{\Delta x}_{k+1}$
  end if
  $\Theta_k \leftarrow \frac{\left\|\overline{\Delta x}_{k+1}\right\|}{\|\Delta x_k\|}$
  if $\Theta_k > \bar{\Theta}$ **then** ▷ Monotonicity Test
    return
  end if
end for

Damped Newton Methods. The natural monotonicity test helps monitor whether the regular Newton algorithm is progressing towards a solution. However, Lemma 2.4 also showed that for any A, the Newton update can always be rescaled such that the next iterate satisfies the associated monotonicity test. This motivates a *damped Newton method* which chooses its stepsizes to always conform to the respective monotonicity bound. In the previous section, we used Theorem 2.5 to derive a choice of A_k for which we could guarantee the largest decrease of the associated level function $T(x_k|A_k)$ for a regular Newton step, i.e. $\lambda = 1$. We will now use the same bound to derive an optimal A_k and λ_k for a maximal guaranteed decrease in the level function $T(x_k|A_k)$ for a damped Newton step, i.e. $0 < \lambda_k \leq 1$. We recall the monotonicity bound (2.11) from Theorem 2.5

$$\|AF(x_k + \lambda\Delta x_k)\| \leq (1 - \lambda + \frac{1}{2}\lambda^2 \overline{h}_k)\, \|AF(x_k)\|$$

where

$$\overline{h}_k := h_k \cdot \kappa\left(AF'(x_k)\right), \quad h_k := \omega\, \|\Delta x_k\|$$

is the *Kantorovich quantity*. Minimizing the upper bound with respect to λ yields the locally optimal damping factor as a function of A.

$$\bar{\lambda}_k(A) := \min\left\{1, \frac{1}{\overline{h}_k}\right\} \tag{2.15}$$

Plugging this damping factor back into (2.11) yields the lowest monotonicity bound under A

$$\frac{\left\|AF(x_k+\overline{\lambda}_k(A)\Delta x_k)\right\|}{\|AF(x_k)\|} \leq \begin{cases} 1-\dfrac{1}{2\overline{h}_k} & , \quad \text{for } \overline{h}_k \leq 1 \\ \overline{h}_k/2 & , \quad \text{for } \overline{h}_k > 1 \end{cases}$$

This monotonicity bound is a more general version of (2.13) and also decreases monotonically for smaller $\overline{h}_k$. As we know, $\overline{h}_k$ is proportional to the condition number $\kappa\left(AF'(x_k)\right) \geq 1$ such that, as in the previous section, $A_k = F'(x_k)^{-1}$ becomes the natural choice for A_k. We see that the natural level functions continue to allow for the best local monotonicity bound under variable $\lambda \in (0,1]$. This means, a damping factor selected on the basis of (2.15) provides, by local approximation, the best candidate for producing an x_k which passes the natural monotonicity test.

In summary, as per the model in Theorem 2.5, the optimal damping strategy for a Fréchet differentiable function $F : U \subset X \to Y$ with F' non-singular on U which satisfies the local Lipschitz condition

$$\left\|F'(x_k)^{-1}(F'(x)-F'(x_k))(x-x_k)\right\| \leq \omega_k \|x-x_k\|^2 \quad \text{for all } x \in U$$

as in (2.5) is given by

$$x_{k+1} = x_k + \lambda_k \Delta x_k, \quad \lambda_k := \min\left\{1, \frac{1}{h_k}\right\}, \quad h_k = \omega_k \|\Delta x_k\|$$

Implementing this theoretical damping strategy directly faces the same challenges as implementing the theoretical error estimate in (2.5) as the damping factor λ_k in each iteration depends on the Kantorovich quantity h_k and by extension the Lipschitz constant ω_k which is difficult to compute. To circumvent this problem, Deuflhard proposed a predictor-corrector scheme in [4] which estimates the local Kantorovich quantity h_k in each iteration to compute the damping factor λ_k. He showed, that it is sufficient to correctly estimate only the first binary digit of h_k in order to guarantee natural monotonicity of the damped Newton step. This result is known as the *bit counting lemma*.

Using the intermediate step (2.12) from the proof of Theorem 2.5, Deuflhard arrived at a correction term for an existing estimate of the optimal damping factor λ

$$[\lambda_k] := \min\left\{\frac{\lambda}{2}, \frac{1}{[h_k]}\right\}, \quad [h_k] := [\omega_k]\|\Delta x_k\| = \frac{2\left\|\overline{\Delta x}_k - (1-\lambda)\Delta x_k\right\|}{\lambda^2 \|\Delta x_k\|} \leq h_k \tag{2.16}$$

Since ω_k is the supremum over all $x \in U$ and we are only sampling over a single x, the corrector only provides a lower bound for $\overline{h}_k$ and therefore overshoots $\overline{\lambda}$. However, the damping factor λ_k will at least halve for each recursive application. To complete the predictor-corrector scheme, Deuflhard proposed a predictor to provide a good initial estimate λ_k^0 in each iteration. The derivation of the predictor requires a slightly modified Lipschitz condition

$$\left\|F'(x_k)^{-1}(F'(x)-F'(x_k))y\right\| \leq \overline{\omega}_k \|x-x_k\| \|y\|$$

where y is assumed to be sufficiently similar in direction to $x - x_k$. Under this condition

$$\begin{aligned} \left\|\overline{\Delta x_k} - \Delta x_k\right\| &= \left\|F'(x_{k-1}^{-1} - F'(x_k)^{-1})F(x_k)\right\| \\ &= \left\|F'(x_k)^{-1}(F'(x_k)-F'(x_{k-1}))\overline{\Delta x}_k\right\| \\ &\leq \overline{\omega}_k \lambda_{k-1} \|\Delta x_{k-1}\| \left\|\overline{\Delta x}_k\right\| \end{aligned}$$

This inequality suggests the predictor

$$[\lambda_k^0] := \min\left\{1, \frac{1}{[\overline{\omega}_k]\|\Delta x_k\|}\right\}, \quad [\overline{\omega}_k] := \frac{\left\|\overline{\Delta x}_k - \Delta x_k\right\|}{\lambda_{k-1}\|\Delta x_{k-1}\|\left\|\overline{\Delta x}_k\right\|} \leq \overline{\omega}_k \tag{2.17}$$

which also overshoots $\overline{\lambda}_k$. This only leaves the choice of an initial damping factor λ_0^0 in the first iteration. Intuitively, this value should reflect how well the problem at hand is approximated by a linear model with $\lambda_0^0 = 1$ corresponding to mildly nonlinear problems and $\lambda_0^0 \ll 1$ corresponding to highly nonlinear problems.

In practice, we apply the described predictor-corrector scheme in each iteration until the next iterate for the current damping factor λ_k passes the natural monotonicity test. In our experiments with non-linear boundary value problems, the natural monotonicity test with a monotonicity bound of $\bar{\Theta} = 1$ was often too strict and caused the damping factors to become too small, resulting in premature termination. A better choice for the monotonicity

bound was $\bar{\Theta} = 2$. Algorithm 2.3 implements the described predictor-corrector scheme in the context of a damped Newton method.

Algorithm 2.3 Damped Newton's Method

$\bar{\Theta} \leftarrow$ monotonicity bound
$\lambda_0^0 \leftarrow$ initial damping factor
$k_{\max} \leftarrow$ maximum number of iterations
$x_0 \leftarrow$ initial guess
for $k = 0, \dots, k_{\max}$ **do**
  Compute Fréchet derivative $F'(x_k)$
  Solve $F'(x_k)\Delta x_k = -F(x_k)$ for Δx_k
  if $\|\Delta x_k\| < \text{TOL}$ **then**
    $x_{k+1} \leftarrow x_k + \Delta x_k$
    return x_{k+1}
  end if
  if $k \geq 1$ **then**
    $\lambda_k^0 \leftarrow \min\left\{1, \frac{1}{\overline{\omega}_k \|\Delta x_k\|}\right\}, \quad \overline{\omega}_k \leftarrow \frac{\left\|\overline{\Delta x}_k - \Delta x_k\right\|}{\lambda_{k-1}\|\Delta x_{k-1}\|\left\|\overline{\Delta x}_k\right\|}$ ▷ Predictor
  end if
  $j \leftarrow 0$
  while true **do**
    Solve $F'(x_k)\overline{\Delta x}_{k+1}^j = -F'(x_k + \lambda_k^j \Delta x_k)$ for $\overline{\Delta x}_{k+1}^j$
    if $j = 0 \wedge \left\|\overline{\Delta x}_{k+1}^j\right\| < \text{TOL} \wedge \|\Delta x_k\| < \sqrt{10 \cdot \text{TOL}} \wedge \lambda_k^j = 1$ **then**
      $x_{k+1} \leftarrow x_k + \Delta x_k + \overline{\Delta x}_{k+1}^j$
      return x_{k+1}
    end if
    $\Theta_k \leftarrow \left\|\overline{\Delta x}_{k+1}^j\right\| / \|\Delta x_k\|$
    $\lambda_k^{j+1} \leftarrow \min\left\{1, \frac{1}{h_k}\right\}, \quad h_k \leftarrow \frac{2\left\|\overline{\Delta x}_{k+1}^j - (1-\lambda_k^j)\Delta x_k\right\|}{\left(\lambda_k^j\right)^2 \|\Delta x_k\|}$ ▷ Corrector
    if $\Theta_k < \bar{\Theta}$ **then**
      $\lambda_k \leftarrow \lambda_k^j$
      $x_{k+1} \leftarrow x_k + \lambda_k \Delta x_k$
      $k \leftarrow k + 1$
      break
    else if $\lambda_k^j = \lambda_{\min}$ **then**
      return
    else
      $\lambda_k^{j+1} \leftarrow \max\left\{\lambda_{\min}, \min\left\{\frac{\lambda_k^j}{2}, \lambda_k^{j+1}\right\}\right\}$
      $j \leftarrow j + 1$
    end if
  end while
end for

2.4. Continuation Methods

Continuation methods, sometimes also referred to as Homotopy methods, are a common technique for globalizing iterative algorithms which require sufficient proximity to the solution for the starting value to guarantee convergence. The idea is to parametrize a problem such that it becomes easier to solve for a specific parameter and then gradually move the parameter towards the target parameter which gives the original problem. This approach is often easy to apply as such parametrization naturally arise in many scientific and engineering problems in the form of physical parameters which influence reaction speeds, material properties, etc. and can be used to control the problem difficulty.

Formally, we shift our perspective from the standalone problem $F(x) = 0$ to the setting of the problem family

$$H(x, \lambda) = 0$$

where $H : U \times \mathbb{R} \to Y$ is a continuously Fréchet differentiable function with $U \subset X$ open. The problem of interest is to find $x \in U$ such that $H(x, \lambda^*) = 0$ for $\lambda^* \in \mathbb{R}$. Presuming the existence of a solution (x^*, λ^*) and that the partial Fréchet derivative at the solution $H_x(x^*, \lambda^*)$ is invertible, the implicit function theorem for Banach spaces [11, Thm. 13.22] guarantees the existence of a unique continuously Fréchet differentiable path $\overline{x} : I \subset \mathbb{R} \to X$ such that

$$H(\overline{x}(\lambda), \lambda) = 0$$

for all λ in a path-connected neighborhood I of λ^*. This *continuation path* is the unique solution of the ordinary differential equation

$$x'(\lambda) = -H_x(x(\lambda), \lambda)^{-1} H_\lambda(x(\lambda), \lambda) \tag{2.18}$$

with initial condition $x(\lambda^*) = x^*$. For a given starting solution $H(x_0, \lambda_0) = 0$ on the continuation path such that $x_0 = \overline{x}(\lambda_0)$ with $\lambda_0 \in I$, following the path will lead to the solution of the original problem. Considering $\overline{x}$ is the solution of the initial value problem (2.18), one may get the idea to compute the continuation path by numerical integration of the differential equation with the initial value $\overline{x}(\lambda_0) = x_0$. However, in [4, p.234], Deuflhard explains why this approach is not recommended in practice. Instead, a predictor-corrector scheme is employed which uses local information about the continuation path to compute an estimate for a new point on the path as close to the target solution as possible and then uses Newton's method to correct the estimate and walk back onto the continuation path.

The predictor $\hat{x}(\lambda_i)$ for the current parameter λ_i can be classified by its local approximation order of the continuation path.

Definition 2.6 (Prediction Order). A continuation method has prediction order p if the predictor $\hat{x}(\lambda)$ satisfies

$$\|\hat{x}(\lambda) - \overline{x}(\lambda)\| \leq \eta \sigma^p$$

where $\sigma = \hat{\lambda} - \lambda$ is the stepsize and η is a constant independent of σ.

The classical continuation method which uses the previous solution as the starting value for the new λ_i has the predictor $\hat{x}_i(\lambda) = x_i$ and therefore has prediction order $p = 1$. A better estimate is produced by the tangent predictor which chooses an estimate along the tangent of the continuation path at the previous solution. The tangent predictor is given by

$$\hat{x}_i(\lambda_i) = x_i + \sigma_i x_i'$$

and has prediction order $p = 2$. From the tangent predictor, it is an easy task to extrapolate prediction methods of arbitrary order using higher order local Taylor approximations. However, these are rarely used in practice as they require accurate higher order derivatives of H which are often difficult to obtain in practice. [4, p. 240]

To realize the tangent predictor, we need a method of computing the tangent of the continuation path at a given point. The initial value problem (2.18) provides a numerically computable expression for the tangent direction by solving the infinite dimensional linear system

$$H_x(x_i, \lambda_i) \cdot \overline{x}'(\lambda_i) = -H_\lambda(x_i, \lambda_i)$$

The partial Fréchet derivatives with respect to x and λ can be computed using automatic differentiation with `DualFun` and `Dual`[1], respectively. The resulting system is almost banded for differential operators and can be solved using the iterative QR-decomposition in Algorithm 1.1. An issue with this expression is that the continuation path is computed as a function of λ such that turning points where the path reverses its direction with respect to λ will return $\|\overline{x}'(\lambda)\| = \infty$. This problem may be fully evaded if we instead consider a regular parametrization $(\overline{x}(\tau), \lambda(\tau))$ with $H(\overline{x}(\tau), \lambda(\tau)) = 0$ which gives the initial value problem

$$\underbrace{\begin{pmatrix} H_x(\overline{x}(\tau_i), \lambda(\tau_i)) & H_\lambda(\overline{x}(\tau), \lambda(\tau_i)) \end{pmatrix}}_{\mathcal{D}H(\overline{x}(\tau_i), \lambda(\tau_i))} \cdot \underbrace{\begin{pmatrix} \overline{x}'(\tau_i) \\ \lambda'(\tau_i) \end{pmatrix}}_{t(\tau)} = 0$$

A suitable candidate for this parametrization is the arc-length parametrization defined as the inverse of

$$\tau(\lambda) = \int_{\lambda_0}^{\lambda} \|\overline{x}'(s)\| \, ds$$

[1] `Dual` is a Julia implementation of Clifford's dual numbers for automatic forward-mode differentiation

which satisfies $\|t(\tau)\| = 1$ for all τ. Computing this t is easy in the finite dimensional case but is less trivial for infinite matrices. Birkisson in [10, p. 181] proposed solving the system

$$\left(\begin{array}{c|c} \lambda'_{k-1} & \overline{x}'_{k-1} \\ \hline H_\lambda & H_x \end{array}\right) t_k = \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}$$

for an approximation of t_k by replacing the requirement $\|t_k\| = 1$ with $\langle t_{k-1}, t_k \rangle = 1$. However, *ApproxFun.jl* is not designed to accommodate this matrix structure out of the box such that an implementation would likely require more low level modifications to the package. In practice, turning points rarely interfere with the tangent calculation due to their usually sparse placement on the continuation path. Therefore, we use the standard parametrization in our implementation. The tangent predictor is implemented in Algorithm 2.4.

Algorithm 2.4 Tangential Predictor

$\lambda_i \leftarrow$ current parameter
$x_i \leftarrow$ current solution
$t_{i-1} \leftarrow$ previous tangent
Compute partial Fréchet derivative $H_x(x_i, \lambda_i)$ (`DualFun`)
Compute partial Fréchet derivative $H_\lambda(x_i, \lambda)$ (`Dual`)
Solve $H_x(x_i, \lambda_i)\overline{x}(\lambda_i) = -F(x_i, \lambda_i)$ for $\overline{x}(\lambda_i)$
Normalize $t_i = \frac{\overline{x}(\lambda_i)}{\|\overline{x}(\lambda_i)\|}$
if $\langle t_i, t_{i-1} \rangle < 0$ **then**
    $t_i \leftarrow -t_i$
end if
return t_i

After computing the tangent vector t which gives a locally optimal linear approximation of the continuation path, a stepsize must be chosen such that the predicted point is still within the local domain of convergence of the Newton method while making as much progress along the path as possible. The natural monotonicity test provides a convenient tool for this task. We recall the monotonicity bound

$$\frac{\overline{\Delta x}_{k+1}}{\Delta x_k} \leq \frac{1}{4}$$

from Table 1 which gives a strong indicator that x_k is within the domain of quadratic convergence of the Newton method. We use this condition to choose a stepsize σ which stays within the domain of quadratic convergence by only accepting stepsizes for which the Newton steps in the corrector satisfy the monotonicity bound. The standard method of obtaining such a σ is to start from a sufficiently large initial guess and simply reduce the stepsize by a constant factor until the condition is satisfied. This method works well in practice, however, using Myovskikh's local convergence result in combination with the approximation order of the continuation, it is possible to derive an even more effective stepsize selector. Given a function $H(x, \lambda)$ which satisfies the special case affine covariant Lipschitz condition

$$\left\| H_x(x, \lambda)^{-1} (H_x(x + \sigma v, \lambda) - H_x(x, \lambda)) v \right\| \leq \sigma\omega \|v\|^2$$

and a continuation method of order p, the Newton corrector is guaranteed to converge quadratically towards the solution path for

$$\|\hat{x}(\lambda) - \overline{x}(\lambda)\| \leq \eta\sigma^p < \frac{2}{\omega}$$

which is equivalent to

$$\sigma < \left(\frac{2}{\omega\eta}\right)^{1/p} \tag{2.19}$$

This is a simple consequence of (2.4) in Mysovskikh's Theorem 2.3. In 1979, Deuflhard used a stepsize bound similar to (2.19) to develop an adaptive stepsize control which, similar to his damped Newton algorithm, predicts the constant ω and additionally η to use the bound

directly as an estimator. Deuflhard used the slightly different affine covariant Lipschitz condition

$$\left\|H_x(x,\lambda)^{-1}(H_x(y,\lambda)-H_x(x,\lambda))\right\| \leq \hat{\omega}_0 \left\|y-x\right\| \quad \text{for all } x,y \in U$$

from the Newton-Kantorovich local convergence Theorem [4, Thm. 2.1] which also holds for Banach spaces and provides a slightly more local Lipschitz constant ω_0 to obtain a slightly better stepsize bound. The stepsize bound analogous to (2.19) under this Lipschitz condition is given by

$$\sigma < \left(\frac{\sqrt{2}-1}{\hat{\omega}_0 \eta}\right)^{1/p} \tag{2.20}$$

Using the bound

$$\Theta_0 \leq \frac{1}{2}\hat{\omega}_0 \eta \sigma^p (1+\frac{1}{2}\hat{\omega}_0 \eta \sigma^p)$$

which is a combined result of the derivation of the stepsize bound (2.20) and the local convergence analysis for simplified Newton methods, Deuflhard developed the following predictor-corrector scheme.

The corrector starts with an initial estimate for the maximally allowed stepsize $\sigma_{\max}$

$$\sigma_1 := [\sigma_{\max}] := \left(\frac{\sqrt{2}-1}{[\hat{\omega}_0\eta]}\right)^{1/p}, \quad [\hat{\omega}_0\eta] := \frac{\sqrt{1+4\Theta_0(\sigma_0)}-1}{\sigma_0^p} \leq \hat{\omega}_0\eta \tag{2.21}$$

This was directly derived from (2.4). Further corrections are computed by recursively applying (2.21) to the previously obtained stepsize σ_k

$$\sigma_{k+1} := \left(\frac{\sqrt{2}-1}{\sqrt{1+4\Theta_0(\sigma_k)}-1}\right)^{1/p} \sigma_k \tag{2.22}$$

The predictor is derived from the bound $\Theta_0 \leq \frac{1}{2}\hat{\omega}_0 \left\|\overline{\Delta x}_0(\sigma)\right\|$ and estimates the constants $\hat{\omega}_0$ and η with

$$[\hat{\omega}_0] := \frac{2\Theta_0(\sigma)}{\left\|\overline{\Delta x}_0(\sigma)\right\|} \leq \hat{\omega}_0, \quad [\eta] := \frac{\left\|\hat{x}(\lambda)-\overline{x}(\lambda)\right\|}{|\sigma|^p} \leq \eta$$

Here, σ is the final stepsize from the previous continuation step. Plugging both estimators into the bound for the maximal stepsize gives

$$\sigma_0 := \left(\frac{\left\|\overline{\Delta x}_0(\sigma)\right\|}{\left\|\hat{x}(\lambda)-\overline{x}(\lambda)\right\|}\frac{\sqrt{2}-1}{2\Theta_0(\sigma)}\right)^{1/p} \sigma \tag{2.23}$$

Deuflhard notes that in the case where $H(x(\sigma),\lambda(\sigma))$ is nearly linear, Θ_0 will be close to 0 causing the predictor to produce unrealistically large stepsizes. As a countermeasure, σ should be bounded by a maximally acceptable stepsize. Algorithm 2.5 implements the described stepsize control scheme in the context of a continuation method for the tangential predictor which has order $p = 2$.

Algorithm 2.5 Continuation Method

$\sigma_{\min} \leftarrow$ minimum stepsize
$\sigma_{\max} \leftarrow$ maximum stepsize
σ_0 initial stepsize
$k_{\max} \leftarrow$ maximum number of iterations
$\lambda^* \leftarrow$ target parameter
$\lambda_0 \leftarrow$ starting parameter
$x_0 \leftarrow$ starting solution
$y_0 \leftarrow (x_0, \lambda_0)$
for $k = 0, \ldots, k_{\max}$ **do**
  $t_k = (\Delta x_k, \Delta\lambda_k) \leftarrow$ normalized tangent vector by Algorithm 2.4
  if $k \geq 1$ **then**
    $\sigma_k^0 \leftarrow \max\left\{\sigma_{\min}, \min\left\{\sigma_{\max}, \left(\frac{\delta_{k-1}}{\varepsilon_{k-1}}\frac{\sqrt{2}-1}{2(\Theta_0)_{k-1}}\right)^{1/2}\sigma_{k-1}\right\}\right\}$ ▷ σ Predictor
  end if
  $j \leftarrow 0$

while true **do**
  if $\Delta\lambda_k > 0$ **then**
    $\sigma_k^j \leftarrow \min\left\{\sigma_k, \frac{\lambda^* - \lambda_k}{\Delta\lambda_k}\right\}$
  end if
  $\hat{y}_k^j \leftarrow y_k + \sigma_k^j t_k$
  Path Correction
    Solve $H(x_k^j, \hat{\lambda}_k^j) = 0$ for x_k^j with Alg. 2.2 for $x_0 = \hat{x}_k^j$, $\overline{\Theta} = 1/4$
    `conv` $\leftarrow$ convergence flag
    $(\Theta_0)_k^j \leftarrow$ natural monotonicity quotient of first Newton step for $\overline{\Delta x}_k^j$
    $\delta_k^j \leftarrow \left\|\overline{\Delta x}_0\right\|$ first Newton step for $\hat{x}_k^j$
    $\varepsilon_k^j \leftarrow \left\|\hat{x}_k^j - x_k^j\right\|$
    $y_k^j \leftarrow (x_k^j, \hat{\lambda}_k^j)$
  end
  if `conv` **then**
    $\sigma_k \leftarrow \sigma_k^j$
    $(\Theta_0)_k \leftarrow (\Theta_0)_k^j$
    $\delta_k \leftarrow \delta_k^j$
    $\varepsilon_k \leftarrow \varepsilon_k^j$
    $y_{k+1} \leftarrow \hat{y}_k^j$
    $k \leftarrow k + 1$
    break
  else if $\sigma_k^j = \sigma_{\min}$ **then**
    return
  else
    $\sigma_k^{j+1} \leftarrow \max\left\{\sigma_{\min}, \min\left\{\sigma_{\max}, \left(\frac{\sqrt{2}-1}{\sqrt{1+4(\Theta_0)_k^j}-1}\right)^{1/2}\sigma_k^j\right\}\right\}$ ▷ σ Corrector
    $j \leftarrow j + 1$
  end if
end while
if $\lambda_k = \lambda^*$ **then**
  return x_k
end if
end for

During the implementation, Algorithm 2.5 initially consistently chose stepsizes that were too large and required many correction steps to find a suitable stepsize. This turned out to be caused by the fact that the calculated tangent is not precise enough for the predictor to behave as a full second order predictor. Therefore, the implementation in *ApproxFunNewton.jl* uses the more conservative choice of $p = 1$. Furthermore, while the stepsize predictor often produces exceptionally good stepsize estimates already at the first attempt, it will also choose aggressively large stepsizes from time to time which may land on iterates where the computation of Newton steps is disproportionately expensive. This is a phenomenon caused by the underlying mechanism of how operators are evaluated in *ApproxFun.jl* which is discussed in more detail in chapter 3. For this reason, *ApproxFunNewton.jl* also implements the more inert standard stepsize predictor (reduce stepsize by constant factor) which generally requires more continuation steps but was in our experiments also less susceptible to overestimating the stepsize by a large margin.

Remark. One should note that Algorithm 2.5 in its given form only corrects the predictor for a fixed parameter and is therefore not able to handle turning points of the continuation path.

CHAPTER 3

Examples

We now want to test the Algorithms developed in chapter 2 on a variety of problems to demonstrate their effectiveness and applicability. All of the examples below have been implemented in Julia 1.9.2 with *ApproxFun.jl* [3] and the here developed *ApproxFunNewton.jl* [13] package. The benchmarks were completed on an M1 Pro Apple Silicon Chip with 16 GB of memory. The full code for the experiments is available at `https://github.com/LucasAschenbach/Bachelor-Thesis-MA6012-Code`.

3.1. Fluid Injection

The Fluid Injection Problem models the fluid mechanical potentials of a fluid which is pressed into a thin tube. The problem is given by

$$\begin{aligned} f''' - R(f'^2 - f * f'') + RA &= 0 \\ h'' + R * f * h' + 1 &= 0 \\ \theta'' + 0.7 * R * f * \theta' &= 0 \\ A' &= 0 \end{aligned}, \qquad \begin{aligned} f(0) = f'(0) = 0, &\quad f(1) = 1, f'(1) = 0 \\ h(0) = 0, &\quad h(1) = 0 \\ \theta(0) = 0, &\quad \theta(1) = 1 \end{aligned}$$

where R is the Reynolds number giving the ratio of inertial to viscous forces to predict the transition between laminar and turbulent flow. This problem gets increasingly difficult for larger Reynolds numbers due to the growing boundary layer at $x = 0$ and is known to be on the more difficult end of the spectrum of boundary value problems. When solving the problem with a shooting method for a Reynolds number of $R = 1,000$, a continuation method is even required.

Using the least imaginative starting values

$$f_0(x) = -2x^3 + 3x^2, \quad h_0(x) = 0, \quad \theta_0 = -(x-1)^2 + 1, \quad A_0 = 0$$

which are the lowest order polynomials to satisfy the boundary conditions, the problem can be solved for $R = 1,000$ with the standard Newton Method from Algorithm 2.2 in just 7 iterations, however, requiring 23.2 seconds and 1.929 GiB of memory allocations in total. Even for $R = 10,000$ the algorithm terminates already after 8 iterations for the same starting values. However, this also increases the solution time further to 141.5 seconds and the allocations to 6.333 GiB.

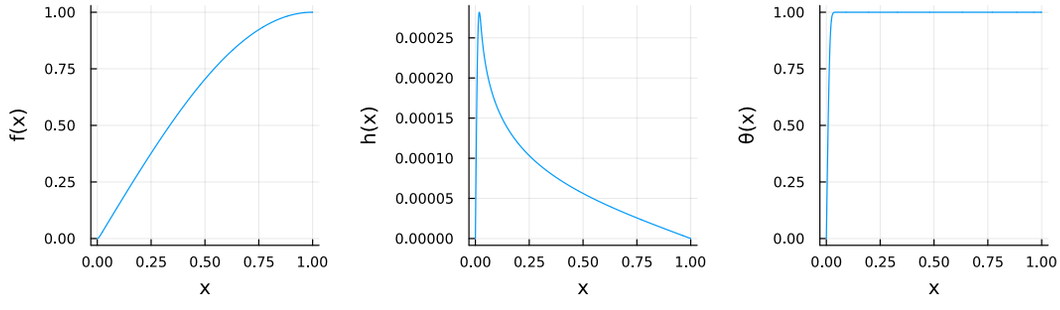


FIGURE 1. Solution of the Fluid Injection Problem for $R = 10,000$, determining the tubal cross section area to be $A = 2.49325$

It is somewhat remarkable that the algorithm converges this reliably for this problem. While the solution times achieved are by no means on the low end for this problem, the example demonstrates the robustness of the method, even for sensitive problems.

3.2. Nerve Pulse

The voltage and membrane permeability of a nerve cell during activation are well described as the solution to an ordinary boundary value problem. The differential equation was established by FitzHugh & Nagumo in 1961 and is given by

$$\begin{aligned} v' &= 3(v + p - v^3/3 - 1.3) \\ p' &= -(v - 0.7 + 0.8p)/3 \end{aligned}, \qquad \begin{aligned} v(0) &= v(T) = 0, \\ p(0) &= p(T) \end{aligned}$$

where T is an unknown period length and v and p are periodic functions representing the voltage and permeability of the nerve membrane, respectively. To solve this problem, the domain is normalized to $[0, 1]$ and T introduced as a constant function in the differential equation.

$$\begin{aligned} v' &= 3T(v + p - v^3/3 - 1.3) \\ p' &= -T(v - 0.7 + 0.8p)/3 \\ T' &= 0 \end{aligned}, \qquad \begin{aligned} v(0) &= v(1) = 0, \\ p(0) &= p(1) \end{aligned}$$

When applying Algorithm 2.2 to this problem with the starting values

$$v_0(x) = 2\sin(2\pi x), \quad p_0(x) = 1 + \cos(2\pi x), \quad T_0(x) = 2\pi$$

the algorithm reaches the solution with 11 iterations. However, each iteration takes a fairly long time to complete, even with a disabled monotonicity test, bringing the total solution time to 891.5 seconds and the memory allocations to 21.793 GiB.

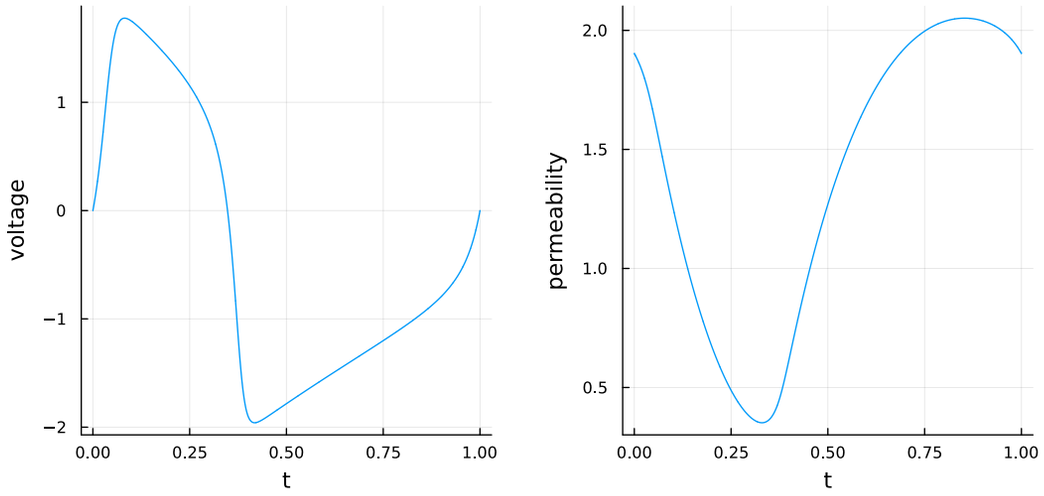


FIGURE 2. Solution of the Nerve Pulse Problem, $T = 10.71081$

An interesting phenomenon to observe in this example is that the algorithm becomes more than 1.5 times as expensive with 1,642.9 seconds and 34.612 GiB of allocations when also performing the natural monotonicity test in each iteration. This is different from the finite dimensional case where the monotonicity test is a cheap operation relative to the newton update. As already mentioned in the previous chapter, this effect is caused by additional post correction steps from Algorithm 1.1 for the changed right hand side in the simplified Newton step.

3.3. Carrier Problem

As an example of an ODE where the damped Newton method from Algorithm 2.3 outperforms the standard Newton method 2.2, consider this modified version of the Carrier Problem

$$0.001u'' + 2(1 - x^2)u + u^2 = 1, \quad u(-1) = u(1) = 0$$

with the starting value

$$u_0(x) = 2(x^2 - 1)\left(1 - \frac{2}{1 + 20x^2}\right)$$

The damped Newton method requires 24 iterations and 131.0 seconds to solve the problem with a tolerance of 10^{-14}. The standard Newton method on the other hand shows no sign of convergence, even after 835.9 seconds and 15 iterations.

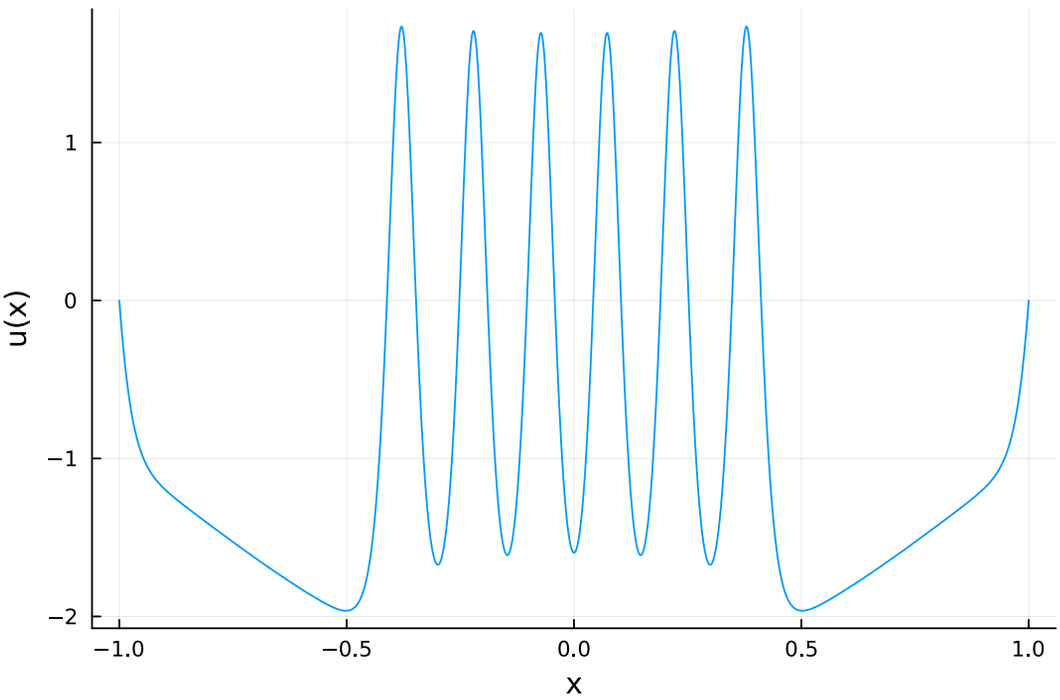


FIGURE 3. Solution of the modified Carrier Problem, using the damped Newton method

3.4. Lubrication

The boundary value problem

$$\epsilon y'(x) = \sin(x)^2 - \lambda \frac{\sin(x)}{y(x)}, \quad y(-\pi/2) = y(\pi/2) = 1$$

models the lubrication of a ball bearing where λ is the unknown injection rate. This problem becomes increasingly difficult for smaller values of ϵ and requires a continuation method to solve. Using the starting value

$$y_0(x) = 1, \quad \lambda_0(x) = 1$$

and starting parameter $\epsilon_0 = 1$, the problem can be solved for $\epsilon = 0.1$ with just 6 continuation steps in 135.8 seconds using the standard stepsize predictor for the continuation.

This is also one example where the more optimistic predictor from Algorithm 2.5 overestimates the stepsize and lands on a point which terminates in a memory overflow.

3.5. Apollo 11 Reentry

As a final example, we consider the Apollo 11 reentry control problem which reveals some limitations of the current implementation of the Ultraspherical Spectral Method for non-linear problems. The goal of the reentry problem is to find a control function $u(t)$ for the angle of a capsule reentering the atmosphere such that the capsule moves along a trajectory which causes the least amount of heat to be generated on the surface of the capsule. Any valid flight path must satisfy the differential equations

$$\begin{aligned} v' &= -\frac{S\rho v^2}{2m} C_D(u) - \frac{g \sin\gamma}{(1+\xi)^2} \\ \gamma' &= \frac{S\rho v}{2m} C_L(u) + \frac{v\cos\gamma}{R(1+\xi)} - \frac{g\cos\gamma}{v(1+\xi)^2} \\ \xi' &= \frac{v\sin\gamma}{R} \end{aligned}$$

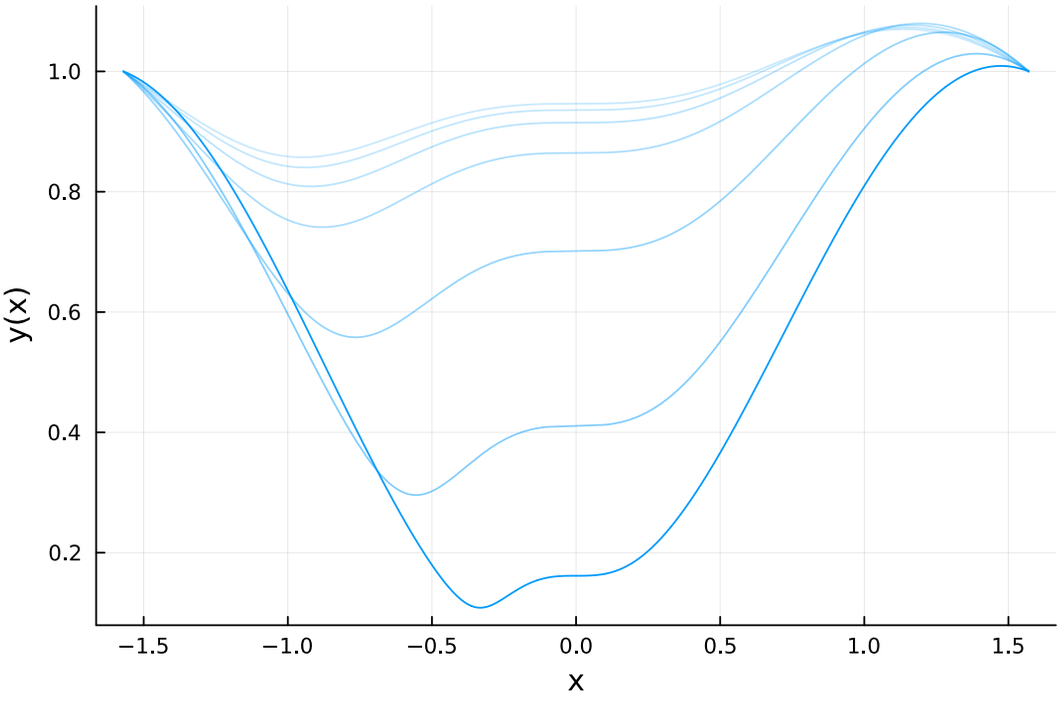


FIGURE 4. Solution of the Lubrication Problem for $\epsilon = 0.1$, determining the injection rate to be $\lambda = 1.01866$

where v is the velocity, γ the flight-path angle relative to the surface of the earth and ξ the altitude measured from the earth's surface normalized by the radius of the earth. From the Pontryagin minimum principle follows that the optimal control u must minimize the Hamiltonian function

$$H := 10v^3\sqrt{\rho} + \lambda_v v' + \lambda_\gamma \gamma' + \lambda_\xi \xi'$$

at every point. This leads to the additional differential equations

$$\lambda_v' = -\frac{\partial H}{\partial v}$$
$$\lambda_\gamma' = -\frac{\partial H}{\partial \gamma}$$
$$\lambda_\xi' = -\frac{\partial H}{\partial \xi}$$

Solving $\frac{\partial H}{\partial u} = 0$ for u gives the optimal control

$$u = \arctan\left(\frac{c_3\lambda_\gamma}{c_2\lambda_v v}\right)$$

This problem is notoriously sensitive to small perturbations in the starting values such that as a first step, the problem is reduced to an auxiliary problem which produces an only near optimal but valid trajectory as starting value for the actual problem. This auxiliary problem is given by the differential equations for the trajectory, the unknown time to arrival, and a parametrized control function with two parameters p_1, p_2

$$\hat{u}(t) = p_1 \arctan(\alpha(p_2 - t))$$

to make the problem uniquely solvable.

Any of our solution attempts with the Ultrapsherical Spectral Method already failed at the auxiliary problem. The reason for this is that the differential operator, even for the auxiliary problem, contains too many operations which are difficult to compute on the ultraspherical coefficient space. This makes the evaluation of the operator in *ApproxFun.jl* very expensive and memory hungry. For example, the division of two functions, while easy to compute on the value space, on the ultrapsherical coefficient space involves either the solution of an infinite linear system with a multiplication operator or the conversion to and from the value space for each evaluation. As these operations accumulate and as the number

of coefficients for the solution in the current iteration grows, each F evaluation becomes more expensive which is also reflected in the methods memory footprint. In our test the differential operator failed to complete even a single F evaluation at the true solution for the auxiliary problem, which was obtained through a multiple shooting method, without causing a memory overflow. This was again tested after removing all Chebyshev coefficients below machine precision but to no avail. Since the evaluation let alone automatic differentiation of the operator is computationally infeasible near the solution, computing a newton update anywhere near the solution is out of the question.

This is an important limitation of this implementation for evaluating operators which was already hinted at by the slow iterations and large memory allocations produced in the other examples. Generally, the observation from this and the previous examples is that the Algorithms presented in chapter 2 tend to reach the solution with a small number of iterations, however, for a moderately complicated operator, these iterations can be very slow and may even become computationally infeasible on consumer hardware as the number of coefficients of the solution function grows.

Bibliography

[1] Sheehan Olver and Alex Townsend. *A practical framework for infinite-dimensional linear algebra*. 2014. arXiv: 1409.5529 [math.NA].

[2] Sheehan Olver and Alex Townsend. *A fast and well-conditioned spectral method*. 2012. arXiv: 1202.1347 [math.NA].

[3] Sheehan Olver and Alex Townsend. *ApproxFun.jl*. 2014. URL: https://github.com/JuliaApproximation/ApproxFun.jl.

[4] Peter Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics. Springer Science & Business Media, 2011. ISBN: 978-3-642-23898-7. DOI: 10.1007/978-3-642-23899-4.

[5] Frank W. J. Olver. "Numerical solution of second-order linear difference equations". In: *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics* (1967), p. 111. URL: https://api.semanticscholar.org/CorpusID:54911876.

[6] L. N. Trefethen. "Householder triangularization of a quasimatrix". In: *IMA Journal of Numerical Analysis* 30.4 (2009), pp. 887–897. DOI: 10.1093/imanum/drp018. URL: http://www.chebfun.org/publications/trefethen_householder.pdf.

[7] Sheehan Olver and Alex Townsend. *InfiniteLinearAlgebra.jl*. 2019. URL: https://github.com/JuliaLinearAlgebra/InfiniteLinearAlgebra.jl.

[8] A. Fletcher. "British Association Mathematical Tables, Vol. X. Bessel Functions, Part II : Functions of Positive Integer Order. Prepared by W G. Bickley L.J. Comrie J.C.P. Miller D.H. Sadler and A.J. Thompson Pp. xl, 255. 60s. 1952. (Published for the Royal Society at the University Press, Cambridge)". In: *The Mathematical Gazette* 37.322 (1953), pp. 307–308. DOI: 10.2307/3610090.

[9] Peter Deuflhard and Florian A. Potra. "Asymptotic Mesh Independence of Newton-Galerkin Methods via a Refined Mysovskii Theorem". In: *SIAM Journal on Numerical Analysis* 29.5 (1992), pp. 1395–1412. ISSN: 00361429. URL: http://www.jstor.org/stable/2158048.

[10] Ásgeir Birkisson. "Numerical solution of nonlinear boundary value problems for ordinary differential equations in the continuous framework". PhD thesis. Oxford University, UK, 2013.

[11] John K Hunter and Bruno Nachtergaele. *Applied Analysis*. WORLD SCIENTIFIC, 2001. DOI: 10.1142/4319. eprint: https://www.worldscientific.com/doi/pdf/10.1142/4319. URL: https://www.worldscientific.com/doi/abs/10.1142/4319.

[12] Peter Deuflhard. "A stepsize control for continuation methods and its special application to multiple shooting techniques". In: *Numerische Mathematik* 33.2 (1979), pp. 115–146. DOI: 10.1007/BF01399549. URL: https://doi.org/10.1007/BF01399549.

[13] Lucas Aschenbach. *ApproxFunNewton.jl*. 2023. URL: https://github.com/LucasAschenbach/ApproxFunNewton.jl.