DISSERTATION

ANALYSIS AND APPLICATION OF THE NONLINEAR POWER METHOD

Submitted by

Sean Eastman

Department of Mathematics

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2005

COLORADO STATE UNIVERSITY

September 6, 2005

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY SEAN EASTMAN ENTITLED "ANALYSIS AND APPLICATION OF THE NONLINEAR POWER METHOD" BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Dr. Eugene Allgower

Dr. Paul DuChateau

Dr. Paul Heyliger

Adviser: Dr. Donald Estep

Department Head: Dr. Simon Tavener

ABSTRACT OF DISSERTATION

ANALYSIS AND APPLICATION OF THE NONLINEAR POWER METHOD

The power method is a standard technique in numerical linear algebra for approximating the dominant eigenpair of a square matrix. Algorithms for implementing the method are simple and efficient, since only knowledge of the matrix action is required. In this thesis, we develop a nonlinear power method (NLPM) for computing the dominant eigenpair of the linearization of a nonlinear map. We implement the method in a matrix–free way by utilizing function differences, and also by reconstructing the entire matrix when matrix–free methods are unavailable. We apply the NLPM to the problems of determining the stability of a system of differential equations, and also to examining the issue of linearization error for computational error estimates for nonlinear elliptic two–point boundary value problems. We examine properties of the map between the forward solution and the linearized dual solution and use the NLPM in various forms to determine the norm of the derivative of this map, hence bounding the effect of linearization. Several numerical examples are presented in Chapter 5.

Sean Eastman
Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523
Fall 2005

ACKNOWLEDGEMENTS

Thanks are due first and foremost to my adviser, Professor Donald Estep. His guidance and support are the driving force behind this work. I would also like to thank Professors Eugene Allgower and Paul DuChateau, both of whom have made a great impact on my development as a student, and as a person. Further thanks are due to Professor G.W. Stewart at the University of Maryland, and Professor Luca Dieci at Georgia Tech. Their insights and suggestions helped me a great deal. Finally I thank Professor Roger Thelwell, for the many comments and corrections he made to this thesis, and more importantly for our friendship.

The research activities of Sean Eastman are partially supported by the National Science Foundation, through grants DMS 0107832.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Overview

The power method is a standard technique in numerical linear algebra for approximating the dominant eigenpair of a square matrix. Algorithms for implementing the method are simple and efficient, since only knowledge of the matrix action is required. In this thesis, we develop a nonlinear power method (NLPM) for computing the dominant eigenpair of the linearization of a nonlinear map. We implement the method in a matrix–free way by utilizing function differences, and also by reconstructing the entire matrix when matrix–free methods are unavailable. We apply the NLPM to the problems of determining the stability of a system of differential equations, and also to examining the issue of linearization error for computational error estimates for nonlinear elliptic two–point boundary value problems. We examine properties of the map between the forward solution and the linearized dual solution and use the NLPM in various forms to determine the norm of the derivative of this map, hence bounding the effect of linearization. Several numerical examples are presented in Chapter 5.

1.2 Motivating example: linearization and *a-posteriori* error estimation

Consider a system $Ax = b$ of linear equations, which is solved in some way to obtain an approximate solution X. The error between the true and approximate solutions is $e = x - X$. Generally this can't be computed, but we can compute the residual $R = AX - b$, which measures how well the approximate solution solves the equation. We introduce a *dual problem* to find a relationship between the true (but unknown) error e and the computable residual R. The dual problem is

$$A^\top \phi = \psi, \tag{1.2.1}$$

where ψ is a unit vector and $A^\top$ the transpose of A (more generally, the adjoint operator). We can get a representation of a projection of the error by computing the inner product

$$|\langle e, \psi \rangle| = |\langle e, A^\top \phi \rangle| = |\langle Ae, \phi \rangle| = |\langle R, \phi \rangle|. \tag{1.2.2}$$

This gives the size of a projection of the error in a specified direction in terms of the inner product of the computable residual and the solution of the dual problem corresponding to the choice of direction. Using the Cauchy–Schwartz inequality,

$$|\langle e, \psi \rangle| \leq \|\phi\| \, \|R\|. \tag{1.2.3}$$

$\|\phi\|$ and $\|R\|$ are referred to as the strong stability factor and strong residual, respectively. In case the problem is nonlinear, we have $f(x) = b$, and the residual error is $R = f(X) - b$, so $f(x) - f(X) = -R$. We use the fundamental theorem of calculus to obtain a linear equation for the error:

$$f(x) - f(X) = \int_0^1 f'(sx + (1-s)X)(x - X)\, ds. \tag{1.2.4}$$

Now, the linear equation for the error is $Ae = -R$, where

$$A = \int_0^1 f'(sx + (1-s)X)\, ds. \tag{1.2.5}$$

This is the linear problem obtained by linearizing f around an average of x and X. The variational analysis for obtaining an estimate on a projection of the error in terms of a stability factor and residual is the same as for the linear problem outlined above. The issue is that the operator in (1.2.5) depends on the unknown true solution x. In practice, this is sidestepped by taking $A = f'(X)$, i.e. we linearize around the computed approximate solution. Linearizing around the approximate solution may produce inaccurate error estimates if nearby solutions have much different stability properties. Understanding the effect of this linearization on the error estimates was the genesis for this project.

If we consider the stability factors as nonlinear functionals of the computed solution, this suggests examining the derivatives of these functionals in order to determine the sensitivity to changes in the input function around which linearization is performed. Unfortunately, the dimension of the input is too large to make this practical. On the other hand, we only really desire information about the largest possible change in the functionals. To this end, we consider the map from the space of the approximate solutions for the forward problem to the space of approximate solutions of the dual problem obtained by linearizing around the approximate solution of the forward problem and then solving by some numerical method.

Suppose that $U \in V$ is a typical input into a nonlinear function F on the space of functions V. Assuming that δ is a small perturbation in V, we write the integral mean value theorem as

$$A_\delta \delta = \int_0^1 F'(U + s\delta)\, ds\, \delta = F(U+\delta) - F(U), \tag{1.2.6}$$

Where F' is the Gatêaux derivative of F. This indicates that we can evaluate $A_\delta \delta$ by computing

$$A_\delta \delta = F(U + \delta) - F(U). \tag{1.2.7}$$

If A_δ were constant, then ideally we would like to compute the largest singular value and the associated left singular vector. This would give the perturbation to U that produces the largest possible change in F. One way to compute this is an implementation of the power method that alternately computes the action of A and $A^\top$ on vectors. In the special case that A is normal ($A^\top A = AA^\top$, where we use the transpose instead of the Hermitian adjoint A^* since we will always assume real entries in A) the dominant singular value and dominant eigenvalue coincide, and we can just use the power method. When A_δ is not constant, we can try to apply the same idea using the differencing to formulate the *Nonlinear Power Method*

- Choose $\delta_0 \in V$
- For $k = 1, 2, \ldots$
  - Set $\tilde{\delta_k} = F(U + \delta_{k-1}) - F(U)$
  - Set $\delta_k = \dfrac{\alpha_k \tilde{\delta_k}}{\|\delta_k\|}$

In this situation, in which we are linearizing, we want the iterates to have small norm and even have norms that tend to zero. We accomplish this using the sequence $\{\alpha_k\}$. This process converges to the dominant eigenvalue/eigenvector pair for the linearization of the map F at the point U. If the linearization is normal, the dominant eigenvalue gives the norm, and we can use this quantity to bound the effect of linearization on certain *a-posteriori* error estimates. For nonnormal maps, we can still estimate the

norm by reconstructing a matrix representation for the linearization relative to some basis.

Chapter 2

THE POWER METHOD

In this chapter, we describe the well–known *power method*, an iterative process for estimating the dominant eigenvector and eigenvalue of a matrix. The power method is related to the QR algorithm, which is a more general iterative procedure for finding all of the eigenvalues of a matrix. It is based on the QR decomposition, which is in turn related to the *Gram–Schmidt* process [109]. Since the focus in this thesis is on dominant eigenpairs only, we have no need for the generality (and computational expense) of the QR algorithm. Theory and implementation for QR methods can be found in many books on numerical linear algebra, for instance [6, 22, 56, 60, 65, 97, 109]. We also examine two common extensions of the power method: *spectral enhancement* and *inverse iteration*.

We generalize the power method to matrices that are subject to perturbations at each stage of the iteration. Traditionally, analysis has been done in this regard with a view to analyzing the effect of finite precision calculations on the power method. The reason for the interest in perturbed matrices in this thesis is completely different, namely we want to determine the dominant eigenpair of the linearization of a nonlinear map. To do this, we begin with a nonlinear vector function and emulate the action of the Jacobian matrix by evaluating function differences. Since we are interested

in the linearization of a nonlinear function at a point, the theoretical interest is in the limit as the size of the perturbations tends to zero. This allows a generalization of the power method to what we call the *nonlinear power method* (NLPM), in which we recover the dominant eigenpair to the linearization by computing function evaluations. As the power method can be implemented with only knowledge of the matrix action, the function differencing in the nonlinear power method serves as a matrix-free method for recovering the relevant information about the Jacobian. This is accomplished by computing a sequence of approximate Gâteaux derivatives, where the i^{th} linearization occurs in the direction of the $(i-1)^{st}$ iterate.

We further generalize the power and nonlinear power methods to infinite dimensions. We do so in order to analyze problems involving operators on function spaces and their Fréchet derivatives. We end the chapter with an indication of exactly how the higher order terms in the linearization can be viewed as a perturbations to the derivative, providing a link between the perturbed power method and the nonlinear power method.

2.1 Finite dimensions

The power method is a classic algorithm for approximating the dominant eigenvalue/eigenvector pair of a matrix. Its use dates back to at least 1913 (see [97], and the references therein) although Bodewig [18] credits its discovery to von Mises in 1929 [108], and comprehensive analysis to Aitken in 1937 [3]. Beginning with any initial vector, a sequence of vectors (sometimes called a *Krylov sequence*, see [65]) is generated by repeated application of the matrix. The component in the direction of the dominant eigenvector grows most quickly, so the sequence of iterates tend in

direction to the dominant eigenvector. The simplest case occurs when the matrix has a single eigenvalue of largest magnitude, and also has a full set of linearly independent eigenvectors. This case is treated in many standard texts [6, 56, 65, 109]. More general treatments are given in [82, 97, 111] that cover the cases of repeated eigenvalues and defective matrices. We review the approach in [97] in §2.1.2.

2.1.1 Nondefective matrices

In [109], Watkins defines an $n \times n$ matrix A to be *simple* if it has n linearly independent eigenvectors, then defines a *defective* matrix as a matrix that is not simple. In [97], Stewart defines a *defective eigenvalue* as an eigenvalue whose algebraic multiplicity exceeds its geometric multiplicity, and he calls a matrix defective if it has one or more defective eigenvalues. Clearly these definitions are the same, and the proof of the convergence of the power method is easiest to understand when the matrix in question is simple. Let A be an $n \times n$ matrix with a full set of linearly independent eigenvectors v_i (which we may assume to have unit norm) eigenvalues λ_i, and assume that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$. Let u_0 be any vector in $\mathbb{R}^n$, and write u_0 as a linear combination of the eigenvectors:

$$u_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n, \quad c_1 \neq 0. \tag{2.1.1}$$

We will see below why it is important that $c_1 \neq 0$. Now, the action of A in the eigendirections is just scalar multiplication by the corresponding eigenvalue, so for any $m \in \mathbb{N}$,

$$A^m u_0 = c_1 \lambda_1^m v_1 + c_2 \lambda_2^m v_2 + \cdots + c_n \lambda_n^m v_n. \tag{2.1.2}$$

Dividing by λ_1^m, we have

$$\frac{A^m u_0}{\lambda_1^m} = c_1 v_1 + \sum_{j=2}^{n} c_j \left|\frac{\lambda_j}{\lambda_1}\right|^m v_j. \tag{2.1.3}$$

Defining $q_m := \dfrac{A^m u_0}{\lambda_1^m}$, we have

$$\begin{aligned} \|q_m - c_1 v_1\| &= \left\| \sum_{j=2}^{n} c_j \left|\frac{\lambda_j}{\lambda_1}\right|^m v_j \right\| \\ &\leq C \left(\frac{|\lambda_2|}{|\lambda_1|}\right)^m; \qquad C := \sum_{j=2}^{n} |c_j|, \end{aligned} \tag{2.1.4}$$

since $|\lambda_2| \geq \lambda_j$ for $j = 3 \ldots n$, and $\|v_j\| = 1$ for $j = 1 \ldots n$. Now given any $\epsilon > 0$,

$$m > \frac{\ln \epsilon - \ln C}{\ln |\lambda_2| - \ln |\lambda_1|} \Rightarrow \|q_m - c_1 v_1\| < \epsilon,$$

thus $q_m \to c_1 v_1$ as $m \to \infty$. It is clear why we require the initial starting vector u_0 to have some nonzero component in the direction of v_1, since $c_1 = 0$ in (2.1.4) implies that the iteration converges to the zero vector. This is due to the fact that we normalized by dividing by λ_1; a different normalization scheme (division by the largest component, for instance) causes the iteration to converge to the eigenvector corresponding to λ_2. In [64], Hotelling put it this way: "The process fails only in the infinitely improbable case of the initial values being linearly dependent upon principal components other than the first; the greatest of these components is then approached."

The power method iterates converge linearly with convergence ratio $\left|\dfrac{\lambda_2}{\lambda_1}\right| < 1$, although convergence may be slower in some cases where $|\lambda_2| = |\lambda_3|$ [109]. In practice, the dominant eigenvalue is not known *a-priori* so the quantity q_m cannot be formed. However, some scheme for normalization must be employed, since the iterates $A^m u_0$ tend in norm either to 0 or ∞,

according to whether $|\lambda_1| < 1$ or $|\lambda_1| > 1$. Dividing each iterate by its norm assures a sequence of unit vectors whose directions tend toward that of v_1. The actual choice of norm is unimportant, and a simple, convenient approach is to divide each iterate by the magnitude of its largest component, producing a sequence whose limit is v_1, with $\|v_1\|_\infty = 1$. A simple algorithm for implementing the power method numerically is given by

Algorithm 2.1.1. *The Power Method*

1. *choose* u_0 *at random*

2. *for* $k = 1, 2, 3, \ldots$

3. $\quad u_k = Au_{k-1}$ *% generate term of Krylov sequence*

4. $\quad \lambda_k = \dfrac{\langle u_{k-1}, u_k \rangle}{\langle u_{k-1}, u_{k-1} \rangle}$ *% approximate eigenvalue*

5. $\quad u_k = \dfrac{u_k}{\|u_k\|}$ *% normalize to avoid over- or underflow*

Of course, we must specify convergence criteria, and there are ways to increase the efficiency and lessen the cost of the algorithm. The primary reason for including this algorithm here is to compare it with algorithms (2.4.1) and (2.4.2) for the nonlinear power method that we develop in §2.4.

2.1.2 Defective matrices

In §2.1.1, we assumed that there existed a set of n linearly independent eigenvectors for the matrix A. This admitted a relatively straightforward proof, but the power method still converges under the relaxed assumption that A just has a single dominant eigenpair (λ_1, v_1) [97]. The proof becomes more complicated due to the increase in complexity of the geometry of the

eigenspaces. Since we cannot write an arbitrary vector as a linear combination of eigenvectors, we must utilize different ideas. Namely, we need a way to express an arbitrary vector in terms of a component in the direction of the dominant eigenvector v_1 and a representation in all the "other directions." The fact that we can do this relies on the following theorem and corollary, which appear respectively as Theorem 1.19 and Corollary 1.20 on page 20 in [97]. These are consequences of a result on the block diagonalization of matrices resulting from the solution of Sylvester's equation [60, 97].

Theorem 2.1.1. *Let the spectrum $\Lambda(A)$ be the union $\bigcup_{i=1}^{k} \mathcal{L}_i$ of disjoint sets $\mathcal{L}_i$ (the sets $\mathcal{L}_i$ are actually multisets, in which elements may be repeated in order to account for eigenvalues with algebraic multiplicity greater than one). Then there is a nonsingular matrix X such that*

$$X^{-1}AX = \begin{bmatrix} L_1 & 0 & \cdots & 0 \\ 0 & L_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L_k \end{bmatrix} := \operatorname{diag}(L_1, \ldots, L_k), \tag{2.1.5}$$

where $\Lambda(L_i) = \mathcal{L}_i$.

Corollary 2.1.1. *Let the matrix A of order n have distinct eigenvalues $\lambda_1, \ldots, \lambda_n$ with multiplicities $m_1, \ldots, m_k$. Then there is a nonsingular matrix X such that*

$$X^{-1}AX = \operatorname{diag}(L_1, \ldots, L_k), \tag{2.1.6}$$

where L_i is of order m_i and has only the eigenvalue λ_i.

Thus, in the case of a single dominant eigenpair, we can find a nonsingular matrix $V = (v_1 \; V_2)$ such that

$$V^{-1}AV = \begin{bmatrix} \lambda_1 & 0 \\ 0 & M \end{bmatrix}. \tag{2.1.7}$$

Furthermore, we can assume that $\|v_1\| = 1$ and $V_2^\top V_2 = I$. Now, v_1 and the columns of V_2 form a basis for $\mathbb{C}^n$, so any vector u_0 may be written as $u_0 = c_1 v_1 + V_2 C_2$. With this notation we have the following theorem, which appears as Theorem 1.1 on page 57 in [97].

Theorem 2.1.2. *Let A have a unique dominant eigenpair (λ_1, v_1), and let A be decomposed in the form (2.1.7), where $V = (v_1 \; V_2)$ satisfies $\|v_1\| = 1$ and $V_2^\top V_2 = I$. Let u_0 be a nonzero starting vector, written as $u_0 = c_1 v_1 + V_2 C_2$. If $c_1 \neq 0$, then*

$$\sin \angle(v_1, A^k u_0) \leq \frac{|\lambda_1|^{-k} \|M^k\|_2 \|C_2/c_1\|_2}{1 - |\lambda_1|^{-k} \|M^k\|_2 \|C_2/c_1\|_2}, \tag{2.1.8}$$

where the angle between two vectors x and y is defined by

$$\angle(x, y) := \cos^{-1} \frac{\langle x, y \rangle}{\|x\| \|y\|}. \tag{2.1.9}$$

Note that the spectral radius $\rho(M) < |\lambda_1|$, so the error in the eigenvector approximation converges to zero at an asymptotic rate of $\left(\frac{\rho(M)}{|\lambda_1|}\right)^k$. However, it need not be the case that $\|M\|_2 < |\lambda_1|$, and in fact if A is highly nonnormal there may be a significant transient phase in which the sine of the angle between v_1 and $A^k u_0$ appears not to converge at all. This is due to the fact that, although the asymptotic behavior of A^k is governed by λ_1, the norm "wins out" temporarily. See [61, 97, 104] for more details.

2.1.3 The perturbed power method

Any numerical algorithm implemented on a computer must deal with the fact that roundoff errors have an effect on the computation. In terms of the power method, rather than starting with a vector u_0 and generating

a sequence of vectors via $u_{k+1} = \nu_k A u_k$, where ν_k is the normalizing factor, the actual computation can be described as [98]

$$u_{k+1} = \nu_k (A + E_k) u_k, \tag{2.1.10}$$

where $\|E_k\|/\|A_k\|$ is small, e.g. the order of the rounding unit [60, 96]. We call this the *perturbed power method* (PPM), since the matrix in question is subjected to a small perturbation by the matrix E_k at step k of the iteration. Some results on perturbed matrix powers are given in [80] and [61], and more recently in [98]. We are interested in a special case in which the sequence of perturbations $\{E_k\}$ tend to zero with increasing k. This is useful in the context of linearization for a nonlinear map, and is markedly different from assessing the effects of roundoff error, in which case all of the perturbations to A are roughly the same size and do not tend to zero. Stewart's paper [98] covers both situations, and we now turn to reviewing the results there that are needed later.

Let A be a matrix of order n with eigenvalues $\lambda_1, \ldots, \lambda_n$ ordered so that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, and let $\rho(A) = |\lambda_1|$ denote the spectral radius of A. We need the following two transformations:

1. For any $\eta > 0$, there is a matrix X such that

$$\|X^{-1} A X\| \leq \rho(A) + \eta. \tag{2.1.11}$$

The proof relies on the Schur triangularization theorem. See [97, 98] or [63]. The result is also true in infinite dimensions [34, 113]: given a bounded linear operator L on a Banach space X with norm $\|\cdot\|$, there exists an equivalent norm $|||\cdot|||$ such that $|||L||| \leq \rho(L) + \eta$.

2. Let λ_1 be a simple eigenvalue of A with right and left eigenvectors v_1 and w_1 normalized so that $\|v_1\| = \|w_1\| = 1$ and $\gamma = \langle v_1, w_1 \rangle > 0$. Let $\sigma = \sqrt{1-\gamma^2}$. Then there is a matrix U with condition number

$$\kappa(U) = \frac{1+\sigma}{\gamma} \tag{2.1.12}$$

in the 2–norm such that

$$U^{-1}AU = \begin{bmatrix} \lambda_1 & 0 \\ 0 & B \end{bmatrix}, \tag{2.1.13}$$

and the first column of U is the dominant eigenvector for A. The proof relies on the CS decomposition [56, 96, 98].

For the PPM, the computation proceeds as

$$u_{k+1} = \nu_k(A + E_k)u_k, \tag{2.1.14}$$

where $\{E_k\}$ are perturbations to the matrix A and ν_k is a normalizing factor. Invoking the two transformations above, we can transform A so that it has the form $\mathrm{diag}(1, B)$, where the spectral radius $\rho(B) < 1$, and we may also assume that $\|B\| \leq \beta < 1$. The vector u_k is normalized so that the first component is 1, and we write

$$u_k = \begin{bmatrix} 1 \\ h_k \end{bmatrix}. \tag{2.1.15}$$

Now, the $(k+1)^{st}$ iteration of the PPM is given by

$$u_{k+1} = (A + E_k)u_k = \begin{bmatrix} 1 + e_{11}^{(k)} & e_{12}^{(k)\top} \\ e_{21}^{(k)} & B + E_{22}^{(k)} \end{bmatrix} \begin{bmatrix} 1 \\ h_k \end{bmatrix}. \tag{2.1.16}$$

Let η_k be an upper bound for $\|h_k\|$. We derive an upper bound η_{k+1} for $\|h_{k+1}\|$ in the form of the quotient of a lower bound on the size of the first

component of $(A+E_k)u_k$ and an upper bound on the size of the rest of the vector. Denoting the components of u_{k+1} by γ_1 (a scalar) and γ_2 (a vector of length $n-1$), we have $\gamma_1 = 1 + e_{11}^{(k)} + \langle e_{12}^{(k)}, h_k \rangle$, so a lower bound on $|\gamma_1|$ is given by

$$\begin{aligned} 1 &= |1 + e_{11}^{(k)} - e_{11}^{(k)} + \langle e_{12}^{(k)}, h_k \rangle - \langle e_{12}^{(k)}, h_k \rangle| \\ &\leq |\gamma_1| + |e_{11}^{(k)}| + |\langle e_{12}^{(k)}, h_k \rangle| \\ &\leq |\gamma_1| + \epsilon_k + \epsilon_k \eta_k, \end{aligned} \tag{2.1.17}$$

so we have

$$1 - (1+\eta_k)\epsilon_k \leq |\gamma_1|. \tag{2.1.18}$$

An upper bound on the second component is given by

$$\begin{aligned} \|\gamma_2\| &= \|e_{21}^{(k)} + (B + E_{22}^{(k)})h_k\| \\ &\leq \|Bh_k\| + \|e_{21}^{(k)}\| + \|E_{22}^{(k)}h_k\| \\ &\leq \beta\eta_k + \epsilon_k(1+\eta_k), \end{aligned} \tag{2.1.19}$$

so a bound for $\|h_{k+1}\|$ is given by

$$\|h_{k+1}\| \leq \eta_{k+1} \equiv \frac{\beta\eta_k + \epsilon_k(1+\eta_k)}{1-(1+\eta_k)\epsilon_k}. \tag{2.1.20}$$

We define

$$\varphi_\epsilon(\eta) = \frac{\beta\eta + \epsilon(1+\eta)}{1-(1+\eta)\epsilon}, \tag{2.1.21}$$

so that $\eta_{k+1} = \varphi_{\epsilon_k}(\eta_k)$. With this notation, we have

Theorem 2.1.3. *For any $\eta_1 > 0$, there is an ϵ_1 such that if the sequence $\{\epsilon_k\}_{k=1}^{\infty}$ approaches zero monotonically, then the sequence defined by*

$$\eta_{k+1} = \varphi_{\epsilon_k}(\eta_k), \qquad k = 1, 2, 3, \ldots, \tag{2.1.22}$$

converges monotonically to zero.

Proof. Note that

$$\varphi_\epsilon'(\eta) = \frac{\beta}{1-(1+\eta)\epsilon} + \frac{\beta\eta + \epsilon(1+\eta)}{[1-(1+\eta)\epsilon]^2}\,\epsilon, \tag{2.1.23}$$

so, given $\eta_1 > 0$, a calculation indicates that taking

$$\epsilon_1 = \frac{3-\beta+2\,\eta_1 - \sqrt{5-2\,\beta+4\,\eta_1+\beta^2+4\,\beta\,\eta_1+4\,{\eta_1}^2\beta}}{2(1+2\,\eta_1+{\eta_1}^2)}$$

gives $\varphi_\epsilon'(\eta) \leq \alpha < 1$ for any $\epsilon < \epsilon_1$ and $\eta < \eta_1$. Furthermore, it follows that since $\{\epsilon_k\}_{k=1}^{\infty}$ decreases monotonically, the sequence $\{\eta_k\}_{k=1}^{\infty}$ decreases monotonically because φ_ϵ is monotone in ϵ. Thus, the sequence $\{\eta_k\}_{k=1}^{\infty}$ converges to its greatest lower bound. Let $\eta_* \equiv \inf\{\eta_k\}_{k=1}^{\infty}$. The sequence of contractions $\varphi_{\epsilon_k}(\eta)$ converges uniformly to $\varphi(\eta) = \beta\eta$ on $[0, \eta_1]$ as $k \longrightarrow \infty$. Consider the following array:

$$\begin{array}{ccccccc}
\varphi_{\epsilon_1}(\eta_1) & \varphi_{\epsilon_2}(\eta_1) & \varphi_{\epsilon_3}(\eta_1) & \cdots & \longrightarrow & \varphi(\eta_1) = \beta\eta_1 \\
\varphi_{\epsilon_1}(\eta_2) & \varphi_{\epsilon_2}(\eta_2) & \varphi_{\epsilon_3}(\eta_2) & \cdots & \longrightarrow & \varphi(\eta_2) = \beta\eta_2 \\
\varphi_{\epsilon_1}(\eta_3) & \varphi_{\epsilon_2}(\eta_3) & \varphi_{\epsilon_3}(\eta_3) & \cdots & \longrightarrow & \varphi(\eta_3) = \beta\eta_3 \\
\downarrow & \downarrow & \downarrow & \ddots & & \downarrow \\
\varphi_{\epsilon_1}(\eta_*) & \varphi_{\epsilon_2}(\eta_*) & \varphi_{\epsilon_3}(\eta_*) & \cdots & \longrightarrow & \varphi(\eta_*) = \beta\eta_*
\end{array} \tag{2.1.24}$$

It is apparent that $\varphi_{\epsilon_k}(\eta_k) \longrightarrow \beta\eta_*$ as $k \longrightarrow \infty$. More precisely, given $\delta > 0$, there is an N such that for $k > N$,

$$|\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_k)| < \delta/2, \text{ and } |\varphi(\eta_k) - \varphi(\eta_*)| < \delta/2, \tag{2.1.25}$$

so that

$$\begin{aligned}
|\eta_{k+1} - \varphi(\eta_*)| &= |\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_*)| \\
&\leq |\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_k)| + |\varphi(\eta_k) - \varphi(\eta_*)| \\
&< \delta.
\end{aligned} \tag{2.1.26}$$

Thus $\eta_{k+1} \longrightarrow \varphi(\eta_*)$, and since we assumed $\eta_{k+1} \longrightarrow \eta_*$, we have

$$\varphi(\eta_*) = \beta\eta_* = \eta_* \Rightarrow \eta_* = 0, \tag{2.1.27}$$

since $0 < \beta < 1$. □

Remark 2.1.1. Recall that

$$u_k = \begin{bmatrix} 1 \\ h_k \end{bmatrix}, \text{ with } \|h_k\| < \eta_k,$$

so by showing that $\lim_{k\to\infty} \eta_k = \eta_* = 0$, we have shown that the sequence of vectors u_k generated by the PPM converge to the standard basis vector e_1. This is the dominant eigenvector for the transformed problem, since the first column of U is v_1, so $Ue_1 = v_1$, and

$$\begin{array}{rrcl} & U^{-1}AUe_1 & = & \lambda_1 e_1 \\ \Leftrightarrow & A(Ue_1) & = & \lambda_1(Ue_1) \\ \Leftrightarrow & Av_1 & = & \lambda_1 v_1. \end{array} \tag{2.1.28}$$

Figure 2.1 illustrates the sequence of functions $\{\varphi_{\epsilon_k}\}_{k=1}^{\infty}$ converging uniformly to the limit $\beta\eta$ together with the sequence $\{\eta_k\}_{k=1}^{\infty}$.

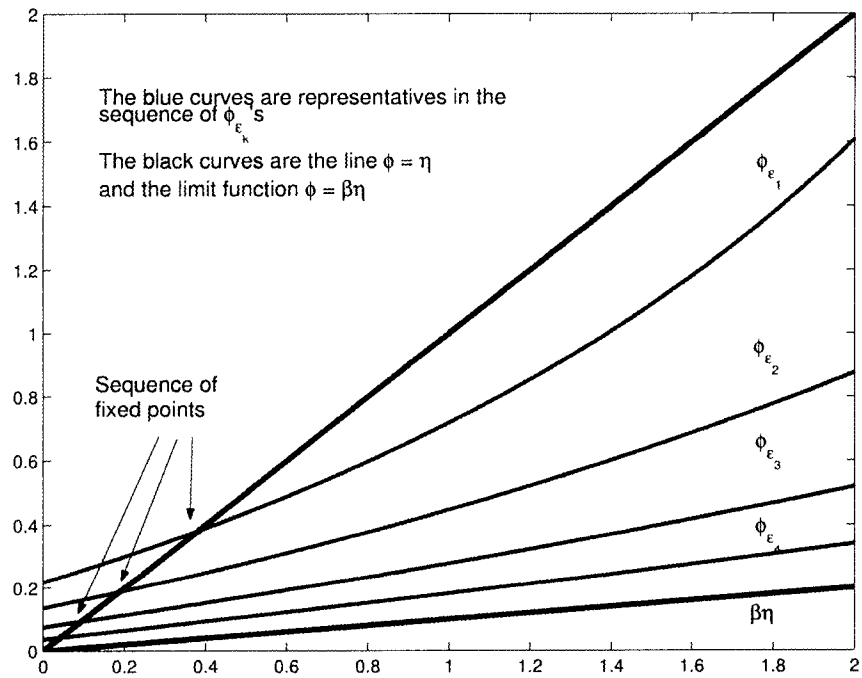

Figure 2.1: Convergence of the perturbed power method.

We mention that ill–conditioning in the dominant eigenvalue may limit the accuracy of this result. Recall that the condition number for the transformation matrix U is given by

$$\kappa(U) = \frac{1+\sigma}{\gamma}, \tag{2.1.29}$$

where γ is the inner product of the normalized left and right eigenvectors of A. Hence, the condition number of U is directly proportional to the secant of the angle between the eigenvectors. This angle is a condition number for the eigenvalue [97]. Thus U is ill–conditioned when λ_1 is ill–conditioned.

2.2 Infinite dimensions

This section extends the power method for matrices to an infinite–dimensional setting. An infinite–dimensional version of the power method was used in [4] to give an elementary proof of the Krein–Rutman theorem in 1991. The Krein–Rutman theorem is an infinite–dimensional extension of a classic theorem of Perron [84] which, roughly speaking, states that a positive square matrix has a simple dominant eigenvalue corresponding to a positive eigenvector. Krein and Rutman [69] extended Perron's finite dimensional result to compact linear operators on Banach spaces in 1950. Earlier proofs using degree theoretic methods are complex, see [92] for example. The approach in [4] is to view the power method as a discrete dynamical system on the unit sphere and prove convergence to the appropriate eigenvector. A dynamical systems approach was also used in [68] to study the behavior of the power method for nonnormal matrices in finite precision arithmetic.

2.2.1 The simplest case

The simplest proof for the power method occurs when the operator has a single dominant eigenvalue and a complete set of eigenvectors, in the sense that the span of the eigenvectors is dense. This is a direct extension to infinite dimensions of the discussion in §2.1.1. In the course of the proof, we need to use the following lemma, which can be found in [70]:

Lemma 2.2.1. *Let $\{x_1, \cdots, x_n\}$ be a linearly independent set of vectors in a normed space X (of any dimension, so possibly $n = \infty$). Then there is a number $C > 0$ such that for every choice of scalars $\alpha_1, \cdots, \alpha_n$, we have*

$$\left(\sum_{j=1}^{n} |\alpha_j| \right) \leq C \left\| \sum_{j=1}^{n} \alpha_j x_j \right\|. \tag{2.2.1}$$

Theorem 2.2.1. *Let A be a compact operator on a Hilbert space $\mathcal{H}$, and suppose that A has a single, simple dominant eigenvalue λ_1 and a complete set of eigenvectors $\{\phi_j\}_{j=1}^{\infty}$. Then the power method iteration converges to a multiple of the dominant eigenvector ϕ_1 corresponding to λ_1, provided the initial starting vector has a nonzero component in the direction of ϕ_1.*

Proof. We can assume with no loss of generality that $\|\phi_j\| = 1$ for all j. Let u_0 be any element of $\mathcal{H}$, and write u_0 as a linear combination of the eigenvectors ϕ_j:

$$u_0 = \sum_{j=1}^{\infty} c_j \phi_j; \quad c_1 \neq 0 \tag{2.2.2}$$

Since $A\phi_j = \lambda_j \phi_j$ for all j, m applications of A gives

$$A^m u_0 = A^m \left(\sum_{j=1}^{\infty} c_j \phi_j \right) = \sum_{j=1}^{\infty} c_j \lambda_j^m \phi_j. \tag{2.2.3}$$

Moving the first term to the other side, dividing by λ_1^m and taking norms gives

$$\begin{aligned}
\left\| \frac{A^m u_0}{\lambda_1^m} - c_1\phi_1 \right\| &= \left\| \sum_{j=2}^{\infty} c_j \left(\frac{\lambda_j}{\lambda_1}\right)^m \phi_j \right\| \\
&\leq \left|\frac{\lambda_2}{\lambda_1}\right|^m \sum_{j=2}^{\infty} |c_j| \quad (\|\phi_j\| = 1 \ \forall j) \\
&\leq \left|\frac{\lambda_2}{\lambda_1}\right|^m \sum_{j=1}^{\infty} |c_j| \\
&\leq \left|\frac{\lambda_2}{\lambda_1}\right|^m C \left\| \sum_{j=1}^{\infty} c_j\phi_j \right\| \quad \text{(lemma 2.2.1)} \\
&= \left|\frac{\lambda_2}{\lambda_1}\right|^m C\|u_0\|. \qquad (2.2.4)
\end{aligned}$$

Now defining $q_m := \dfrac{A^m u_0}{\lambda_1^m}$, given any $\epsilon > 0$,

$$m > \frac{\ln\epsilon - \ln(C\|u_0\|)}{\ln|\lambda_2| - \ln|\lambda_1|} \Rightarrow \|q_m - c_1 v_1\| < \epsilon, \qquad (2.2.5)$$

thus $q_m \to c_1 v_1$ as $m \to \infty$. □

Remark 2.2.1. The assumption that A is compact rules out the possibility of spectral values other than eigenvalues. It may well be that the power method still converges for noncompact operators possessing a dominant eigenvalue.

2.2.2 The power method for nonnormal operators

In this section, we show that under suitable conditions, the power method iteration converges for nonnormal operators on a Hilbert space. The technique used here also applies to normal operators, subsuming the results of the previous subsection. A major advantage of this technique is that it is applicable to the perturbed power method in infinite dimensions, as we show in §2.2.3.

Let A be a compact operator on a Hilbert space $\mathcal{H}$. Suppose that A has a single, simple dominant eigenpair (λ, v), with $\|v\| = 1$. Let $\mathcal{M}$ be the one-dimensional subspace spanned by v, so $\mathcal{H} = \mathcal{M} \oplus \mathcal{M}^{\perp}$. Since $\mathcal{M}$ is an invariant subspace, we can represent A formally as the 2×2 matrix with operator entries [30]

$$A = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix}, \tag{2.2.6}$$

where $A_1 \in \mathcal{B}(\mathcal{M})$, $A_2 \in \mathcal{B}(\mathcal{M}^{\perp}, \mathcal{M})$, and $A_3 \in \mathcal{B}(\mathcal{M}^{\perp})$. Let u be any element in $\mathcal{H}$ that does not lie entirely in $\mathcal{M}^{\perp}$. Let $u_{\mathcal{M}}$ and $u_{\mathcal{M}^{\perp}}$ be the components of u in $\mathcal{M}$ and $\mathcal{M}^{\perp}$ respectively, so that $u = u_{\mathcal{M}} + u_{\mathcal{M}^{\perp}}$. We can express the action of A on u as a matrix–vector product:

$$Au = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix} \begin{bmatrix} u_{\mathcal{M}} \\ u_{\mathcal{M}^{\perp}} \end{bmatrix} = \begin{bmatrix} A_1 u_{\mathcal{M}} + A_2 u_{\mathcal{M}^{\perp}} \\ A_3 u_{\mathcal{M}^{\perp}} \end{bmatrix} \tag{2.2.7}$$

Let $\mu = \max\{\|A_2\|, \|A_3\|\}$. In this analysis, we assume that $\mu < |\lambda|$. Applying A again, we have

$$\begin{aligned} A^2 u &= \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix} \begin{bmatrix} A_1 u_{\mathcal{M}} + A_2 u_{\mathcal{M}^{\perp}} \\ A_3 u_{\mathcal{M}^{\perp}} \end{bmatrix} \\ &= \begin{bmatrix} A_1^2 u_{\mathcal{M}} + A_1\big(A_2 u_{\mathcal{M}^{\perp}}\big) + A_2\big(A_3 u_{\mathcal{M}^{\perp}}\big) \\ A_3^2 u_{\mathcal{M}^{\perp}} \end{bmatrix}. \end{aligned} \tag{2.2.8}$$

Continuing in this way, after m applications of A, we obtain

$$A^m u = \begin{bmatrix} A_1^m u_{\mathcal{M}} + \displaystyle\sum_{j=1}^{m} A_1^{m-j}\Big(A_2 A_3^{j-1} u_{\mathcal{M}^{\perp}}\Big) \\ A_3^m u_{\mathcal{M}^{\perp}} \end{bmatrix}. \tag{2.2.9}$$

Since $\mathcal{M}$ is the one–dimensional subspace spanned by the dominant eigenvector v and $u_{\mathcal{M}} \in \mathcal{M}$, we have $u_{\mathcal{M}} = \alpha_0 v$ for some constant α_0. Furthermore, A_1 is the restriction of A to $\mathcal{M}$, so $A_1^m u_{\mathcal{M}} = \alpha_0 \lambda^m v$. Since $A_3 \in \mathcal{B}(\mathcal{M}^{\perp})$, $A_3^j u_{\mathcal{M}^{\perp}} \in \mathcal{M}^{\perp}$ for all j, and because $A_2 \in \mathcal{B}(\mathcal{M}^{\perp}, \mathcal{M})$,

$A_2 A_3^j u_{\mathcal{M}^\perp} \in \mathcal{M}$, therefore all of the terms in the sum in the first component of (2.2.9) are scalar multiples of v, and the j^{th} term in the sum contains the factor λ^{m-j}. Denoting the coefficient of $A_2 A_3^{j-1} u_{\mathcal{M}^\perp}$ by α_j, we may write

$$A^m u = \begin{bmatrix} \alpha_0 \lambda^m v + \sum_{j=1}^{m} \alpha_j \lambda^{m-j} v \\ A_3^m u_{\mathcal{M}^\perp} \end{bmatrix}. \tag{2.2.10}$$

Dividing by λ^m gives

$$\frac{A^m u}{\lambda^m} = \begin{bmatrix} \alpha_0 v + \sum_{j=1}^{m} \frac{\alpha_j}{\lambda^j} v \\ \frac{A_3^m u_{\mathcal{M}^\perp}}{\lambda^m} \end{bmatrix}. \tag{2.2.11}$$

Our objective is to show that (2.2.11) converges to a multiple of v as m increases. Doing so requires two things: showing that the second component converges to zero, and showing that the multiple of v in the first component converges. The former is easy, since we assumed that $\|A_3\| < |\lambda|$:

$$\left\| \frac{A_3^m u_{\mathcal{M}^\perp}}{\lambda^m} \right\| \leq \left(\frac{\|A_3\|}{|\lambda|} \right)^m \|u_{\mathcal{M}^\perp}\| \to 0 \text{ as } m \to \infty. \tag{2.2.12}$$

The latter is accomplished by noticing that the sum in (2.2.11) does not increase without bound, due to the assumption on the sizes of $\|A_2\|$ and $\|A_3\|$:

$$\begin{aligned} |\alpha_j| = \|\alpha_j v\| &= \|A_2 A_3^{j-1} u_{\mathcal{M}^\perp}\| \\ &\leq \|A_2\| \|A_3\|^{j-1} \|u_{\mathcal{M}^\perp}\| \\ &\leq \mu^j \|u_{\mathcal{M}^\perp}\|. \end{aligned} \tag{2.2.13}$$

Therefore, the second term in the first component of (2.2.11) can be bounded as

$$\left\| \sum_{j=1}^{m} \frac{\alpha_j}{\lambda^j} v \right\| \leq \sum_{j=1}^{m} \left| \frac{\mu}{\lambda} \right|^j \|u_{\mathcal{M}^\perp}\| = \frac{1 - \left| \frac{\mu}{\lambda} \right|^m}{1 - \left| \frac{\mu}{\lambda} \right|} \|u_{\mathcal{M}^\perp}\|. \tag{2.2.14}$$

Since $\left|\frac{\mu}{\lambda}\right| < 1$,

$$\lim_{m\to\infty} \left\| \alpha_0 v + \sum_{j=1}^{m} \frac{\alpha_j}{\lambda^j} v \right\| \leq |\alpha_0| + \frac{\|u_{\mathcal{M}^\perp}\|}{1 - \left|\frac{\mu}{\lambda}\right|} < \infty, \tag{2.2.15}$$

hence the first component in (2.2.11) converges to a multiple of v.

Note that if A is normal, then both $\mathcal{M}$ and $\mathcal{M}^\perp$ are invariant subspaces, and $A_2 \equiv 0$ in (2.2.6) [30]. This eliminates the summation term in (2.2.9), and the power method iteration is

$$\frac{A^m u}{\lambda^m} = \begin{bmatrix} \alpha_0 v \\ \frac{A_3^m u_{\mathcal{M}^\perp}}{\lambda^m} \end{bmatrix}, \tag{2.2.16}$$

which clearly converges to $\alpha_0 v$ by (2.2.12).

2.2.3 The perturbed power method

As in §2.2.2, let A be a compact operator on a separable Hilbert space $\mathcal{H}$ with a single, simple dominant eigenpair (λ, v), with $\|v\| = 1$. Also, we assume that $|\lambda|$=1 to avoid division by λ^m in the PPM algorithm. This amounts to using $\lambda^{-1}A$ in place of A, so there is no loss of generality. Let $\mathcal{H} = \mathcal{M} \oplus \mathcal{M}^\perp$, with $\mathcal{M}$ and $\mathcal{M}^\perp$ as before. Again, write A as a 2×2 matrix with operator entries

$$A = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix}, \tag{2.2.17}$$

where $A_1 \in \mathcal{B}(\mathcal{M})$, $A_2 \in \mathcal{B}(\mathcal{M}^\perp, \mathcal{M})$, and $A_3 \in \mathcal{B}(\mathcal{M}^\perp)$. In fact, normalizing so that $|\lambda| = 1$ gives $A_1 = I$. Furthermore, let E_k be a sequence of compact perturbations such that $\|E_k\| \to 0$, and define $P_n = \prod_{k=1}^{n}(A + E_k)$. The objective is to show that, given any $u_1 \in \mathcal{H}$, the sequence $\{u_n\}$ defined by $u_{n+1} := P_n u_1$ converges to a multiple of v. We can show the PPM converges using the technique presented in [98]. This analysis closely follows the development in §2.1.3, except that now the setting is infinite–dimensional.

The perturbation operators E_k can be written as

$$E_k = \begin{bmatrix} e_{11}^k & e_{12}^k \\ e_{21}^k & e_{22}^k \end{bmatrix}, \tag{2.2.18}$$

where $e_{11}^k \in \mathcal{B}(\mathcal{M})$, $e_{12}^k \in \mathcal{B}(\mathcal{M}^\perp, \mathcal{M})$, $e_{21}^k \in \mathcal{B}(\mathcal{M}, \mathcal{M}^\perp)$, and $e_{22}^k \in \mathcal{B}(\mathcal{M}^\perp)$. Applying P_k to u_k yields a vector u_{k+1} with a component in $\mathcal{M}$ and a component in $\mathcal{M}^\perp$. We normalize so that the component in $\mathcal{M}$ is v, with $\|v\| = 1$, and write

$$u_k = \begin{bmatrix} v \\ h_k \end{bmatrix}; \; v \in \mathcal{M}, \; h_k \in \mathcal{M}^\perp. \tag{2.2.19}$$

Let $\|A_2\| = \alpha$, $\|A_3\| = \beta$, and $\|E_k\| = \epsilon_k$. As before, we assume that $\alpha < 1$ and $\beta < 1$, so the norm of the operator A is achieved at the spectral radius. If this condition is not satisfied, we are not guaranteed convergence in the iteration below.

Let η_k be an upper bound for $\|h_k\|$. We derive an upper bound η_{k+1} for $\|h_{k+1}\|$ in the form of the quotient of a lower bound on the size of the component of $(A + E_k)u_k$ in $\mathcal{M}$ and an upper bound on the size of the component in $\mathcal{M}^\perp$. We have

$$(A + E_k)u_k = \begin{bmatrix} I + e_{11}^k & A_2 + e_{12}^k \\ e_{21}^k & A_3 + e_{22}^k \end{bmatrix} \begin{bmatrix} v \\ h_k \end{bmatrix} \tag{2.2.20}$$

The first component $v_{\mathcal{M}} = (I + e_{11}^k)v + (A_2 + e_{12}^k)h_k$. Since $\|v\| = 1$, we have

$$\begin{aligned} 1 &= \|v + e_{11}^k v - e_{11}^k v + A_2 v - A_2 v + e_{12}^k h_k - e_{12}^k h_k\| \\ &\leq \|v_{\mathcal{M}}\| + \|e_{11}^k v\| + \|A_2 v\| + \|e_{12}^k h_k\| \\ &\leq \|v_{\mathcal{M}}\| + \epsilon_k + \alpha\eta_k + \epsilon_k\eta_k. \end{aligned} \tag{2.2.21}$$

Therefore,

$$1 - \alpha\eta_k - (1 + \eta_k)\epsilon_k \leq \|v_{\mathcal{M}}\|. \tag{2.2.22}$$

Now, the second component $v_{\mathcal{M}^\perp} = e_{21}^k v + (A_3 + e_{22}^k)h_k$, so

$$\|v_{\mathcal{M}^\perp}\| \leq \epsilon_k + \|A_3\|\eta_k + \epsilon_k\|\eta_k\| = \beta\eta_k + \epsilon_k(1+\eta_k), \tag{2.2.23}$$

hence

$$\|h_{k+1}\| \leq \eta_{k+1} \equiv \frac{\beta\eta_k + \epsilon_k(1+\eta_k)}{1-\alpha\eta_k - (1+\eta_k)\epsilon_k}. \tag{2.2.24}$$

Now, define

$$\varphi_\epsilon(\eta) := \frac{\beta\eta + \epsilon(1+\eta)}{1-\alpha\eta_k - (1+\eta)\epsilon}, \tag{2.2.25}$$

so that $\eta_{k+1} = \varphi_{\epsilon_k}(\eta_k)$. The following theorem differs from Theorem 5.2 in [98] because the term $\alpha\eta_k$ places additional constraints on acceptable starting values for the sequences $\{\eta_k\}$ and $\{\epsilon_k\}$. Nonetheless, with this notation, we have

Theorem 2.2.2. *For $0 < \eta_1 < \dfrac{1-\sqrt{\beta}}{\alpha}$, there is an ϵ_1 such that if the sequence $\{\epsilon_k\}_{k=1}^\infty$ approaches zero monotonically, then the sequence defined by*

$$\eta_{k+1} = \varphi_{\epsilon_k}(\eta_k), \qquad k = 1, 2, 3, \ldots, \tag{2.2.26}$$

converges monotonically to zero.

Proof. The derivative of ϕ_ϵ is

$$\varphi'_\epsilon(\eta) = \frac{\beta+\epsilon}{1-\alpha\,\eta-\epsilon\,(1+\eta)} + \frac{(\beta\,\eta+\epsilon\,(1+\eta))\,(\alpha+\epsilon)}{(1-\alpha\,\eta-\epsilon\,(1+\eta))^2}, \tag{2.2.27}$$

and a calculation shows that if

$$\epsilon_1 \leq \frac{2\,\alpha\sqrt{\beta} - \beta\,\alpha^2 + 2\,\sqrt{\beta}\alpha^2 - 2\,\alpha\beta + \alpha^2 + \alpha^3}{2\,\alpha + 1 - 2\,\sqrt{\beta} + \alpha^2 + \beta - 2\,\sqrt{\beta}\alpha},$$

then $\varphi'_\epsilon(\eta) < 1$ for any $\epsilon < \epsilon_1$ and $\eta < \eta_1$. Furthermore, it follows that since $\{\epsilon_k\}_{k=1}^\infty$ decreases monotonically, the sequence $\{\eta_k\}_{k=1}^\infty$ decreases monotonically as well. Since $0 \leq \eta_k$ for all k, this sequence converges to its greatest

lower bound. Let $\eta_* \equiv \inf\{\eta_k\}_{k=1}^{\infty}$. The sequence of contractions $\varphi_{\epsilon_k}(\eta)$ converges uniformly to $\varphi(\eta) = \dfrac{\beta\eta}{1-\alpha\eta}$ on $[0, \eta_1]$ as $k \to \infty$. Consider the following array:

$$\begin{array}{ccccccc}
\varphi_{\epsilon_1}(\eta_1) & \varphi_{\epsilon_2}(\eta_1) & \varphi_{\epsilon_3}(\eta_1) & \cdots & \longrightarrow & \varphi(\eta_1) = \dfrac{\beta\eta_1}{1-\alpha\eta_1} \\
\varphi_{\epsilon_1}(\eta_2) & \varphi_{\epsilon_2}(\eta_2) & \varphi_{\epsilon_3}(\eta_2) & \cdots & \longrightarrow & \varphi(\eta_2) = \dfrac{\beta\eta_2}{1-\alpha\eta_2} \\
\varphi_{\epsilon_1}(\eta_3) & \varphi_{\epsilon_2}(\eta_3) & \varphi_{\epsilon_3}(\eta_3) & \cdots & \longrightarrow & \varphi(\eta_3) = \dfrac{\beta\eta_3}{1-\alpha\eta_3} \\
\downarrow & \downarrow & \downarrow & \ddots & & \downarrow \\
\varphi_{\epsilon_1}(\eta_*) & \varphi_{\epsilon_2}(\eta_*) & \varphi_{\epsilon_3}(\eta_*) & \cdots & \longrightarrow & \varphi(\eta_*) = \dfrac{\beta\eta_*}{1-\alpha\eta_*}
\end{array} \tag{2.2.28}$$

Evidently, $\varphi_{\epsilon_k}(\eta_k) \longrightarrow \dfrac{\beta\eta_*}{1-\alpha\eta_*}$ as $k \longrightarrow \infty$. More precisely, given $\delta > 0$, there is an N such that for $k > N$,

$$|\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_k)| < \delta/2, \text{ and } |\varphi(\eta_k) - \varphi(\eta_*)| < \delta/2, \tag{2.2.29}$$

so that

$$\begin{aligned}
|\eta_{k+1} - \varphi(\eta_*)| &= |\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_*)| \\
&\leq |\varphi_{\epsilon_k}(\eta_k) - \varphi(\eta_k)| + |\varphi(\eta_k) - \varphi(\eta_*)| \\
&< \delta.
\end{aligned} \tag{2.2.30}$$

Thus $\eta_{k+1} \longrightarrow \varphi(\eta_*)$, and since we assumed $\eta_{k+1} \longrightarrow \eta_*$, we have

$$\eta_* = \varphi(\eta_*) = \frac{\beta\eta_*}{1-\alpha\eta_*}. \tag{2.2.31}$$

Solving for η_*, we get $\eta_* = 0$ or $\eta_* = \dfrac{1-\beta}{\alpha}$. It follows that $\{\eta_k\}_{k=0}^{\infty}$ converges to $\eta_* = 0$, because it is decreasing monotonically, and since $\beta < 1$, we have $\eta_1 < \dfrac{1-\sqrt{\beta}}{\alpha} < \dfrac{1-\beta}{\alpha}$.

□

Remark 2.2.2. Recall that η_k is an upper bound for $\|h_k\|$, where h_k is the component of the k^{th} iterate of the PPM that lies in $\mathcal{M}^{\perp}$. Thus $\eta_k \to 0$

indicates that the successive iterates converge to a vector that lies entirely in $\mathcal{M}$; that is, the iterates converge to a multiple of the dominant eigenvector v.

Remark 2.2.3. The number α is a bound for $\|A_2\|$. Now, $A_2 \in \mathcal{B}(\mathcal{M}^\perp, \mathcal{M})$, which in some sense measures the departure from normality for the operator A. For normal operators, $\mathcal{M}$ is a reducing subspace, and we can take $A_2 = 0$. It is clear that the acceptable range of starting values for η_1 increases as α decreases, so if $\alpha = 0$, the hypothesis on η_1 coincides with Theorem 5.2 in [98].

Remark 2.2.4. The monotonic behavior of the sequence $\{\eta_k\}_{k=1}^{\infty}$ ensures that the iteration converged to the correct fixed point. Even if this is not the case, convergence to $\eta_* = 0$ still holds since

$$\beta < 1 \Rightarrow \lim_{\epsilon \to 0} \varphi_\epsilon'(0) < 1 \text{ and } \lim_{\epsilon \to 0} \varphi_\epsilon'\left(\frac{1-\beta}{\alpha}\right) = \frac{1}{\beta} > 1.$$

Remark 2.2.5. We assume that $\alpha < 1$ and $\beta < 1$. However, the result in [98] holds for matrices whose dominant singular value may be larger than the size of the dominant eigenvalue. This is accomplished by defining a transformation that induces a norm that approximates the spectral radius to arbitrary accuracy [97, 98]. Such transformations also hold in infinite dimensions [34, 113], so by invoking such a transformation (see (2.1.11)), we can make the same conclusions as Stewart in [98] in an infinite–dimensional setting.

Remark 2.2.6. The assumption that A and $\{E_k\}_{k=1}^{\infty}$ are compact does not play a role in the proof and is possibly unnecessary. However, it may be necessary in practice to make restrictive assumptions such as compactness or closedness. The compactness assumption makes it possible to avoid the possibility of residual or continuous spectra.

2.3 Extensions of the power method

In its most basic form, the power method produces a sequence of vectors and a sequence of numbers that converge to the dominant eigenpair of a matrix. We have shown that this idea can be extended to matrices that experience perturbations at each step of the iteration. We have also seen that the PPM still works in an infinite–dimensional setting. We now review two common extensions of the power method. *Spectral Enhancement* can be used to recover subdominant eigenpairs of a matrix by shifting the spectrum so that a different eigenvalue becomes dominant. This technique is often used in conjunction with *inverse iteration*, which is the power method applied to the inverse matrix. We use these extensions in later chapters to investigate the stability of a nonlinear system of ordinary differential equations and to bound the effect of linearization for *a-posteriori* error estimates of finite element computations for nonlinear differential equations.

2.3.1 Spectral enhancement

Since the rate of convergence of the power method is determined by the magnitude of the ratio of the two largest eigenvalues, the process converges very slowly if this ratio is near unity. It is possible to overcome this difficulty by introducing a shift ρ and performing the power method. That is, given an $n \times n$ matrix A with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, perform the power method on the "shifted" matrix $A - \rho I$, which has eigenvalues $\lambda_1 - \rho, \lambda_2 - \rho, \ldots, \lambda_n - \rho$. Appropriately chosen, the shift ρ creates a larger separation between the dominant eigenvalues $\lambda_1 - \rho$ and $\lambda_2 - \rho$, thus increasing the convergence rate of the method. This is known as *spectral enhancement* [97].

Spectral enhancement is not new. In 1955, J.H. Wilkinson's paper [110] explicitly employed this idea. His paper was a rebuttal to a note written by E. Bodewig, who claimed at the time that the only reason the power method enjoyed more popularity than his own computational method of solution via the characteristic equation [16, 17] was that "...the iteration method seems to be simpler and more mechanical in its application" [18]. Bodewig gives an example of a 4×4 matrix and starting vector for which the power method provides only four digits of accuracy after 1200 iterations, explaining that the slow convergence is due to the closeness in size of the two dominant eigenvalues, and the near orthogonality of the starting vector to the dominant eigenvector. He explains that the convergence is not appreciably improved by the choice of a different starting vector, and also states (with no reasons given): "...nor should we waste time by computing the eigenproblem of matrices of the form $A + c$ with varying c's. The time is better spent computing the characteristic equation..." [18]. In Wilkinson's response, he revisits the 4×4 example, introducing the shift $\rho = 2$, and "all four roots were found in a few seconds iteration time..." [110]. By way of comparison, I performed these experiments on MATLAB using the same schemes implemented by the authors. While it is true that the unshifted problem never gives better than four digits accuracy in the dominant eigenvalue in 1200 iterations, using $\rho = 2$ as spectral enhancement produced fourteen accurate digits after 78 iterations, when compared with MATLAB's built–in eig function. Furthermore, Wilkinson points out that Bodewig's example does not speak to other issues, such as preserving accuracy with larger matrices, which the power method handles better than a method that seeks to find the characteristic equation.

2.3.2 Inverse iteration

While spectral enhancement can improve the convergence properties of the power method, it may not always be the case that there exists a shift for which one eigenvalue becomes much larger than the others. On the other hand, there are always shifts for which one eigenvalue is much smaller than the others. This suggests coupling spectral enhancement with the power method applied to the inverse matrix. It is easy to see that if λ is an eigenvalue of the matrix A corresponding to the eigenvector v, and ρ is a scalar, then $\frac{1}{\lambda - \rho}$ is an eigenvalue of $(A - \rho I)^{-1}$ with the same eigenvector:

$$Av = \lambda v \Rightarrow Av - \rho v = \lambda v - \rho v, \tag{2.3.1}$$

so

$$(A - \rho I)v = (\lambda - \rho)v \Rightarrow \frac{1}{(\lambda - \rho)}v = (A - \rho I)^{-1}v. \tag{2.3.2}$$

Thus, if ρ is an initial approximation to some eigenvalue λ_k of A, performing the power method with the matrix $(A - \rho I)$ produces a sequence that converges to the eigenvector corresponding to λ_k. In implementing the algorithm, the k^{th} iterate is produced by solving the linear system

$$(A - \rho I)u_k = u_{k-1}, \tag{2.3.3}$$

then normalizing so that $\|u_k\| = 1$ prior to proceeding to the next iterate. The process tends to converge very quickly, since $\rho - \lambda_k$ is generally much smaller than $\rho - \lambda_i$ for $i \neq k$, provided λ_k is reasonably separated from the other eigenvalues. For example, if we begin with an initial vector

$$u_0 = c_1 v_1 + c_2 v_2 + \ldots + c_n v_n, \tag{2.3.4}$$

then

$$u_1 = (A - \rho I)^{-1} u_0 = \frac{c_1}{\lambda_1 - \rho} v_1 + \frac{c_2}{\lambda_2 - \rho} v_2 + \ldots + \frac{c_n}{\lambda_n - \rho} v_n. \qquad (2.3.5)$$

Provided c_k is roughly the same size as the other constants, the value $\frac{c_k}{\lambda_k - \rho}$ is much larger than the other components of the vector u_1, so that even the first iterate provides a good approximation to a multiple of the eigenvector corresponding to λ_1. Of course, this argument is based on expanding the initial vector in a basis of eigenvectors, as in §2.1.1. Even if this is not possible for the matrix A, the conclusion is the same provided the eigenvalue λ_k is simple.

The matrix $(A - \rho I)$ is ill–conditioned if ρ is close to an eigenvalue of A, but the inverse power method with shifting still works well, even though we solve the ill–conditioned linear system $(A - \rho I)^{-1} u_k = u_{k-1}$ at each stage of the iteration. In [109], Watkins explains why this is so by viewing the numerical solution as an exact solution to a perturbed problem and considering the residual. Provided the eigenvector corresponding to the desired eigenvalue is well–conditioned, a small residual implies that the iterates are good approximations and the routine can be terminated once the residual is made sufficiently small. However, if the eigenvalue itself is poorly conditioned, it is possible that the residuals never become sufficiently small [112].

2.4 Linearization and the perturbed power method

Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot, \cdot \rangle$, and F a map on $\mathcal{H}$ with Fréchet derivative dF. In this section, we show that the difference $F(x + h) - F(x)$ can be interpreted as the action of the derivative

dF at x acting on h, except that the derivative is subject to a bounded, linear perturbation whose norm decreases with $\|h\|$. Essentially, we take the higher order error term in the definition of the linearization and incorporate its effect together with the action of the derivative. It is necessary to view the linearization in this way in order to understand the differences $F(x+h) - F(x)$ in terms of the perturbed power method. By definition, we have [1, 114]

$$F(x+h) - F(x) = dF(x)h + r(h); \quad r(h) \text{ is } o(\|h\|), \tag{2.4.1}$$

where $o(\|h\|)$ means $\dfrac{\|r(h)\|}{\|h\|} \to 0$ as $h \to 0$. Now, given $h \in \mathcal{H}$, define a map $E_h : \mathcal{H} \to \mathcal{H}$ by

$$E_h(x) := \frac{\langle x, h\rangle r(h)}{\|h\|^2}. \tag{2.4.2}$$

The map E_h is linear: for $x, y \in \mathcal{H}$, and $\alpha, \beta \in \mathbb{R}$,

$$\begin{aligned} E_h(\alpha x + \beta y) &= \frac{\langle \alpha x + \beta y, h\rangle r(h)}{\|h\|^2} \\ &= \frac{\alpha\langle x, h\rangle r(h)}{\|h\|^2} + \frac{\beta\langle y, h\rangle r(h)}{\|h\|^2} \\ &= \alpha E_h(x) + \beta E_h(y). \end{aligned} \tag{2.4.3}$$

Boundedness is also straightforward: for $x \in \mathcal{H}$,

$$\|E_h(x)\| = \left\| \frac{\langle x, h\rangle r(h)}{\|h\|^2} \right\| \leq \frac{\|r(h)\|}{\|h\|^2}\|h\|\|x\| \leq \frac{\|r(h)\|}{\|h\|}\|x\|, \tag{2.4.4}$$

and it follows that

$$\|E_h\| \leq \frac{\|r(h)\|}{\|h\|}, \tag{2.4.5}$$

so $\|E_h\| \to 0$ as $h \to 0$. Also, the value at h is

$$E_h(h) = \frac{\langle h, h\rangle r(h)}{\|h\|^2} = r(h), \tag{2.4.6}$$

so $E_h(h)$ is $o(\|h\|)$. Now, we can write

$$\begin{aligned} F(x+h) - F(x) &= dF(x)h + r(h) \\ &= [dF(x) + E_h]h, \end{aligned} \tag{2.4.7}$$

so we can interpret the difference $F(x+h)-F(x)$ as the action of $dF(x)+E_h$ on h. This is important for an algorithm that utilizes $F(x+h) - F(x)$ to compute the action of the derivative in a matrix–free way, so reconstruction of $dF(x)$ is not necessary. This is an advantage when the dimension is large, because numerically computing $dF(x)$ is prohibitively expensive. By casting the differencing as $[dF(x) + E_h]h$, we can appeal to the results about the convergence of the PPM and gain confidence that the algorithm converges to the correct eigenpair for the derivative $dF(x)$. We refer to the process of emulating the action of $dF(x) + E_h$ on h by computing the difference $F(x+d) - F(x)$, as the *nonlinear power method* (NLPM).

For applications in which we desire information about the dominant eigenvalue and eigenvector only, we can implement the NLPM in a matrix–free way that directly extends the power method, except we evaluate function differences to mimic the matrix action on vectors. Thus, given a nonlinear function F whose Fréchet derivative exists and has a single dominant eigenvalue at a point U, we can implement the NLPM according to the following algorithm:

Algorithm 2.4.1. *The Nonlinear Power Method (NLPM)–Case 1*

1. *Choose a sequence* $1 > \alpha_1 > \alpha_2 > \ldots$ *of numbers converging monotonically to zero.*

2. *Input* U, F.

3. *Choose a starting vector* δ_0 *with* $\|\delta_0\| = 1$.

4. *For* $k = 1, 2, \ldots$

5. *Set* $\tilde{\delta}_k = F(U + \delta_{k-1}) - F(U)$ *% emulate* $dF(U)\delta_{k-1}$ *by differencing*

6. *Set* $\lambda_k = \dfrac{\langle \tilde{\delta}_k, \delta_{k-1} \rangle}{\langle \delta_{k-1}, \delta_{k-1} \rangle}$ *% approximate eigenvalue*

7. *Set* $v_k = \dfrac{\tilde{\delta}_k}{\|\tilde{\delta}_k\|}$ *% normalized eigenvector approximation*

8. *Set* $\delta_k = \alpha_k v_k$ *% decrease size of* $(k+1)^{st}$ *iterate by the factor* α_k

The sequence $\{\lambda_k\}$ converges to the dominant eigenvalue and $\{v_k\}$ to the corresponding normalized eigenvector of $dF(U)$. The vectors $\{\delta_k\}$ converge to zero, but their directions tend toward that of the dominant eigenvector as the iteration proceeds. By employing equation (2.4.7) iteratively, at step k, we have

$$v_k = \frac{\prod_{i=1}^{k-1} \alpha_i}{\prod_{i=1}^{k} \|\tilde{\delta}_i\|} \left[\prod_{i=0}^{k-1} (dF(U) + E_{\delta_i}) \right] \delta_0. \tag{2.4.8}$$

If F is linear, then

$$F(U + \delta) - F(U) = F(\delta) = dF(U)\delta, \tag{2.4.9}$$

and in this case all the perturbation matrices are zero, there is no need to drive the sequence $\{\delta_k\}$ to zero, so we may take $\alpha_k = 1$ for all k. The products are replaced by powers, and equation (2.4.8) above is

$$v_k = \frac{dF(U)^{k-1}}{\|dF(U)^{k-1}\|} v_0, \tag{2.4.10}$$

which shows that the algorithm reduces to the familiar power method in the linear case.

The results in [98] indicate that the convergence of the power method is not affected by perturbations whose norms tend to zero, which is a condition required by the linearization in the NLPM. However, the theoretical result does not account for finite precision errors in the power method computation. The analysis indicates that the presence of roundoff error does not significantly alter the convergence result, under fairly mild restrictions on the matrix. In [98] Stewart shows that, in the case where the perturbations do not converge to zero, the iteration defined by (2.1.22) converges to

$$\eta_* = \frac{2}{c + \sqrt{c^2 - 4}}, \tag{2.4.11}$$

where

$$c = \frac{1 - \beta - 2\epsilon}{\epsilon}. \tag{2.4.12}$$

Now, provided β is not too near one (recall that β is essentially the size of the sub–dominant eigenvalue), then the size of η_* is roughly the same size as ϵ. For instance, if the computations are done in IEEE double precision, ϵ is roughly 10^{-14} for a matrix of size $10,000 \times 10,000$. If $\beta = 0.8$, then $\eta_* = 5 \times 10^{-14}$. Thus, the numerical result is close to the theoretical result ($\eta_* = 0$) for large matrices. Ill–conditioning in the dominant eigenvalue also affects the accuracy of the result, as the above discussion applies to the transformed problem $\mathrm{diag}(1, B)$, and the condition number of the transformation matrix is proportional to the condition of the dominant eigenvalue.

The optimal minimum size of ϵ in algorithm (2.4.1) is the same as that for the error analysis of a difference quotient in one dimension: we should let ϵ get as small as $\sqrt{8\epsilon_0}$, where ϵ_0 is the size of the rounding unit. To see this, let $f : \mathbb{R}^n \to \mathbb{R}^n$ and assume that f' and f'' exist as Fréchet derivatives

on a neighborhood N of x. Then Taylor's theorem implies [114]

$$f(x+\epsilon) - f(x) = f'(x)\epsilon + R, \tag{2.4.13}$$

where the remainder R satisfies

$$\|R\| \leq \frac{1}{2} \sup_{\tau \in [0,1]} \|f''(x+\tau\epsilon)[\epsilon,\epsilon]\|. \tag{2.4.14}$$

Now, since f is continuously Gâteaux differentiable in all directions, we may write

$$f(x+\epsilon w) - f(x) = f'(x)\epsilon w + R, \quad \|w\| = 1. \tag{2.4.15}$$

Since f'' is continuous, there is a constant C that bounds its value on the closure of N, so we get

$$\left\| \frac{f(x+\epsilon w) - f(x)}{\epsilon} - f'(x)w \right\| \leq \frac{C\epsilon}{2}. \tag{2.4.16}$$

From standard results on round–off error for calculations with real numbers [109], we know that $|x - \tilde{x}| \leq \epsilon_0 |x|$, where $\tilde{x}$ is an approximation to the real number x which has s digits, and this holds in higher dimensions. For instance, suppose that $x, \tilde{x}, y,$ and $\tilde{y} \in \mathbb{R}^n$, so that the i^{th} component of $\tilde{x}$ is an s–digit approximation to the i^{th} component of x, and similarly for y

and $\tilde{y}$. If $\|\cdot\|$ is the Euclidean norm, we have

$$\begin{aligned}
\|(x-y)-(\tilde{x}-\tilde{y})\| &= \left(\sum_{i=1}^{n}|(x_i-y_i)-(\tilde{x_i}-\tilde{y_i})|^2\right)^{\frac{1}{2}} \\
&\leq \left(\sum_{i=1}^{n}\left(|x_i-\tilde{x_i}|+|y_i-\tilde{y_i}|\right)^2\right)^{\frac{1}{2}} \\
&= \left(\sum_{i=1}^{n}|x_i-\tilde{x_i}|^2+|y_i-\tilde{y_i}|^2+2|x_i-\tilde{x_i}||y_i-\tilde{y_i}|\right)^{\frac{1}{2}} \\
&\leq \left(\sum_{i=1}^{n}\epsilon_0^2|x_i|^2+\epsilon_0^2|y_i|^2+2\epsilon_0^2|x_i||y_i|\right)^{\frac{1}{2}} \\
&= \epsilon_0\left(\sum_{i=1}^{n}|x_i|^2+|y_i|^2+2|x_i||y_i|\right)^{\frac{1}{2}} \\
&= \epsilon_0\left(\sum_{i=1}^{n}|x_i+y_i|^2\right)^{\frac{1}{2}} \\
&= \epsilon_0\|x+y\| \\
&\leq \epsilon_0(\|x\|+\|y\|).
\end{aligned} \tag{2.4.17}$$

Therefore, the roundoff error in the finite difference approximation in n–dimensions is

$$\frac{\|f(x+\epsilon w)-f(x)\|}{\epsilon} \leq \frac{4\epsilon_0\|f(x)\|}{\epsilon}, \tag{2.4.18}$$

and so the true error of approximation is the sum:

$$\text{total error} \leq C\left(\frac{\epsilon}{2}+\frac{4\epsilon_0}{\epsilon}\right).$$

In the above we assume that the constant C bounds the value both of f and f'' on $\bar{N}$. Minimization with respect to ϵ gives $\sqrt{8\epsilon_0}$ for the size that the perturbations should be allowed to reach.

In the application to the effects of linearization on *a-posteriori* error estimation in chapter 4, the quantity of interest is not the dominant eigenvalue, but rather the norm of the derivative operator, which is given

by the dominant singular value. In cases where the derivative operator is not normal, a matrix–free version of the NLPM cannot be implemented to recover the norm of the operator. Instead, we reconstruct the Jacobian using a full set of differences. We assume that we are working in a finite–dimensional vector space, possibly a finite–dimensional subspace of an infinite–dimensional space. Let $\{e_j\}_{j=1}^n$ be a basis for the space. Given $\epsilon > 0$, we define the matrix $D_\epsilon F(U)$ columnwise, where the i^{th} column is given by

$$[D_\epsilon F(U)]_i = \left[\frac{F(U + \epsilon e_i) - F(U)}{\epsilon}\right]. \tag{2.4.19}$$

Since we assume that the differential at U exists, we have

$$\begin{aligned} D_\epsilon F(U) &= [dF(U)e_1 + c_1(\epsilon), dF(U)e_2 + c_2(\epsilon), \cdots, dF(U)e_n + c_n(\epsilon)] \\ &= dF(U) + E_\epsilon, \end{aligned} \tag{2.4.20}$$

where the matrix $E_\epsilon = [c_1(\epsilon), c_2(\epsilon), \cdots, c_n(\epsilon)]$, and $\|E_\epsilon\| = o(\epsilon)$. In this setting, the algorithm is as follows:

Algorithm 2.4.2. *The Nonlinear Power Method (NLPM)–Case 2*

1. *Choose* v_0 *with* $\|v_0\| = 1$ *and* $\epsilon_0 > 0$
2. *For* $k = 1, 2, \ldots$
3. *Calculate the matrix* $D_{\epsilon_{k-1}} F(U)$
4. *Set* $v_k = D_{\epsilon_{k-1}} F(U) v_{k-1}$
5. *Set* $\lambda_k = \langle v_k, v_{k-1} \rangle$
6. *Set* $v_k = \dfrac{v_k}{\|v_k\|}$

7. Set $\epsilon_k < \epsilon_{k-1}$

In this case, at step k we have

$$v_k = \frac{\left[\prod_{i=0}^{k-1} D_{\epsilon_i} F(U)\right] v_0}{\left\| \left[\prod_{i=0}^{k-1} D_{\epsilon_i} F(U)\right] v_0 \right\|} = \frac{\left[\prod_{i=0}^{k-1} (dF(U) + E_{\epsilon_k})\right] v_0}{\left\| \left[\prod_{i=0}^{k-1} (dF(U) + E_{\epsilon_k})\right] v_0 \right\|}. \tag{2.4.21}$$

Now, $\epsilon_k \to 0 \Rightarrow \|E_{\epsilon_k}\| \to 0$, and again the analysis indicates that the method converges to the dominant eigenvector for $dF(U)$ and that $\{\lambda_k\}$ is a sequence of increasingly accurate approximations to the dominant eigenvalue. Just as with case 1, the expression (2.4.21) reduces to the power method (2.4.10) in case F is linear. Algorithms (2.4.1) and (2.4.2) are easily seen to be straightforward extensions of the standard power method algorithm (2.1.1) that we presented in §2.1.1.

This algorithm is not practical to implement, since computing v_k in this manner requires re–calculating the matrix $D_{\epsilon_k} F(U)$ at each iteration. This is too expensive for problems of large dimension. A more practical implementation is to fix a small value of ϵ and run the power method. From (2.4.20), the power method computes an approximation to the perturbed eigenpair $(\tilde{\lambda}, \tilde{v})$ of the matrix $dF(U) + E_\epsilon$. From results on the perturbation theory of eigenvalues and eigenvectors [97], we have

$$|\lambda - \tilde{\lambda}| = O(\|E\|), \tag{2.4.22}$$

and

$$\|v - \tilde{v}\| = O(\|E\|), \tag{2.4.23}$$

where (λ, v) is the dominant eigenpair of $dF(U)$. By reconstructing an approximation to the full matrix $dF(U)$, we can generalize algorithm (2.4.2) to recover the dominant singular value (and hence approximate $\|dF(U)\|$)

by using $(D_\epsilon F(U))^\top$ on alternate iterations of the power method and taking the square root of the result.

We close this chapter by noting that our use of the term "nonlinear power method" differs from that of [55] and [34], where the authors use the term to describe iterative methods for solving nonlinear eigenvalue problems of the form

$$\begin{cases} \rho L(u) = f(x, u(x)), & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega, \end{cases} \tag{2.4.24}$$

where

$$L(u) = -\sum_{i,k=1}^{N} \partial_i(a_{ik}(x)\partial_k u(x)) + a(x)u(x) \tag{2.4.25}$$

is a uniformly elliptic formally self–adjoint differential operator on a bounded, open region $\Omega \subset \mathbb{R}^N$, with various smoothness assumptions on a_{ik}, a, f and $\partial\Omega$.

Chapter 3

APPLICATION TO STABILITY CALCULATIONS

In this chapter, we utilize the NLPM to investigate the stability properties of systems of ordinary differential equations. After reviewing basic stability theory for linearized systems, we discuss how to implement the nonlinear power method to determine stability without reconstructing the entire Jacobian matrix. We also indicate how spectral enhancement can be incorporated into the algorithm.

3.1 Nonlinear systems of ordinary differential equations

In this section, we review the basics of stability theory for a system of nonlinear ordinary differential equations. See any of [29, 58, 62, 83, 101] for a comprehensive treatment.

We consider the nonlinear, autonomous system of differential equations

$$\dot{u} = f(u), \tag{3.1.1}$$

where $u \in \mathbb{R}^n$, and $f : E \to \mathbb{R}^n$, where E is an open subset of $\mathbb{R}^n$. The fundamental existence–uniqueness theorem says the following:

Theorem 3.1.1. *Let E be an open subset of $\mathbb{R}^n$ containing u_0 and assume that $f \in C^1(E)$. Then there exists an $a > 0$ such that the initial value problem*

$$\begin{cases} \dot{u} = f(u), \\ u(0) = u_0, \end{cases} \tag{3.1.2}$$

has a unique solution $u(t)$ on the interval $[-a, a]$.

If f is merely continuous, the system (3.1.2) has a solution, but uniqueness is not guaranteed.

The Jacobian of f at a point u is denoted by $Df(u)$. A point $u_0 \in \mathbb{R}^n$ is an *equilibrium point* of (3.1.1) if $f(u_0) = 0$. An equilibrium point is called *hyperbolic* if none of the eigenvalues of the Jacobian matrix $Df(u_0)$ has zero real part. The equilibrium point is called a *source* if all of the eigenvalues of $Df(u_0)$ have positive real part, and a *sink* if they all have negative real part. In case $Df(u_0)$ has at least one eigenvalue with positive real part and one with negative real part, the equilibrium point is called a *saddle*. The system

$$\dot{u} = Df(u_0)u \tag{3.1.3}$$

is called the *linearization* of (3.1.1) at u_0. If u_0 is a hyperbolic equilibrium point, then the Hartman–Grobman theorem states that the qualitative structure of the nonlinear system (3.1.1) is the same as the linearization (3.1.3) [57]. Thus, the eigenvalues of the Jacobian $Df(u_0)$ play an important role in determining the stability of the nonlinear system (3.1.1).

3.2 Stability calculations

We give some examples of how the nonlinear power method is used to investigate the stability of an equilibrium point for a nonlinear system of ODE's.

3.2.1 Inverse iteration

We use the NLPM to determine the stability properties of the nonlinear system (3.1.1), by determining the signs of the eigenvalues of the linearization (3.1.3). If the system (3.1.3) is very large, it may be impractical or even impossible to form the Jacobian matrix $Df(u_0)$ and compute an eigen–decomposition. An example is the space discretization of a parabolic partial differential equation. The NLPM provides a way to analyze the problem in a matrix–free fashion, in which the action of the Jacobian is approximated by function evaluations and differencing.

Suppose that $f : E \to \mathbb{R}^n$ is $C^1(E)$. If u_0 is an equilibrium point for f and $Df(u_0) \neq 0$, then by the inverse function theorem, f has a continuously differentiable inverse on an open set W containing $f(u_0)$. Denote the inverse by g. Now, for $u \in W$, starting with

$$f(g(u)) = u, \tag{3.2.1}$$

and using the chain rule, we obtain

$$D(f(g(u))) = Df(g(u))Dg(u) = Du = I, \tag{3.2.2}$$

and it follows that

$$Dg(u) = \left[Df(g(u))\right]^{-1}. \tag{3.2.3}$$

If u_0 is an equilibrium solution, then

$$f(u_0) = 0 \Rightarrow g(0) = u_0, \tag{3.2.4}$$

and, in particular,

$$Dg(0) = \left[Df(u_0)\right]^{-1}. \tag{3.2.5}$$

In this case, the simple relationship between Df and Dg means that their structures are similar, e.g. both Dg and Df are symmetric (or not) simultaneously. In the case that both are symmetric, the eigenvalues are all real and the analysis is simpler. Later, when we analyze the error of linearization for computational error estimates, the map under study is not as directly related to the "forward" operator, and issues such as symmetry and normality are less clear. When we compute the dominant eigenvalue of $Dg(0)$, we are computing the reciprocal of the smallest eigenvalue in magnitude of $Df(u_0)$. If the eigenvalues $\{\lambda_1, \cdots, \lambda_n\}$ of $Df(u_0)$ are ordered so that $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_2 < \lambda_1$, where $|\lambda_n| \geq \cdots \geq |\lambda_2| > |\lambda_1|$, and we determine that λ_1 is negative, then u_0 is a sink. Thus, we just need to determine the sign of the dominant eigenvalue of $Dg(0)$. Note that the condition on the eigenvalues is satisfied in some interesting cases, as in the semi-discretization of a parabolic problem. We demonstrate this idea with the bistable problem

$$\begin{cases} u_t = u_{xx} + \lambda(u - u^3), \ 0 < x < 1, \ 0 < t, \\ u_x(0,t) = u_x(1,t) = 0, \ 0 < t, \ u(x,0) \text{ given.} \end{cases} \tag{3.2.6}$$

The problem has attracting steady state solutions $u \equiv 1$ and $u \equiv -1$, and is a prototypical example of *metastability*, whereby generic solutions converge to one of these steady states, but in doing so remain nearly stationary for long periods of time [20, 24, 25, 46, 50].

First, we find $u_0 = g(0)$ by solving $f(u) = 0$ by some numerical method. In order to determine the dominant eigenvalue of the solution operator of the linear problem obtained by linearizing around the stationary solution, we choose a random data vector d_0 and compute successive approximations to the dominant eigenvalue of $g'(0)$ by subtracting:

$$d_1 := g(0 + d_0) - g(0) \approx Dg(0)d_0. \tag{3.2.7}$$

In general, the n^{th} iterate is given by

$$d_n := g(0 + d_{n-1}) - g(0) \approx Dg(0)d_{n-1}. \tag{3.2.8}$$

At each step of the iteration, the vector d_n is rescaled to have decreasing norms since we are linearizing and the new value is used in the next step of the iteration. This way, we emulate the action of $Dg(0) = [Df(u_0)]^{-1}$ on vectors by computing the differences $g(0+d_{n-1}) - g(0)$, hence the process of computing the differences is a way to implement the inverse power method for the Jacobian matrix in the linearization (3.1.3).

In this notation, "$g(d)$" means "solve the equation $f(u) = d$ for u," which we do numerically by solving the nonlinear system of equations $f(u) - d = 0$. We wish to make the algorithm as matrix-free as possible. One approach is Broyden's method, which is analogous to Newton's method, except that a secant approximation to the Jacobian is used [33]. The idea is that approximating the Jacobian leads to an affine model that satisfies a secant equation. In $n \geq 2$ dimensions, the secant equation does not completely specify the affine model, so further conditions are imposed. Namely, we try to minimize the change in the model and still satisfy the secant equation. This leads to *Broyden's update*: the approximate Jacobian for the succeeding iteration uses the current approximation, plus a rank one update. In [33], the algorithm is given as:

Algorithm 3.2.1. *Broyden's Method*

Given $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n, x_0 \in \mathbb{R}^n,$ *and* $A_0 \in \mathbb{R}^{n \times n}$

1. *Do for* $k = 0, 1, \ldots :$
2. *Solve* $A_k s_k = -F(x_k)$ *for* s_k % s_k *is the current Newton step*
3. $x_{k+1} := x_k + s_k$
4. $y_k := F(x_{k+1}) - F(x_k)$
5. $A_{k+1} := A_k + \dfrac{(y_k - A_k s_k) s_k^T}{s_k^T s_k}.$ % *add rank one update*

There is some flexibility in the choice of A_0, the initial approximation to the Jacobian. One possibility is to use finite differences to approximate the Jacobian at the initial point x_0. This may be costly if function evaluations are expensive, and inaccurate if x_0 is far from a root. A common alternative that is simple and effective is to choose $A_0 = -I$. In essence, this choice allows the system to choose the direction of the first step. In practice, this choice does not significantly alter the performance of the method, and is clearly more advantageous in terms of computational expense. We can further reduce the computational cost by implementing a limited memory Broyden method, which reduces the memory storage of the Broyden matrix from n^2 elements to $2pn$, where $p < n$ [105].

So, to use the NLPM to determine the stability of an equilibrium solution for the system (3.1.1), we use the following algorithm.

Algorithm 3.2.2. *NLPM for Stability Calculations*

1. *Compute the equilibrium solution* u_0 *using Broyden's method to solve* $f(u) = 0$. *That is, compute* $u_0 = g(0)$.

2. *choose an initial starting vector* δ_0, *with* $\|\delta_0\| = 1$.

3. *Do for* $k = 1, 2, 3, \ldots$

4. $\delta_k = g(\delta_{k-1}) - g(0)$ *% emulate* $Dg(0)\delta_{k-1}$

5. $\lambda_k = \dfrac{\langle \delta_{k-1}, \delta_k \rangle}{\|\delta_k\|^2}$ *% approximate eigenvalue*

6. $\delta_k = \dfrac{\delta_k}{\|\delta_k\|}$ *% normalized approximate dominant eigenvector*

7. $\delta_k = \alpha_k \dfrac{\delta_k}{\|\delta_k\|}$ *% decrease norm of next iterate by the factor* α_k.

As an example, we consider a semi–discretization of the bistable reaction-diffusion equation for the vector $u = [u_1, \ldots, u_n]^T$, written

$$\dot{u} = Au + \xi(u - u^3), \quad u(0) = u_0, \tag{3.2.9}$$

where the matrix A is the tridiagonal discrete Laplace operator, ξ is a real parameter and the nonlinear term is to be interpreted as $\xi(u_i - u_i^3)$ in the i^{th} position. Thus, this system approximates the bistable equation,

$$\begin{cases} u_t = u_{xx} + \xi(u - u^3), & 0 < x < 1, \quad 0 < t, \\ u_x(0,t) = u_x(1,t) = 0, & 0 < t, \quad u(x,0) \text{ given}, \end{cases} \tag{3.2.10}$$

using lumped mass quadrature. Notice that $u \equiv 0, u \equiv 1$, and $u \equiv -1$ are solutions, and correspond to the steady–state solutions of the PDE.

Example 3.2.1. Let $n = 6$ and $\xi = 0.1$. Beginning with an initial guess near $u \equiv 1$, the algorithm first finds the equilibrium solution using Broyden's method, then chooses an initial starting vector and uses the relations in (3.2.7) to compute successive approximations to the dominant eigenpair of the linearization. After 26 iterations, the computed eigenvalue is -4.9999972. By way of comparison, a centered difference approximation to

the Jacobian of $Au + \xi(u - u^3)$ at $u \equiv 1$ was performed and MATLAB was used to compute the following eigenvalues for the inverse:

$$-5.0000, -2.1370, -0.8333, -0.4545, -0.3125, \text{ and } -0.2543.$$

Since the eigenvalues of interest are the reciprocals, we conclude that the smallest eigenvalue in magnitude of $f'(u_0)$ is -0.2, and the equilibrium solution is stable. The components of the approximate dominant eigenvector from this implementation of the nonlinear power method and from MATLAB are shown in table 3.1.

Approximate dominant eigenvector	
Power Method	MATLAB
0.40824784206273	0.40824829013435
0.40824807675615	0.40824829028643
0.40824817659398	0.40824829043852
0.40824837291689	0.40824829059060
0.40824853552591	0.40824829226353
0.40824873892687	0.40824828906976

Table 3.1: Components of the dominant eigenvector

Instead, if we begin with an initial guess close to $u \equiv 0$, the nonlinear power method gives 9.9810649 as the approximate dominant eigenvalue (and the MATLAB calculation gives 10). We conclude that $u \equiv 0$ is an unstable equilibrium.

3.2.2 Inverse iteration with shifting

In §3.2.1, we characterize the stability of a hyperbolic equilibrium point u_0 for the nonlinear system (3.1.1). We did so by using Broyden's method to compute the solution u_0 such that $f(u_0) = 0$, and then determined the magnitude and sign of the smallest eigenvalue of the Jacobian $Df(u_0)$ via

inverse iteration. That is, we compute the largest eigenvalue of $Df(u_0)^{-1}$ in a matrix–free fashion by repeatedly performing Broyden's method on a set of directed perturbations to u_0. In this way, we emulate the action of $Df(u_0)^{-1}$ on a sequence of vectors whose norms tended to zero. The results in chapter 2 indicate that this process converges to the dominant eigenpair of $Df(u_0)^{-1}$. This example arose from the semi–discretization of a parabolic problem, so the ordering of the eigenvalues ensures that determining the sign of the smallest eigenvalue in magnitude for the Jacobian matrix $Df(u_0)$ is sufficient to determine the stability of the system.

We can generalize this to a routine that emulates the inverse power method with shifting. In this situation, we want the function differences to emulate the action of $[Df(u_0) - \rho I]^{-1}$ on a sequence of vectors, where ρ is a constant and the diagonal matrix ρI shifts the eigenvalues of $Df(u_0)^{-1}$ by the amount ρ. Unlike the matrix case, where the difference between inverse iteration with and without shifts is simply a matter of adding a multiple of the identity and solving a sequence of linear systems, in the nonlinear context the alteration is more involved. Shifting the eigenvalues of the Jacobian requires altering the function, so we need to make the alterations in a way that they do not affect the problem.

The system (3.1.1) of ODE's and the relationships (3.2.4) and (3.2.5) made it possible to compute information about $Df(u_0)^{-1}$ by performing the differences $g(0+d) - g(0)$. In order to correctly shift the eigenvalues of $Df(u_0)$, given a scalar ρ, we define

$$\tilde{f}(u) := f(u) - \rho u, \tag{3.2.11}$$

so that

$$D\tilde{f}(u) = Df(u) - \rho I. \tag{3.2.12}$$

Suppose that $\tilde{f}(u)$ is invertible with inverse $\tilde{g}(u)$. We can proceed as in §3.2.1 to derive a formula for the derivative of $\tilde{g}(u)$. By differentiating both sides of the identity

$$u = \tilde{g}(\tilde{f}(u)), \tag{3.2.13}$$

we get

$$\begin{aligned} I &= D[\tilde{g}(\tilde{f}(u))] \\ &= D\tilde{g}(\tilde{f}(u))D\tilde{f}(u) \\ &= D\tilde{g}(f(u) - \rho u)(Df(u) - \rho I). \end{aligned} \tag{3.2.14}$$

Therefore,

$$D\tilde{g}(f(u) - \rho u) = (Df(u) - \rho I)^{-1}. \tag{3.2.15}$$

Now, if u_0 is an equilibrium point for $f(u)$, then

$$D\tilde{g}(-\rho u_0) = (Df(u_0) - \rho I)^{-1}. \tag{3.2.16}$$

There are two key differences between the unshifted and shifted cases. In the unshifted case, we emulated the action of $Df(u_0)^{-1}$ by performing the finite differences $g(0+d) - g(0) \approx Dg(0) = Df(u_0)^{-1}$. Here, we emulate the action of the shifted Jacobian $[Df(u_0) - \rho I]^{-1}$ by emulating the action of the Jacobian $D\tilde{g}(-\rho u_0)$. That is, we perform the finite differences

$$\tilde{g}(-\rho u_0 + d) - \tilde{g}(-\rho u_0) \approx D\tilde{g}(-\rho u_0)d = [Df(u_0) - \rho I]^{-1}d. \tag{3.2.17}$$

Note that the point about which the linearization occurs is $-\rho u_0$, where u_0 is the equilibrium point for $f(u)$, and we use $\tilde{g}$ for function evaluations instead of g. This amounts to changing the code so that Broyden's method

solves $f(u) - \rho u = d$, instead of $f(u) = d$. However, the equilibrium point u_0 solves $\tilde{g}(-\rho u_0)$, since $\tilde{g}(-\rho u_0)$ means "solve $\tilde{f}(u) = -\rho u_0$ for u":

$$\begin{aligned} \tilde{f}(u) &= -\rho u_0 \\ \Rightarrow f(u) - \rho u &= -\rho u_0, \end{aligned} \tag{3.2.18}$$

and (3.2.18) is satisfied at u_0 since $f(u_0) = 0$. Thus $\tilde{g}(-\rho u_0)$ is the same as $f(u) = 0$ and needs only to be computed once. The quantities $\tilde{g}(-\rho u_0 + d)$ are computed using Broyden's method with $\tilde{f}$ in place of f.

Example 3.2.2. Adding a spectral shift to the inverse power method can be useful in cases where the eigenvalues do not have the same sign. We repeat the computations in example (3.2.1), except that we let the parameter $\xi = -0.1$. In this case, $u \equiv 1$ is still a solution, but the eigenvalues no longer satisfy the order property discussed in §3.2.1. In fact, using MATLAB to compute an eigen-decomposition of the centered difference approximation to the Jacobian, we find that the eigenvalues are

$$-14.7168, -1.2500, -0.5555, -0.3571, -0.2831 \text{ and } 5.0000.$$

In this case, the eigenvalue 5 corresponds to the eigenvalue $\lambda = \frac{1}{5}$ for the linearized system $f'(u_0)$, so $u_0 \equiv 1$ is a hyperbolic fixed point. Upon performing the NLPM in this example, the algorithm returns $\lambda = -14.7169$ after 26 iterations. This is an approximation to the dominant eigenvalue of the inverse to the linearized system, and corresponds to the eigenvalue -0.06794 for $f'(u_0)$. Unfortunately, this says nothing about the stability of the linearized system. We alter the problem slightly by introducing the shift $\rho = 1$ and performing the NLPM on the shifted problem. Now, MATLAB computes these eigenvalues:

$$-1.2500, -0.9364, -0.5556, -0.3571, -0.2632, \text{ and } -0.2207.$$

After 26 iterations, the algorithm converges to -1.2500, which corresponds to the eigenvalue -0.8 for the shifted linearized system. Upon adding back the shift $\rho = 1$, we find that the power method converges to the eigenvalue corresponding to the positive eigenvalue $\lambda = 0.2$ and we can correctly conclude that the linearization is unstable.

Remark 3.2.1. It is interesting to analyze the reason that the shift $\rho = 1$ causes the NLPM to converge to the correct eigenvalue of the linearized system when the magnitudes of the two largest eigenvalues of the matrix are -14.7168 and 5, respectively. The reason is that the shift affects the spectrum of the inverse (see equation (3.2.16)), which in this case is

$$-3.5323, -2.8003, -1.8002, -0.8000, -0.0679, \text{ and } 0.2000,$$

so the shifted spectrum is

$$-4.5323, -3.8003, -2.8002, -1.8000, -1.0679, \text{ and } -0.8000.$$

The reciprocals are

$$-0.2206, -0.2631, -0.3571, -0.5556, -0.9364, \text{ and } -1.2500.$$

Thus, the shift $\rho = 1$ causes the eigenvalue corresponding to the positive eigenvalue in the linearized system to become dominant. Of course, not every choice of ρ has this effect, and other information about the system may be needed in order to make an appropriate choice for a shift ρ.

Remark 3.2.2. In both of the examples presented in this chapter, we ran the power method for 26 iterations. This number was chosen because we set the algorithm to divide the norms of successive iterates by 2 in the linearization process. 26 is then the number of iterations required to reduce the perturbation to $\sqrt{\epsilon_0}$, where ϵ_0 is the machine number.

Remark 3.2.3. The computer programs used to generate the results in examples (3.2.1) and (3.2.2) can be found in appendix A.

Chapter 4

APPLICATION TO *A-POSTERIORI* ERROR ESTIMATION

Recent approaches to computational error estimation based on duality [42, 47, 48, 49, 50, 52] provide useful methods for determining information about the error of a numerical solution to a nonlinear problem. The error estimates themselves rely on the solution of a *dual problem*, which must be linearized around the approximate solution since the coefficients of the "correct" dual problem require the true (and unknown) solution. Various norms of the dual solution are called *stability factors*, and are important components of reliable *a-posteriori* bounds on the error in the desired functional information. The effect of the linearization on these bounds is unclear, and this "error of linearization" is the starting point for the ideas in this thesis.

The original idea for investigating the sensitivity of the error estimates on linearization was to implement a NLPM to produce a perturbation to the approximate solution that caused the largest change in the dual solution, thus giving some indication of the magnitude and direction of the worst possible effect of linearization on the error bound. The operator we investigate in this context is the map from the approximate finite element solution around which we linearize, to the corresponding solution of the

dual problem. The hope was that this could be done in a matrix–free manner, since the idea was to apply this to problems of very large dimension, i.e. discretizations of elliptic problems. However, the power method cannot accomplish this goal unless the derivative of the operator under consideration is normal. Unfortunately, the map from the finite element solution to the dual solution turns out to be generally nonnormal. By performing the NLPM alternately with the action of the derivative and its transpose, we can emulate a singular value decomposition that does converge to the desired direction of largest change. However, emulating the action of the transpose requires numerically reconstructing the matrix, so our procedure is very memory intensive.

The outline for this chapter is as follows: First, we compare and contrast *a-priori* and *a-posteriori* error estimates, and review some recent developments in *a-posteriori* estimation, primarily for finite element computations. We give more details about the specific type of *a-posteriori* estimators relevant to this thesis. Next, we discuss the general linearization issue in the context of two–point boundary value problems. Then, we define the map Φ between the approximate solution U to the forward problem and the dual solution ϕ that corresponds to linearizing around U. The remainder of the chapter is devoted to an examination of this map. We use the Green's function to derive an explicit formula for the action of the derivative. We make an important connection to an earlier paper of Rheinboldt [85] on the conditioning of nonlinear problems, and end the chapter with some discussion of how the ideas are implemented numerically.

4.1 *A-priori* and *a-posteriori* error estimation

Solving partial differential equations with a computer generally involves computing an approximation to the true solution of the problem. Quantifying the error involved in the approximation is an important part of the process, as "accurate numerical approximations may give meaningful information about exact solutions, while completely inaccurate numerical approximations do not contain meaningful information" [48].

Estimates of the error which are computed before the numerical calculation is performed are called *a-priori* estimates, and estimates that utilize a particular computed solution (and are thus computed after the approximation has been made) are called *a-posteriori* estimates. In [42], the authors say the following about the two types of error estimates:

> Traditionally, the error analysis of numerical methods for differential equations has been of *a-priori* type with a dependence on the unknown exact solution and appears to require a 'complete understanding' of the problem in order to be useful from a quantitative point of view. In the *a-posteriori* error analysis the 'understanding' is replaced by computing, which opens the possibility of quantitative error control for complex problems. In the spirit of synthesis, we conclude that *a-priori* and *a-posteriori* analysis serve different purposes and both are fundamental.

A-priori estimates are useful for proving that a numerical scheme converges to the exact solution as the discretization parameter gets small, or that a numerical method produces the best possible approximation to the true solution in some norm. In [47] for example, Estep studies the initial value problem

$$\begin{cases} \dot{y} + f(y,t) = 0, & 0 < t \leq T,) \\ y(0) = y_0, \end{cases} \tag{4.1.1}$$

in $\mathbb{R}^d$, $d \geq 1$. The piecewise constant discontinuous Galerkin (dG) scheme is given by

$$Y(t_n) + \int_{t_{n-1}}^{t_n} f(Y(t_n), s)\, ds = Y(t_{n-1}), \tag{4.1.2}$$

where Y is a piecewise constant approximant to the solution y on each interval $[t_{n-1}, t_n]$. An *a-priori* analysis shows that the dG approximation converges as quickly as any other approximation of y in the finite element space, and an optimal order error bound is given by

$$\|Y(t_n) - y(t_n)\| \leq \tilde{S}(t_n) \max_{m \leq n} k_m \max_{[t_{m-1}, t_m]} \left\| \frac{dy}{dt} \right\|, \tag{4.1.3}$$

where k_m is the m^{th} time step. Here, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$, and $\tilde{S}(t_n)$ is a constant determined by the accumulation of error over the interval of computation. A typical bound for $\tilde{S}(t_n)$ is given by

$$\tilde{S}(t_n) \leq \frac{e^{Lt_n} - 1}{L}. \tag{4.1.4}$$

Such estimates typically result from the application of Gronwall's inequality, and account for the worst possible rate of error growth. Note also that the bound depends on derivatives of the unknown function $y(t)$. The bound (4.1.3) is sufficient to prove convergence, but the constant L may be quite large in general, and therefore (4.1.3) may only be roughly the same size as the true error $\|Y_n - y(t_n)\|$ for a short transient period, so (4.1.3) may grossly overestimate the true error of the calculation. For this reason, *a-priori* estimates are typically unsuitable for garnering information about how to control the error of an approximation. For this, we use *a-posteriori* estimates.

A-posteriori estimates provide bounds that involve the computed solution itself. These error bounds are given in terms of computable quantities,

and are well suited for adaptive error control methods, meaning the estimate is used to make some improvement to an iterative scheme (e.g. adaptively refining the mesh in a sequence of finite element calculations) with a view to reducing the error and limiting computational expense. In contrast to the *a-priori* estimate (4.1.3), Estep derives the *a-posteriori* estimate

$$\|Y(t_n) - y(t_n)\| \leq S(t_n) \max_{m<n} k_m \frac{Y(t_m) - Y(t_{m-1})}{k_m} \tag{4.1.5}$$

for the dG finite element method for an autonomous problem. Here the bound does not depend on unknown derivatives of y, but it does contain the term $S(t_n)$, which is again a constant that measures the accumulation of error. The author shows how to avoid using overly–large bounds on $S(t_n)$ by deriving an *a-posteriori* approximation to $S(t_n)$ that converges to $S(t_n)$ as Y converges to y, and this provides the means to estimate the accumulation of error in a particular computation. Thus the *a-posteriori* bound can be used to govern the adaptive aspects of a particular numerical calculation by providing local bounds that are roughly the same size as the error in order to indicate which regions of the domain require refinement in order to lower the error in those regions without unnecessarily refining in regions where the error is small.

4.2 Review of *a-posteriori* results

Research in the area of *a-posteriori* error estimation is relatively new, spawned primarily by the success of the finite element method. Many credit its beginning with the seminal paper [7] of Babuška and Rheinboldt in 1978, in which the authors present various *a-posteriori* estimators for finite element solutions to ordinary differential equations. Their paper presents

the basic error analysis for linear, self–adjoint, elliptic two–point boundary value problems of the form

$$\begin{cases} L(u) = -\dfrac{d}{dx}\left(a(x)\dfrac{du}{dx}\right) + b(x)u = f, \ x \in (0,1) \\ u(0) = u(1) = 0, \end{cases} \tag{4.2.1}$$

as well as the eigenvalue problem

$$\begin{cases} L(u) = -\dfrac{d}{dx}\left(a(x)\dfrac{du}{dx}\right) + b(x)u = \lambda u, \\ u(0) = u(1) = 0. \end{cases} \tag{4.2.2}$$

They also present results for the parabolic problem

$$-\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right) + b(x)u = f(x,t), \qquad \forall x \in I \tag{4.2.3}$$

with the boundary conditions

$$\begin{cases} u(0,x) = 0, \quad x \in \bar{I}, \\ u(t,0) = u(t,1) = 0, \quad t \geq 0. \end{cases} \tag{4.2.4}$$

The authors derive computable *a-posteriori* estimates for each type of problem in an asymptotic form for $h \to 0$ where h measures the size of the elements. These ideas are extended and more fully analyzed in the context of adaptive mesh refinement in several later papers by the same authors [8, 9, 10, 86]. Since then, much work has been done in the field, particularly as it relates to adaptive techniques for the finite element method. See [2, 11, 14, 15, 19, 26, 27, 28, 37, 43, 44, 45, 51, 67, 75, 78, 115].

In 1994, Verfürth [106] classifies *a-posteriori* error estimates within the framework of finite element methods into four categories:

1. *Residual estimates*: Estimate the error of the computed numerical solution by a suitable norm of its residual with respect to the strong form of the equation.

2. *Solution of local problems*: Solve locally discrete problems similar to, but simpler than the original problem and use appropriate norms of the local solutions for error estimation.

3. *Sharp a-priori estimates*: Derive sharp *a-priori* estimates and use suitable higher order difference quotients of the computed numerical solution to estimate the higher order derivatives appearing in the *a-priori* error estimates.

4. *Averaging methods*: Use some local averaging technique for error estimation.

He provides references for research on each type of estimator in [106] and also [107]. In [106], the author presents a general framework for deriving residual based *a-posteriori* error estimates in the energy norm for approximate solutions to nonlinear problems. In this abstract approach, he considers nonlinear equations of the form

$$F(u) = 0 \tag{4.2.5}$$

and corresponding discretizations

$$F_h(u_h) = 0 \tag{4.2.6}$$

where $F \in C^1(X, Y^*), F_h \in C(X_h, Y_h^*), X_h \subset X$ and $Y_h \subset Y$ are finite dimensional subspaces of Banach spaces X and Y, with dual spaces X^* and Y^* respectively. A similar approach is taken in [12] for nonlinear problems

$$F(u) = 0 \quad \text{in } X^* \tag{4.2.7}$$

where X is a Hilbert space with dual space X^* and $F : X \to X^*$. Here, the authors prove upper and lower bounds for *a-posteriori* error estimates in the L_2 norm.

Later, Verfürth's book [107] provides a review of research in *a-posteriori* estimates, particularly as it applies to adaptive mesh refinement techniques. He uses Possion's equation with mixed Dirichlet and Neumann boundary conditions as a model problem and derives error estimators of the four types described above, and discusses their efficiency. He then reviews the theory for nonlinear equations and presents several finite element discretizations for various elliptic PDE. The book also contains nearly one hundred references to papers and textbooks on *a-posteriori* methods in computational error estimation. However, Verürth makes no mention of the duality/variational approach to *a-posteriori* error estimation that we discuss in this thesis.

Our approach to *a-posteriori* error estimation has its roots in the works of Johnson, Eriksson, Estep, Larson, Williams and Hansbo, in which estimations of the error are computed in terms of residuals and a variational analysis using duality. This approach is discussed in detail in the next sections. The basic ideas were first introduced for linear parabolic equations by Eriksson and Johnson [67], where it is possible to bound the stability factors *a-priori*. Estep first developed the idea of solving the adjoint problem to compute stability factors in the context of nonlinear ODE's [47], and later extended these ideas to PDE's. The monograph [52] contains a rigorous and comprehensive analysis of this method applied to systems of reaction–diffusion equations. It is in [52] and [47] that the issue of linearization error for *a-posteriori* estimates for nonlinear equations is first mentioned. More recent works involving *a-posteriori* error estimates based on duality and variational analysis can be found in [23] and [81]. Adjoint methods are also used in other contexts, such as parameter recovery for inverse problems [103].

4.3 *A-posteriori* estimates based on residuals and variational analysis

In this section, we discuss the approach to *a-posteriori* error analysis that we use in this paper. We begin with systems of algebraic equations, which provides a simple and familiar context in which to fix ideas. We then discuss these ideas in the context of numerical solutions for two-point boundary value problems.

4.3.1 An analogous approach in the solution of algebraic equations

Our approach to *a-posteriori* error analysis is relatively simple to explain in the context of the numerical solution of a system of algebraic equations. The problem here is to estimate the error of a numerical solution $X \approx x$ of a system of algebraic equations

$$F(x) = b \tag{4.3.1}$$

where the data b, nonlinearity F, and the solution x all have the same finite dimension. We assume that a numerical solution X of (4.3.1) has been computed in some fashion and we seek to estimate the unknown error $e = x - X$.

To start with, we consider the simpler case of a linear system of equations,

$$Ax = b. \tag{4.3.2}$$

The residual error of X is defined as

$$R := AX - b, \tag{4.3.3}$$

and is generally not zero. The point is to relate the error e to the computable residual error R. First, we can use the fact that the residual error of the true solution is zero to write

$$Ae = -R, \tag{4.3.4}$$

and then try to obtain an estimate of the error by solving this equation approximately in some fashion. This is not the approach that we use in this thesis but it is related to the classic method of estimating the error using high order asymptotic error estimates.

Instead, we settle for the less ambitious goal of obtaining an estimate on the size of a projection of the error. We introduce the dual problem

$$A^\top \phi = \psi, \tag{4.3.5}$$

where ψ is any unit vector. Using $\langle \cdot, \cdot \rangle$ to denote some inner product with corresponding norm $\| \cdot \|$, we get an exact representation of a projection of the error by computing the inner product

$$|\langle e, \psi \rangle| = |\langle e, A^\top \phi \rangle| = |\langle Ae, \phi \rangle| = |\langle R, \phi \rangle| \tag{4.3.6}$$

which leads directly to the error bound

$$|\langle e, \psi \rangle| \leq \|\phi\| \, \|R\|. \tag{4.3.7}$$

Since the residual R is computable, if we solve the dual problem numerically or obtain an estimate on the size of ϕ in some other way, then we obtain an estimate on the size of the projection of the error in the direction of the data for the dual problem. If we could be so fortunate to choose $\psi = \dfrac{e}{\|e\|}$ for example, then we obtain an estimate on $\|e\|$. To obtain an estimate on

the size of the first component of e, we choose $\psi = [1\,0\,0\,\cdots\,0]^\top$, whereas to obtain an estimate on the average of the components of the error, we choose $\psi = [1\,1\,\cdots\,1]^\top$.

The quantity $\|\phi\|$ is called the stability factor for this problem, and it is related to the condition number of A. In fact, it follows that

$$\left|\left\langle \frac{e}{\|x\|}, \psi \right\rangle\right| \leq \mathrm{cond}_\psi(A)\, \frac{\|R\|}{\|b\|}, \tag{4.3.8}$$

where $\mathrm{cond}_\psi(A) = \|\phi\|\,\|A\| = \|A^{-\top}\psi\|\,\|A\|$ can be interpreted as the condition number of the matrix A related to ψ. Hence the stability factor is a measure of the sensitivity of numerical solutions of the problem to computational errors.

Returning to the nonlinear problem (4.3.1), the residual error is now

$$R = F(X) - b, \tag{4.3.9}$$

which immediately gives

$$F(x) - F(X) = -R. \tag{4.3.10}$$

To obtain a linear equation for the error, we use the mean value theorem for integrals in the form

$$F(x) - F(X) = \int_0^1 F'(sx + (1-s)X)\,(x - X)\,ds \tag{4.3.11}$$

where F' is the Jacobian matrix of F. Applying this to the last relation, we get

$$\tilde{A}e = -R \tag{4.3.12}$$

with

$$\tilde{A} = \int_0^1 F'(sx + (1-s)X)\,ds, \tag{4.3.13}$$

which is the matrix obtained by averaging F' around points on the line joining x and X. We now follow the variational analysis for the linear problem outlined above to obtain an estimate on a projection of the error in terms of the residual and the stability factor corresponding to the linear dual problem defined using $\tilde{A}$.

4.3.2 Linearization error for nonlinear problems

The analysis in the §4.3.1 applies when $F(x) = b$ is an ordinary or partial differential equation, which may be nonlinear. If we denote the true and approximate solutions by u and U, respectively, then we have

$$\begin{aligned} -R &= F(u) - F(U) \\ &= \left(\int_0^1 F'(U + s(u - U))\, ds \right) (u - U) \\ &\equiv Ae. \end{aligned} \tag{4.3.14}$$

It is clear from (4.3.14) that computing the error estimate requires knowledge of the true solution. An error of linearization is introduced when we make the replacement

$$A = \int_0^1 F'(U + s(u - U))\, ds \mapsto \tilde{A} = F'(U), \tag{4.3.15}$$

and form the dual problem $\tilde{A}^\top \tilde{\phi} = \psi$, which corresponds to linearizing around the approximate solution. While the disadvantage is that this is the solution to the wrong dual problem, the solution $\tilde{\phi}$ is computable, whereas the dual solution to $A^\top \phi = \psi$ is unknowable since A depends on u. In practice, the computational error estimates tend to be accurate on a wide variety of nonlinear problems. Ideally we would like to understand the true effect of this error of linearization. This is not possible since we do not know

the direction from the approximate solution to the true solution. However, we can study the effect on the dual solution of introducing a small perturbation to the approximate solution about which we are linearizing. The claim is that if we could find the perturbation that causes the largest possible change in $\tilde{\phi}$, we could then find bounds for the effect of the linearization error, provided that the dual solution varies smoothly as a function of the forward solution, and the true and computed solutions are sufficiently close.

4.3.3 Two–point boundary value problems

Let $u = u(x)$ and $U = U(x)$ denote the true and approximate solutions to the two–point boundary value problem

$$\begin{cases} -((a(u(x),x)u'(x))' + b(u(x),x)u'(x) = f(u(x),x), \\ u(0) = u(1) = 0. \end{cases} \tag{4.3.16}$$

For ease of notation, we suppress the dependence of the coefficients, data and solution on the independent variable, and write

$$-(a(u)u')' + b(u)u' = f(u), \tag{4.3.17}$$

where it is understood that $u = u(x)$, $a(u) = a(u(x),x)$, $b(u) = b(u(x),x)$, and $f(u) = f(u(x),x)$. The residual $R(U)$ is the quantity remaining when the approximate solution is substituted into the equation, and the true solution solves the equation exactly, so we have

$$-(a(U)U')' + b(U)U' - f(U) = R(U), \tag{4.3.18}$$

and

$$-(a(u)u')' + b(u)u' - f(u) = 0. \tag{4.3.19}$$

Subtracting these equations gives

$$-\{(a(u)u') - (a(U)U')\}' + \{b(u)u' - b(U)U'\} - \{f(u) - f(U)\} = -R(U). \quad (4.3.20)$$

Linearizing the terms on the left–hand side gives

$$-\left\{\int_0^1 \frac{d}{ds}\Big[a(su+(1-s)U)(su'+(1-s)U')\Big]\,ds\right\}' + \left\{\int_0^1 \frac{d}{ds}\Big[b(su+(1-s)U)(su'+(1-s)U')\Big]\,ds\right\} - \left\{\int_0^1 f'(su+(1-s)U)\,ds\right\}(u-U) = -R(U). \quad (4.3.21)$$

Now, equation (4.3.21) can be written as

$$-(\tilde{a}e' + \tilde{\alpha}e)' + (\tilde{b}e' + \tilde{\beta}e) - \tilde{f}e = -R(U), \quad (4.3.22)$$

where $e := u - U$ is the difference between the true and approximate solutions, and

$$\begin{aligned}
\tilde{a} &:= \int_0^1 a(su+(1-s)U)\,ds,\\
\tilde{\alpha} &:= \int_0^1 a'(su+(1-s)U)(su'+(1-s)U')\,ds,\\
\tilde{b} &:= \int_0^1 b(su+(1-s)U)\,ds,\\
\tilde{\beta} &:= \int_0^1 b'(su+(1-s)U)(su'+(1-s)U')\,ds,\\
\tilde{f} &:= \int_0^1 f'(su+(1-s)U)\,ds. \quad (4.3.23)
\end{aligned}$$

In the above, and in what follows, the prime symbol on any of the coefficients a, b or f denotes differentiation with respect to the first dependent variable, but not the second. In other words, we take $a'(u)$ to mean $\dfrac{da}{du}$, rather than

$\frac{da}{dx}$, and similarly for $b'(u)$ and $f'(u)$. However, u', U' and ϕ' still refer to differentiation with respect to x.

Note that

$$\begin{aligned}
\frac{d}{dx}[\tilde{a}e] &= \frac{d}{dx}\left[\int_0^1 a(su+(1-s)U)\,ds(u-U)\right] \\
&= \int_0^1 \frac{d}{dx}(a(su+(1-s)U)(u-U))\,ds \\
&= \int_0^1 \Big(a'(su+(1-s)U)(su'+(1-s)U')(u-U) \\
&\qquad + a(su+(1-s)U)(u'-U')\Big)\,ds \\
&= \tilde{a}e' + \tilde{\alpha}e, \qquad (4.3.24)
\end{aligned}$$

and similarly for $\frac{d}{dx}[\tilde{b}e]$, so we can write (4.3.22) as

$$-((\tilde{a}e)')' + (\tilde{b}e)' - \tilde{f}e = -R \qquad (R = R(U)). \qquad (4.3.25)$$

We can derive the exact dual problem for (4.3.16) from (4.3.25), and then obtain the dual problem corresponding to linearizing around the approximate solution U. To derive the dual problem, we multiply (4.3.25) by a function $\phi = \phi(x)$, with $\phi(0) = \phi(1) = 0$, and integrate by parts:

$$\begin{aligned}
\int_0^1 -R\phi\,dx &= \int_0^1 -((\tilde{a}e)')'\phi + (\tilde{b}e)'\phi - \tilde{f}e\phi\,dx \\
&= \int_0^1 (\tilde{a}e)'\phi' - (\tilde{b}e)\phi' - \tilde{f}e\phi\,dx \\
&= \int_0^1 \tilde{a}e\phi'' - (\tilde{b}e)\phi' - \tilde{f}e\phi\,dx \\
&= \int_0^1 (\tilde{a}\phi'' - \tilde{b}\phi' - \tilde{f}\phi)e\,dx \qquad (4.3.26)
\end{aligned}$$

Here,

$$\begin{cases} \tilde{a}\phi'' - \tilde{b}\phi' - \tilde{f}\phi = \psi, \\ \phi(0) = \phi(1) = 0 \end{cases} \qquad (4.3.27)$$

is the dual problem for the boundary value problem (4.3.16). Equation (4.3.27) should be solved in order to produce the appropriate projection of the error e in the direction of ψ as described in (4.3.6). Of course, the coefficients $\tilde{a}, \tilde{b}$, and $\tilde{f}$ depend on the unknown true solution u, so we cannot solve (4.3.27) for ϕ. Replacing u with U in (4.3.23) for $\tilde{a}, \tilde{b}$, and $\tilde{f}$ corresponds to linearizing around U, and yields the following approximate dual problem:

$$\begin{cases} a(U)\phi'' - b(U)\phi' - f'(U)\phi = \psi, \\ \phi(0) = \phi(1) = 0. \end{cases} \tag{4.3.28}$$

We solve (4.3.28) numerically using the finite element method, which requires expressing the problem in weak form. Equation (4.3.28) is equivalent to

$$\begin{cases} -\left(a(U)\phi'\right)' + \left(a'(U)U' - b(U)\right)\phi' - f'(U)\phi = \psi, \\ \phi(0) = \phi(1) = 0. \end{cases} \tag{4.3.29}$$

See (4.7.2) and the relevant codes in (B.2.2) for the methods we use to solve this problem.

4.3.4 An alternate route to linearization

So far, the route has been to discretize a given problem, then linearize. By utilizing the Gatêaux derivative, we show that the same formula for the linearized problem arises when we linearize prior to discretization. Upon writing (4.3.17) in weak form, the situation is as follows: find $u \in V$ such that

$$\int_0^1 a(u)u'v'\,dx + \int_0^1 b(u)u'v - \int_0^1 fv = 0. \tag{4.3.30}$$

This is a root equation for a nonlinear form $\langle\langle F(u), v\rangle\rangle = 0$, where $\langle\langle F(\cdot), \cdot\rangle\rangle : V \times V \to 0$. The Gatêaux derivative of $\langle\langle F(u), v\rangle\rangle$ is $\langle\langle F'(u)w, v\rangle\rangle$ acting on $V \times V \times V$. We need to compute

$$\lim_{\delta \to 0} \frac{1}{\delta} \left[\langle\langle F(u + \delta w), v\rangle\rangle - \langle\langle F(u), v\rangle\rangle\right]. \tag{4.3.31}$$

Using $\langle \cdot, \cdot \rangle$ to denote the L_2 inner product, (4.3.31) becomes

$$\lim_{\delta \to 0} \frac{1}{\delta} \Big\{ \langle a(u+\delta w)(u+\delta w)', v' \rangle + (b(u+\delta w)(u+\delta w)', v\rangle - \langle f(u+\delta w), v\rangle \\ - \Big(\langle a(u)u', v' \rangle + \langle b(u)u', v\rangle - \langle f(u), v\rangle \Big) \Big\}. \quad (4.3.32)$$

Collecting terms, we get

$$\lim_{\delta \to 0} \frac{1}{\delta} \left[\langle\langle F(u+\delta w), v \rangle\rangle - \langle\langle F(u), v \rangle\rangle \right] = \\ \frac{1}{\delta} \big\langle (a(u+\delta w)u' - a(u)u'), v' \big\rangle + \frac{1}{\delta} \big\langle a(u+\delta w)\delta w', v' \big\rangle \\ + \frac{1}{\delta} \big\langle (b(u+\delta w)u' - b(u)u'), v \big\rangle + \frac{1}{\delta} \big\langle (b(u+\delta w)\delta w', v \big\rangle \\ - \frac{1}{\delta} \big\langle (f(u+\delta w) - f(u), v) \big\rangle. \quad (4.3.33)$$

Now, using

$$\begin{aligned} a(u+\delta w) - a(u) &= \int_0^1 a'(u+\delta s w)\delta w \, ds, \\ b(u+\delta w) - b(u) &= \int_0^1 b'(u+\delta s w)\delta w \, ds, \\ f(u+\delta w) - f(u) &= \int_0^1 f'(u+\delta s w)\delta w \, ds, \end{aligned} \quad (4.3.34)$$

and assuming a, b and f are smooth enough to justify interchanging the limit and the integral, we can express the difference quotient as

$$\lim_{\delta \to 0} \frac{1}{\delta} \left[\langle\langle F(u+\delta w), v \rangle\rangle - \langle\langle F(u), v \rangle\rangle \right] = \\ \big\langle a'(u+\delta s w)u'w, v' \big\rangle + \big\langle a(u+\delta w)w', v' \big\rangle + \big\langle b'(u+\delta s w)u'w, v \big\rangle \\ + \big\langle b(u+\delta w)w', v \big\rangle - \big\langle f'(u+\delta s w)w \, ds, v \big\rangle. \quad (4.3.35)$$

Letting $\delta \to 0$, we get a formula for the linearization form:

$$\langle\langle F'(u)w, v \rangle\rangle = \langle a'(u)u'w, v' \rangle + \langle a(u)w', v' \rangle \\ + \langle b'(u)u'w, v \rangle + \langle b(u)w', v \rangle - \langle f'(u)w, v \rangle. \quad (4.3.36)$$

We compute the dual to the linearized form by exchanging w and v, and integrating by parts where appropriate. This gives

$$-\langle (a(u)w')', v\rangle + \langle a'(u)u'w', v\rangle - \langle b(u)w', v\rangle - \langle f'(u)w, v\rangle. \qquad (4.3.37)$$

This is the left–hand side of the weak form of the dual problem corresponding to linearization around u.

4.3.5 Some *a-posteriori* estimates

We use the expression (4.3.22) for the residual and the relation $\langle e, \psi\rangle = \langle -R, \phi\rangle$ to derive an *a-posteriori* estimate for the projection $\langle e, \psi\rangle$ in terms of the computed solution U to the forward problem, the dual solution ϕ, and the coefficients a, b, and f:

$$\begin{aligned}
\int_0^1 e\psi\, dx &= \int_0^1 \left\{ -(\tilde{a}e' + \tilde{\alpha}e)' + (\tilde{b}e' + \tilde{\beta}e) - \tilde{f}e \right\} \phi\, dx \\
&= \int_0^1 (\tilde{a}e' + \tilde{\alpha}e)\phi' + (\tilde{b}e' + \tilde{\beta}e)\phi - \tilde{f}e\phi\, dx \\
&= \int_0^1 \{a(u)u' - a(U)U'\}\,\phi' + \{b(u)u' - b(U)U'\}\,\phi \\
&\qquad - \{f(u) - f(U)\}\,\phi\, dx \\
&= \left\{ \int_0^1 a(u)u'\phi' + b(u)u'\phi - f(u)\phi\, dx \right\} \\
&\qquad + \left\{ \int_0^1 -a(U)U'\phi' - b(U)U'\phi + f(U)\phi\, dx \right\} \\
&= \int_0^1 -a(U)U'\phi' + \{f(U) - b(U)U'\}\phi\, dx. \qquad (4.3.38)
\end{aligned}$$

The term on the fifth line vanishes because u solves the forward problem exactly.

Since U is a Galerkin approximation to u, we can use the Galerkin orthogonality property of the residual [42] to write the estimate in terms of

the *dual weights* $\phi - \pi_h\phi$, where $\pi_h\phi$ is an interpolant of ϕ into the same space as U:

$$\int_0^1 e\psi\,dx = \int_0^1 -a(U)U'(\phi-\pi_h\phi)'+\{f(U)-b(U)U'\}(\phi-\pi_h\phi)\,dx. \quad (4.3.39)$$

This requires solving the dual problem using a higher order method than used for U. Performing integration by parts on the second order term and writing the integral as a sum of element contributions, we can express $\langle e, \psi\rangle$ in terms of the residual and the dual weights:

$$\begin{aligned}\int_0^1 e\psi\,dx &= \sum_{j=1}^{m+1}\int_{I_j} R(U)(\phi-\pi_h\phi)\,dx \\ &\quad + \sum_{j=1}^{m} a(U_j,x_j)(\phi(x_j)-\pi_h\phi(x_j))[U']_j. \quad (4.3.40)\end{aligned}$$

Here, $[U']_j$ denotes the jump in the derivative of U at the j^{th} node, in cases where U is not sufficiently smooth. We also account for the error due to quadratures in (4.3.40) by adding the quantities

$$\begin{aligned}&-\int_0^1 \{aU'(\pi_h\phi)' - \overline{aU'(\pi_h\phi)}\} \\ &\qquad -\int_0^1 \{bU'\pi_h\phi - \overline{bU'\pi_h\phi}\} \\ &\qquad\qquad +\int_0^1 \{f\pi_h\phi - \overline{f\pi_h\phi}\}. \quad (4.3.41)\end{aligned}$$

Note that Galerkin orthogonality does not enter these terms. In other words, the error of the finite element solution is determined by $\phi - \pi_h\phi$ and ϕ. The overbar indicates that some quadrature rule is used to evaluate that part of the integral, and these terms represent the error due to the quadrature. In practice, a very high order quadrature rule is used to evaluate the "unbarred" terms since we never evaluate the integrals exactly. The code we use in this paper uses a five point Gauss rule for the barred terms,

and a thirty point Gauss rule for the "exact" integrands. Using such accurate quadrature rules essentially renders the quadrature error negligible. In codes where the trapezoidal rule or Simpson's rule are used for quadrature, the effect of quadrature error is significant.

The *a-posteriori* estimates given above involve both the forward solution $U(x)$, the dual solution $\phi(x)$, and some of their derivatives. The major motivation for this thesis is an investigation of how replacing (4.3.27) with (4.3.28) affects these error estimates. We utilize the NLPM to produce perturbations to U that cause the largest change in ϕ, and hence on the estimates. In the next section we define a mapping between the approximate forward solution U and the dual solution corresponding to the linearized problem (4.3.28) and show that the operator norm of the map plays a role in bounding the effect of linearization.

4.4 The mapping $\Phi : U \rightarrow \phi$

In this section, we define a nonlinear map between the approximate forward solution U to (4.3.16) and the dual solution ϕ corresponding to linearization around U, as given by (4.3.28). We investigate some relationships between the norm of the derivative of Φ and the size of the changes in the stability factors for the error bounds discussed in the previous section.

4.4.1 The map Φ

From (4.3.15), the dual problem corresponding to linearizing around U is

$$\tilde{A}^{\top}\tilde{\phi} = \psi \Rightarrow F'(U)^{\top}\tilde{\phi} = \psi \Rightarrow \tilde{\phi} = F'(U)^{-\top}\psi. \tag{4.4.1}$$

Suppose that $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ so that $F'(U)$ denotes the Jacobian of F at U, which we assume is continuously invertible in a neighborhood of U. Given

dual data ψ, define the map $\Phi_\psi : \mathbb{R}^n \to \mathbb{R}^n$ by

$$\Phi_\psi(U) = F'(U)^{-\top}\psi \text{ for } U \in \mathbb{R}^n. \tag{4.4.2}$$

Thus if U is a computed solution to a nonlinear problem $F(u) = b$, then for fixed dual data ψ, Φ_ψ maps the approximate solution U to the dual solution $\tilde{\phi}$. Here, we assume that the solutions of the forward and dual problems are in the same space. When solving two point boundary value problems in practice, we solve the forward problem (4.3.16) in the space $V_h^{(1)}$ of piecewise linear polynomials, and the dual problem (4.3.29) in the space $V_h^{(2)}$ of piecewise quadratic polynomials. We use the fact that $V_h^{(1)} \subset V_h^{(2)}$ and, in this context, consider Φ_ψ as a mapping from $V_h^{(2)}$ to itself.

Now, consider the linearization

$$\begin{cases} \Phi_\psi(U + \epsilon w) = \Phi_\psi(U) + \Phi'_\psi(U)\epsilon w + o(\|\epsilon w\|), \\ \|w\| = 1,\ 0 < \epsilon \ll 1. \end{cases} \tag{4.4.3}$$

Note that $\Phi'_\psi(U)$ is the Jacobian of the function given by the inverse of the Jacobian of the transpose of F at U acting on ψ:

$$\Phi'_\psi(U) = \left([F'(U)^{-\top}]\psi\right)' \in L(\mathbb{R}^n, \mathbb{R}^n). \tag{4.4.4}$$

Thus,

$$\begin{aligned} \|\Phi_\psi(U + \epsilon w) - \Phi_\psi(U)\| &\leq \|\Phi'_\psi(U)\epsilon w\| + o(\epsilon) \\ &\leq \epsilon\,\sigma + o(\epsilon), \end{aligned} \tag{4.4.5}$$

where σ is the largest singular value of the Jacobian matrix $\Phi'_\psi(U)$. If this matrix is normal and $\|\cdot\|$ is the spectral norm, then $\|\Phi'_\psi(U)\| = |\lambda|$, where $|\lambda|$ is the spectral radius of $\Phi'_\psi(U)$. Since $\|w\| = 1$, in this case we have

$$\|\phi_{\epsilon w} - \tilde{\phi}\| \leq \epsilon\,|\lambda| + o(\epsilon). \tag{4.4.6}$$

Here, we have defined

$$\phi_{\epsilon w} := \Phi_\psi(U + \epsilon w), \tag{4.4.7}$$

so $\phi_{\epsilon w}$ is the dual solution corresponding to linearizing around $U + \epsilon w$. Therefore, we conclude that asymptotically as $\epsilon \to 0$ we have

$$\sup_{\substack{\phi=\Phi_\psi(U+\epsilon v) \\ \|v\|=1}} \|\phi - \tilde{\phi}\| \leq C\epsilon\,|\lambda|, \tag{4.4.8}$$

for some constant C. This result is *a-posteriori* in the sense that it does not rely upon any knowledge of the true solution u. However, it is *a-priori* in the sense that introducing the norms in (4.4.5) gives a worst case bound for the effects of perturbations to U. This in turn complicates the process for non–normal operators, since then the dominant eigenvalue does not give the norm. However, if $\Phi'_\psi(U)$ is normal, then (4.4.5) says that if we perturb U a little bit in any direction, the maximum change in the size of the dual solution is, under appropriate conditions, asymptotically bounded by the size of the dominant eigenvalue of the map Φ_ψ, and in general the dominant singular value gives a measure of the sensitivity to linearization around the approximate solution U:

$$\begin{aligned} |\langle R, \phi_{\epsilon w}\rangle| &\leq |\langle R, \tilde{\phi}\rangle + \langle R, \phi_{\epsilon w} - \tilde{\phi}\rangle| \\ &\leq |\langle R, \tilde{\phi}\rangle| + \|R\|\,\|\phi_{\epsilon w} - \tilde{\phi}\| \\ &\leq |\langle R, \tilde{\phi}\rangle| + \epsilon\|R\|\,\sigma + o(\epsilon). \end{aligned} \tag{4.4.9}$$

If we knew that $\Phi'_\psi(U)$ was normal, we could use the nonlinear power method to compute the dominant eigenvalue by iteratively computing the differences $\Phi_\psi(U + \epsilon w) - \Phi_\psi(U)$. However, as we show below, normality of the map $\Phi'_\psi(U)$ is highly unlikely, except in some very simple cases.

We derive a formula for $\Phi'_\psi(U)$. First, write the dual problem as

$$F'(U)^\top \Phi_\psi(U) = \psi. \tag{4.4.10}$$

To simplify notation, suppress dependence on U and omit the subscript on Φ and simply write $F'\Phi = \psi$. Now differentiate:

$$(F'^{\top}\Phi)' = [F'^{\top}]'\Phi + F'^{\top}\Phi' = 0. \tag{4.4.11}$$

Note that $[F'^{\top}]'$ is not the same as the second derivative of the vector valued function F, due to the transpose between the derivatives. Still, it is identified with a bilinear map on the input space. Since Φ is a vector–valued function, the product $[F'^{\top}]'\Phi$ corresponds to the bilinear map with one argument fixed, hence is a linear operator. The second term in the sum is the product of two Jacobian matrices, so (4.4.11) is the sum of two matrices. It follows that

$$\begin{aligned} \Phi' &= -(F')^{-\top}[F'^{\top}]'\Phi. \\ &= -(F')^{-\top}\Big([F'^{\top}]' \circ (F')^{-\top}\Big)\psi. \end{aligned} \tag{4.4.12}$$

In other words, the derivative $\Phi'_{\psi}(U)$ is the product of two matrices: the negative inverse transpose of the Jacobian of the forward operator, and the linear operator obtained by the composing the derivative of the transpose of the Jacobian of F with the inverse transpose of the Jacobian of F acting on the dual data ψ. There are not any obvious reasons why this product should produce a normal matrix, and in fact probably almost never does. The following simple 2×2 example shows some of the difficulties that can arise in trying to analyze the function associated with linearization error.

Example 4.4.1. Consider the system of nonlinear equations $f(\vec{x}) = 0$, where $\vec{x} = [x, y]^{\top} \in \mathbb{R}^2$, defined by

$$f(\vec{x}) := \begin{bmatrix} xy - 1 \\ \dfrac{x^2 - 1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{4.4.13}$$

which has solutions $\vec{x}_1 = [1,1]^\top$ and $\vec{x}_2 = [-1,-1]^\top$. The Jacobian (denoted by a "prime" symbol) and its inverse are given by

$$f' = \begin{bmatrix} y & x \\ x & 0 \end{bmatrix} \quad \text{and} \quad (f')^{-1} = \begin{bmatrix} 0 & \frac{1}{x} \\ \frac{1}{x} & -\frac{y}{x^2} \end{bmatrix}. \tag{4.4.14}$$

Now, fixing dual data $\vec{\psi} = [\psi_1, \psi_2]^\top$, the map $\Phi_\psi(\vec{x})$ is given by

$$\Phi_\psi(\vec{x}) = f'(\vec{x})^{-\top}\psi = \begin{bmatrix} 0 & \frac{1}{x} \\ \frac{1}{x} & -\frac{y}{x^2} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} \frac{\psi_2}{x} \\ \frac{\psi_1}{x} - \frac{y\psi_2}{x^2} \end{bmatrix}, \tag{4.4.15}$$

and the Jacobian matrix is

$$\Phi'_\psi(\vec{x}) = \begin{bmatrix} -\frac{\psi_2}{x^2} & 0 \\ \frac{-x\psi_1 + 2y\psi_2}{x^3} & -\frac{\psi_2}{x^2} \end{bmatrix}. \tag{4.4.16}$$

Note that this matrix has the form $\begin{bmatrix} a & 0 \\ b & a \end{bmatrix}$, so it is not normal (unless $b = 0$), and it has a defective eigenvalue. On the other hand, by simply switching the places of the component functions for f, we get

$$\tilde{f}(\vec{x}) := \begin{bmatrix} \frac{x^2 - 1}{2} \\ xy - 1 \end{bmatrix} \Rightarrow \tilde{\Phi}'_\psi(\vec{x}) = \begin{bmatrix} \frac{-x\psi_1 + 2y\psi_2}{x^3} & \frac{-\psi_2}{x^2} \\ \frac{-\psi_2}{x^2} & 0 \end{bmatrix}. \tag{4.4.17}$$

This matrix is symmetric (hence normal) and so the dominant eigenvalue gives the norm. The size of the eigenvalue clearly depends on the components of the approximate solution, as well as the choice of dual data.

As a check, we compute the derivative for the function given in (4.4.13) using the formula (4.4.12). We have

$$(f')^\top = \begin{bmatrix} y & x \\ x & 0 \end{bmatrix}, \text{ and } -(f')^{-\top} = \begin{bmatrix} 0 & -\frac{1}{x} \\ -\frac{1}{x} & \frac{y}{x^2} \end{bmatrix}. \tag{4.4.18}$$

Now, $\left([f'^{\top}]' \circ (f')^{-\top}\right)\psi$ is computed by calculating the Jacobian matrix of each row of $(f')^{\top}$ and multiplying on the right by $[(f')^{-\top}\psi]^{\top}$:

$$\begin{aligned}\left([f'^{\top}]' \circ (f')^{-\top}\right)\psi &= \left[\frac{\psi_2}{x}, \quad \frac{\psi_1}{x} - \frac{y\psi_2}{x^2}\right]\frac{\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}}{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}} \\ &= \begin{bmatrix}\frac{\psi_1}{x} - \frac{y\psi_2}{x^2} & \frac{\psi_2}{x} \\ \frac{\psi_2}{x} & 0\end{bmatrix}\end{aligned} \tag{4.4.19}$$

Now, combining (4.4.18) and (4.4.19) gives

$$\Phi' = \begin{bmatrix}0 & -\frac{1}{x} \\ -\frac{1}{x} & \frac{y}{x^2}\end{bmatrix}\begin{bmatrix}\frac{\psi_1}{x} - \frac{y\psi_2}{x^2} & \frac{\psi_2}{x} \\ \frac{\psi_2}{x} & 0\end{bmatrix} = \begin{bmatrix}-\frac{\psi_2}{x^2} & 0 \\ \frac{-x\psi_1 + 2y\psi_2}{x^3} & -\frac{\psi_2}{x^2}\end{bmatrix}, \tag{4.4.20}$$

which agrees with the result of the direct calculation.

The derivative is normal if F is a diagonal map. In this situation, the i^{th} component function of the forward operator depends only on the i^{th} variable, so the Jacobian matrix is diagonal, and it follows that $-(F')^{-\top}$ is also diagonal. In addition, each matrix in the formation of $[F'^{\top}]'$ has zeros in all entries except the (i,i) position, which contains the second partial derivative of F with respect to the i^{th} variable. Therefore, regardless of the values of the components of ϕ, the matrix $[F'^{\top}]'\phi$ is diagonal. In this case, Φ'_{ψ} is the product of two diagonal matrices, so it is certainly normal. This is demonstrated in the next example.

Example 4.4.2. We consider the 2×2 nonlinear system $f(x) = 0$, where $f : \mathbb{R}^2 \to \mathbb{R}^2$ is given by

$$\begin{bmatrix}x_1 \\ x_2\end{bmatrix} \mapsto \begin{bmatrix}\frac{x_1^2}{2} - \frac{1}{2} \\ \frac{x_2^3}{3} - \frac{1}{3}\end{bmatrix}. \tag{4.4.21}$$

Note that f is a diagonal map. Now, suppose we have computed an approximate solution $X = [.95, .95]^\top$ to the true solution $x = [1, 1]^\top$. Suppose further that we choose dual data ψ in the direction $[1, 2]^\top$, which is normalized so that $\|\psi\| = 1$. The dual solutions corresponding to linearization around the approximate solution X and the true integral average $\int_0^1 f'(sx + (1-s)X\, ds$ are

$$\tilde{\phi} = \begin{bmatrix} 0.4707 \\ 0.9911 \end{bmatrix} \text{ and } \phi = \begin{bmatrix} 0.4587 \\ 0.9406 \end{bmatrix}. \tag{4.4.22}$$

The map $\Phi : X \to \phi$ and its derivative are given by

$$\Phi_\psi(X) = \begin{bmatrix} \dfrac{\psi_1}{x_1} \\ \dfrac{\psi_2}{x_2^2} \end{bmatrix} \text{ and } \Phi'_\psi(X) = \begin{bmatrix} -\dfrac{\psi_1}{x_1^2} & 0 \\ 0 & -\dfrac{2\psi_2}{x_2^3} \end{bmatrix}. \tag{4.4.23}$$

In figure 4.1 we evaluate the linearized dual solution $\Phi_\psi(X + \delta)$ for various perturbations δ. In this example, the true error is $\|e\| = \|x - X\| = 0.0707$, so we take $\|\delta\| = 0.0707$. Figure 4.1 shows the results of the NLPM finding the direction of largest change.

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0.95 \\ 0.95 \end{bmatrix}, \quad \|\delta\| = 0.0707$$

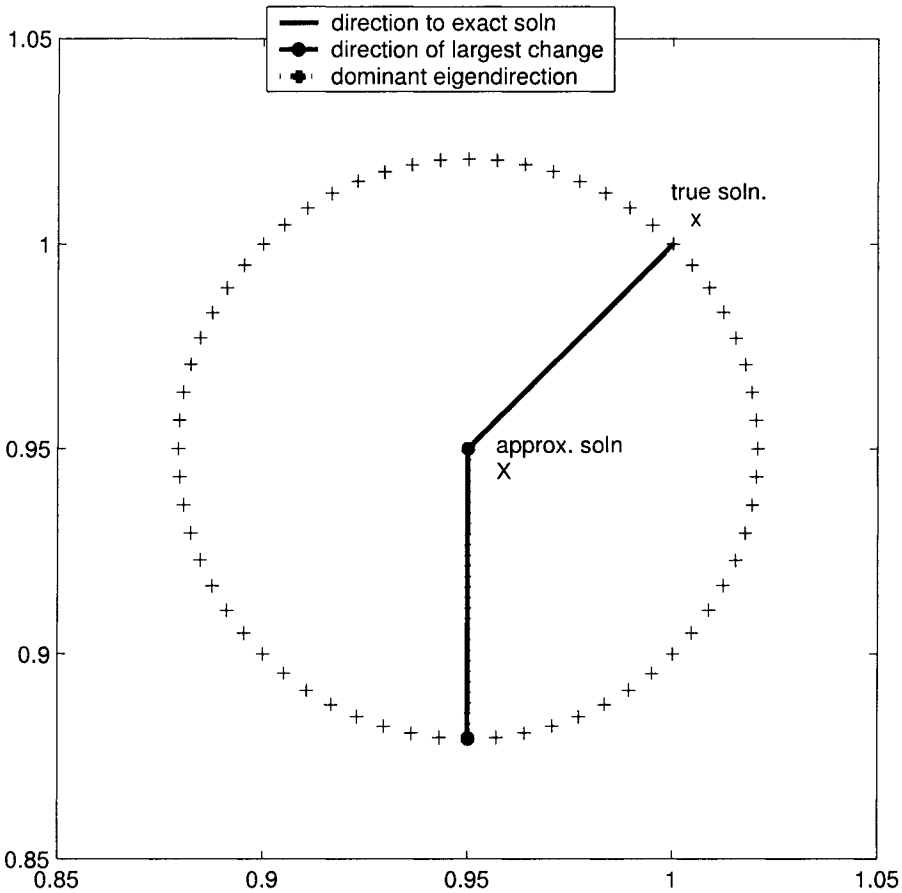

Figure 4.1: Results of the nonlinear power method.

In figure 4.2, we plot the size of the dual solution in the direction of each perturbation δ from X, and see that the NLPM has chosen the direction of the largest perturbation. The line labelled "true norm" in the plot indicates the size of the change in the dual solution corresponding to moving in the "correct" direction from $f'(X)$ to $\int_0^1 f'(sx + (1-s)X\, ds$. This value was computed using symbolic packages in MATLAB.

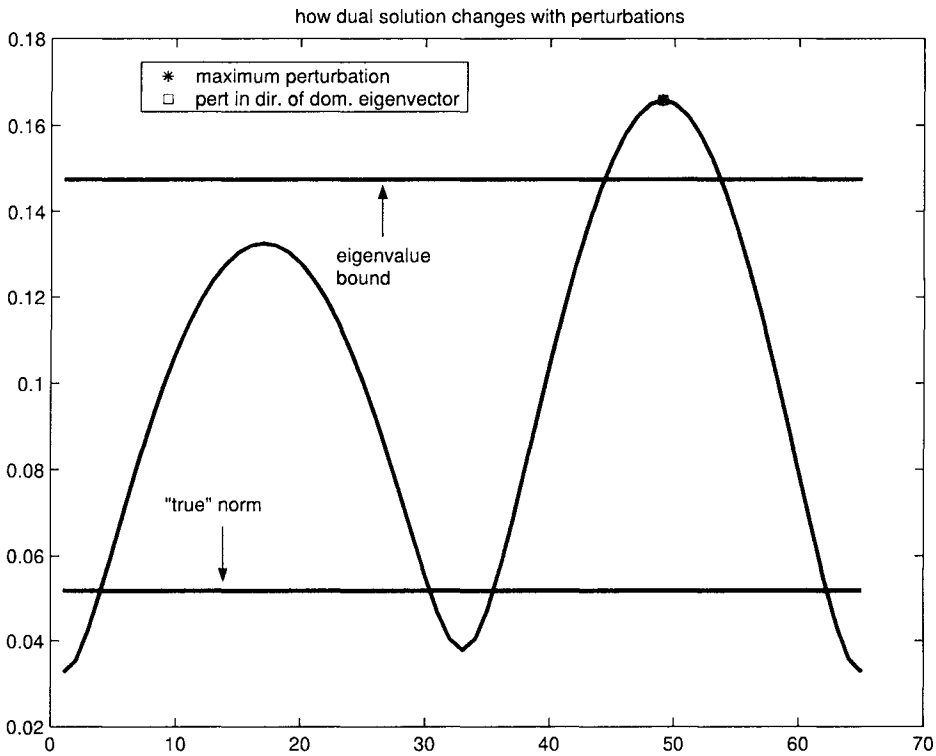

Figure 4.2: Changes in the size of dual solution.

Since we are able to compute all the information exactly in this problem, we can provide a full account of the error due to linearization. From the variational analysis, we know that the projection of the error e in the direction of the dual data ψ is given by

$$|\langle e, \psi \rangle| = |\langle R, \phi \rangle|, \tag{4.4.24}$$

yet what we compute in practice is $|\langle R, \tilde{\phi} \rangle|$ where $\tilde{\phi}$ is the dual solution corresponding to linearization around the approximate solution X. Now, for the computed value $X = [.95, .95]^\top$, the corresponding residual is

$$f\left(\begin{bmatrix} .95 \\ .95 \end{bmatrix} \right) = \begin{bmatrix} -0.0488 \\ -0.0475 \end{bmatrix}, \tag{4.4.25}$$

so using the dual solution $\tilde{\phi}$, we compute the projection of the error to be 0.0701. On the other hand, using the exact dual solution ϕ, we find that true projection of the error in the direction ψ is 0.0671, so that the error due to linearization is 0.0030. That is, linearization caused about a 4.5% change in the error estimate. Since the matrix $\Phi'_\psi(X)$ is normal, the norm is given by the magnitude of the dominant eigenvalue, which the NLPM computed to be 2.0864. So, from our analysis in chapter 4, we have

$$\begin{aligned} \|\Phi_\psi(X+\delta) - \Phi_\psi(X)\| &\leq \|\Phi'_\psi(X)\|\|\delta\| \\ &= (2.0864)(.0707) \\ &= .1475. \end{aligned} \tag{4.4.26}$$

Notice in figure 4.2 that there are perturbations to the approximate solution that produce changes larger than $\|\Phi'_\psi(X)\|\|\delta\|$, due to the higher order terms in the linearization. We repeat this experiment with an increasingly

accurate sequence of approximate solutions that approach the true solution $x = [1, 1]^\top$ along the line $y = x$:

$$X_1 = \begin{bmatrix} 0.9500 \\ 0.9500 \end{bmatrix}, \; X_2 = \begin{bmatrix} 0.9625 \\ 0.9625 \end{bmatrix}, \; X_3 = \begin{bmatrix} 0.9750 \\ 0.9750 \end{bmatrix}, \; X_4 = \begin{bmatrix} 0.9825 \\ 0.9825 \end{bmatrix}.$$

We show in figure 4.3 how the difference between $\|\Phi'_\psi(X)\|\|\delta\|$ and the computed maximum decreases as the size of the perturbations get smaller. Once again, the upper horizontal line is the quantity $\|\Phi'_\psi(X_j)\|$, and the lower horizontal line is the value corresponding to linearization around the theoretically correct average between x and X_j.

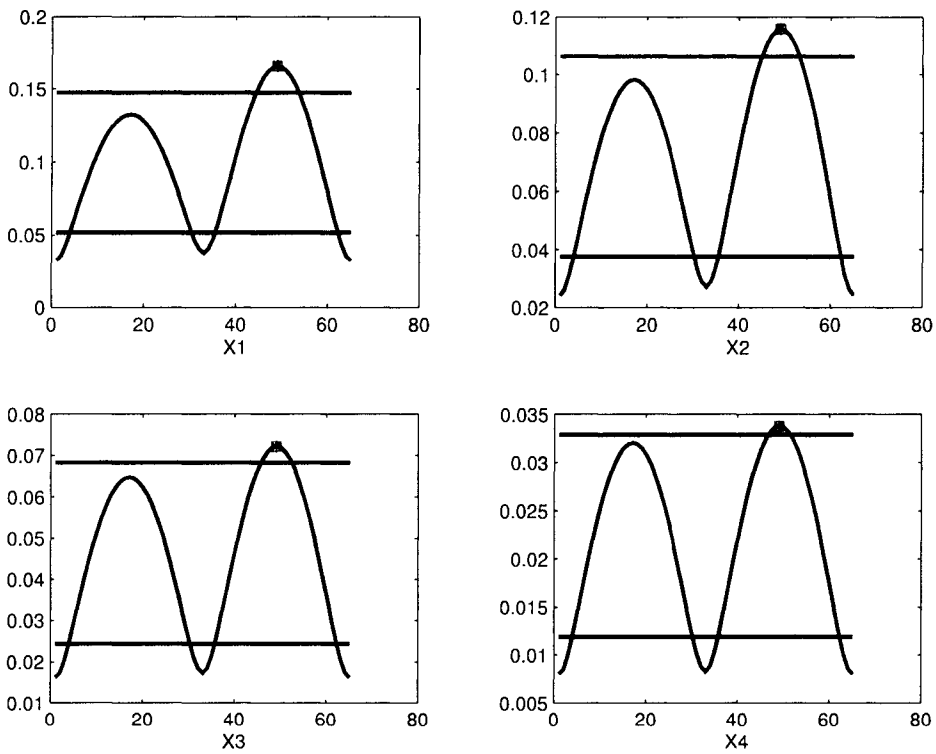

Figure 4.3: A sequence of increasingly accurate approximations.

Table 4.4.2 displays the quantities given by the bound 4.4.9. Here, ϕ_v is the dual solution corresponding to linearization in the direction of the dominant eigenvalue v, and $\tilde{\phi}$ is the dual solution corresponding to linearization around X_j. At each stage, we scale the perturbation so that $\|v\| = \|e\| = \|X_j - x\|$.

approx. soln	$\|e\|$	$\|\langle R, \phi_v \rangle\|$	$\|\langle R, \tilde{\phi} \rangle\|$	$\lambda \|R\| \|\delta\|$	$\|\langle R, \tilde{\phi} \rangle\| + \lambda \|R\| \|\delta\|$
X_1	.070711	0.077948	0.070065	0.010046	0.080112
X_2	.053033	0.056147	0.051962	0.005485	0.057447
X_3	.035355	0.036021	0.034263	0.002368	0.036630
X_4	.017677	0.017377	0.016948	0.000575	0.017523

Table 4.1: Quantities from the bound (4.4.9)

In order to utilize a matrix–free NLPM for computing the dominant singular value of $\Phi'_\psi(U)$, we need a matrix–free way to compute the action of $\Phi'_\psi(U)^\top$, then we can proceed by alternately computing function evaluations and transposes, which is equivalent to performing the power method on $\Phi'_\psi(U)^\top \Phi'_\psi(U)$. The square root of the dominant eigenvalue then gives the dominant singular value for $\Phi'_\psi(U)$. Given a function f, the situation is as follows: We implement the NLPM by using the linearization relationship

$$f(x+d) - f(x) = f'(x)d + o(d). \tag{4.4.27}$$

In order to adapt this to a singular value calculation based only on function evaluations, we need to emulate the iteration

$$d \mapsto f'(x)d \mapsto f'(x)^\top [f'(x)d] \mapsto f'(x) \left[f'(x)^\top [f'(x)d]\right] \dots . \tag{4.4.28}$$

Beginning with a random d, the first step is to compute

$$f(x+d) - f(x) := d_{new}. \tag{4.4.29}$$

For step two, we need the function g such that

$$g(x + d_{new}) - g(x) = f'(x)^\top d_{new}, \tag{4.4.30}$$

then we rescale and repeat. However, the following 2×2 example shows that it is not possible to find the required function g, unless the Jacobian f' is symmetric, in which case singular value methods are irrelevant.

Example 4.4.3. If $f : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is given by

$$f(x) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}, \tag{4.4.31}$$

then the transpose of the Jacobian matrix is

$$f'(x)^\top = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_2}{\partial x} \\ \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_2}{\partial y} \end{bmatrix}. \tag{4.4.32}$$

Now, the function $g : \mathbb{R}^2 \mapsto \mathbb{R}^2$ that is needed for function evaluations in the algorithm must satisfy

$$g(x + v) - g(x) = f'(x)^\top v \tag{4.4.33}$$

So if $g = \begin{bmatrix} g_1(x, y) \\ g_2(x, y) \end{bmatrix}$, the following relations must hold:

$$\frac{\partial f_1}{\partial x} = \frac{\partial g_1}{\partial x}, \quad \frac{\partial f_2}{\partial x} = \frac{\partial g_1}{\partial y}, \quad \frac{\partial f_1}{\partial y} = \frac{\partial g_2}{\partial x}, \quad \frac{\partial f_2}{\partial y} = \frac{\partial g_2}{\partial y}. \tag{4.4.34}$$

Now

$$\frac{\partial f_1}{\partial x} = \frac{\partial g_1}{\partial x} \Rightarrow f_1(x, y) = g_1(x, y) + h_1(y), \tag{4.4.35}$$

and

$$\frac{\partial f_2}{\partial y} = \frac{\partial g_2}{\partial y} \Rightarrow f_2(x, y) = g_2(x, y) + h_2(x). \tag{4.4.36}$$

Differentiating, we get

$$\begin{aligned} \frac{\partial f_1}{\partial y} &= \frac{\partial g_1}{\partial y} + h_1'(y), \\ \frac{\partial f_2}{\partial x} &= \frac{\partial g_2}{\partial x} + h_2'(x). \end{aligned} \tag{4.4.37}$$

Since $\frac{\partial g_1}{\partial y} = \frac{\partial f_2}{\partial x}$, substituting into the first equation in (4.4.37) gives

$$\frac{\partial f_1}{\partial y} = \frac{\partial f_2}{\partial x} + h_1'(y). \tag{4.4.38}$$

Using the second relation in (4.4.34) and substituting for $\frac{\partial f_2}{\partial x}$ gives

$$\frac{\partial f_1}{\partial y} = \frac{\partial g_2}{\partial x} + h_2'(x) + h_1'(y). \tag{4.4.39}$$

But $\frac{\partial g_2}{\partial x} = \frac{\partial f_1}{\partial y}$, so

$$-h_1'(y) = h_2'(x) \Rightarrow h_1 = h_2 = C. \tag{4.4.40}$$

Therefore, the function g can only differ from f by a constant, so that f must have a symmetric Jacobian. The extension of this example to n dimensions is straightforward, albeit tedious.

Unfortunately, calculation of the transpose requires the full matrix. We can still use the power method in the form of algorithm (2.4.2) to compute the dominant singular value by approximating each column of the matrix by perturbing the approximate solution U on basis vectors. This method is fine for problems of reasonably low dimension, but becomes a significant issue when the dimension is large. There appears to be no alternative, however.

4.4.2 A difficulty

A difficulty with using the result (4.4.8) to judge the effect of linearization error on the error estimate for the original problem is that we cannot guarantee that the solution to the true dual problem can be expressed as $\Phi_\psi(U + \epsilon w)$, so we cannot use this result directly. The reason is essentially the failure of the mean value theorem in higher dimensions [79]: for a general function $F : \mathbb{R}^n \to \mathbb{R}^n$ with component functions F_i, we have $F(u) - F(U) = B(u, U)(u - U)$, where

$$B(u,U) = \begin{pmatrix} \nabla F_1(U + s_1(u-U)) \\ \vdots \\ \nabla F_n(U + s_n(u-U)) \end{pmatrix}. \tag{4.4.41}$$

The $s_i \in [0,1]$ are distinct, hence B cannot in general be represented by $F'(U + \epsilon w)$. However, in the special case when F is a diagonal map, the Jacobian is a diagonal matrix and the mean value theorem does imply the existence of a point z in a $\|u - U\|$ neighborhood of U such that

$$\begin{aligned} F(u) - F(U) &= \int_0^1 F'(U + s(u-U))\, ds(u-U) \\ &= F'(z)(u-U), \end{aligned} \tag{4.4.42}$$

and the result (4.4.8) applies. In the application to *a-posteriori* error estimation, the code producing the dual solution ϕ given forward solution U is basically a "black box" in the sense that it cannot be explicitly represented as a multi-dimensional mapping with known component functions. We can investigate changes in outputs ϕ for perturbations to inputs U, but a numerical reconstruction of $B(u, U)$ in (4.4.41) is just not possible.

In the more general situation, we write

$$\|\bar{\phi} - \tilde{\phi}\| \leq \|\bar{\phi} - \phi_m\| + \|\phi_m - \tilde{\phi}\| \tag{4.4.43}$$

where ϕ_m is chosen to obtain a dual solution of the appropriate form, and to yield easy estimates for the terms on the right hand side. For example, we could take $\phi_m = \Phi_\psi(U + \frac{1}{2}(u - U))$, the dual solution corresponding to linearization about the midpoint. We can view this as an evaluation of A via a midpoint quadrature rule:

$$A = \int_0^1 F'(U + s(u - U))\, ds \mapsto A_m = F'(U + \frac{1}{2}(u - U)). \tag{4.4.44}$$

We can use (4.4.8) to bound $\|\phi_m - \bar{\phi}\|$ and use quadrature error results to bound $\|\bar{\phi} - \phi_m\|$.

Suppose we know that $\|u - U\| = \epsilon \ll 1$. Assuming F has continuous third order derivatives, we can write a Taylor expansion for the integrand about the midpoint $U + \frac{1}{2}(u - U)$. To simplify the expression, let $S = U + s(u - U)$, $m = U + \frac{1}{2}(u - U)$, and $p = S - m = (s - \frac{1}{2})(u - U)$. Now,

$$F'(S) = F'(m) + F''(m)[p] + \frac{1}{2}F'''(\xi_s)[p, p]. \tag{4.4.45}$$

The third derivative operator is evaluated at a point ξ_s that depends on s, and is on the line joining u to U. The square bracket notation indicates that the second and third derivatives are bi- and tri- linear operators, respectively, and they act on points $p \in \mathbb{R}^n$ and $[p, p] \in \mathbb{R}^n \times \mathbb{R}^n$ to produce linear operators on $\mathbb{R}^n$. Integrating both sides of this equation with respect to s, the middle term drops out, and we get

$$\int_0^1 F'(S)\, ds = F'(m) + \int_0^1 \frac{1}{2}F'''(\xi_s)[p, p]\, ds. \tag{4.4.46}$$

It follows that

$$\begin{aligned} \|A - A_m\| &\leq \sup_{\xi \in [U,u]} \|F'''(\xi)\| \|u - U\|^2 \int_0^1 \frac{1}{8}(2s - 1)^2\, ds \\ &\leq \frac{\epsilon^2}{24} \sup_{\xi \in [U,u]} \|F'''(\xi)\|, \end{aligned} \tag{4.4.47}$$

where $[U, u]$ denotes the line joining U to u. Transposing the operators, we have

$$\|A^\top - A_m^\top\| \leq \frac{\epsilon^2}{24} \sup_{\xi \in [U,u]} \|F'''(\xi)\|. \tag{4.4.48}$$

We consider $A^\top$ as a perturbation to $A_m^\top$; that is, $A^\top = A_m^\top + E$. Then, we have $\|A^\top - A_m^\top\| = \|E\|$ and

$$\|A_m^{-\top} - A^{-\top}\| = \|A_m^{-\top} - (A_m^\top + E)^{-1}\|. \tag{4.4.49}$$

This difference is bounded as follows [63]:

$$\begin{aligned} \|A_m^{-\top} - (A_m^\top + E)^{-1}\| &\leq \|A_m^{-\top}\| \, \|(A_m^\top + E)^{-1}\| \, \|E\| \\ &\leq C \sup_{\xi \in [U,u]} \|F'''(\xi)\| \epsilon^2. \end{aligned} \tag{4.4.50}$$

Finally, we can bound the error for $\|\bar{\phi} - \tilde{\phi}\|$:

$$\|\bar{\phi} - \tilde{\phi}\| \leq \|\bar{\phi} - \phi_m\| + \|\phi_m - \tilde{\phi}\| \leq C\epsilon^2 + \epsilon\sigma. \tag{4.4.51}$$

The constant C depends on A_m, E and F'''. In order to ensure that C is finite, we need some assumptions on the conditioning of the problem with respect to invertibility. Specifically, we want to assume that inverting the Jacobian at the true solution u is reasonably well conditioned in the sense that $F'(u)^{-1}$ is bounded away from the set of singular matrices. Then, there is a finite constant C that uniformly bounds all of the matrix norms in the above discussion.

The salient feature of the bound (4.4.51) is that $\|\bar{\phi} - \tilde{\phi}\|$ is primarily governed by σ, the size of the dominant singular value for $\Phi_\psi(U)$, since the first term on the right hand side involves ϵ^2, which will be much smaller than ϵ for small values of ϵ.

By considering a composite quadrature rule, we can obtain a similar bound for $\|\bar{\phi}-\tilde{\phi}\|$ that does not require such strong smoothness assumptions on $F(u)$. Given $m \in \mathbb{N}$, let

$$\begin{aligned} \phi_m &= \frac{1}{m}\sum_{i=1}^{m} \Phi_\psi \left(U + \frac{i}{m}(u-U) \right) \\ &= \frac{1}{m}\sum_{i=1}^{m} F' \left(U + \frac{i}{m}(u-U) \right)^{-\top} \psi. \end{aligned} \tag{4.4.52}$$

Now, the operator

$$A_m := \frac{1}{m}\sum_{i=1}^{m} F' \left(U + \frac{i}{m}(u-U) \right) \tag{4.4.53}$$

can be interpreted as a "left endpoint approximation" to the operator given by $A = \int_0^1 F'(U + s(u-U))\, ds$, and an estimate for the error of approximation is given by

$$\begin{aligned} \|A - A_m\| &= \left\| \int_0^1 F'(U+s(u-U))\, ds - \frac{1}{m}\sum_{i=1}^{m} F'\big(U + \frac{i}{m}(u-U)\big) \right\| \\ &= \left\| \sum_{i=1}^{m} \int_{\frac{i-1}{m}}^{\frac{i}{m}} \left(F'(U+s(u-U)) - F'\big(U+\frac{i}{m}(u-U)\big) \right) ds \right\| \\ &\leq \frac{1}{m} \max_{\xi\in[U,u]} \|F''(\xi)\|\, \|u-U\| \sum_{i=1}^{m} \max_{s\in[\frac{i-1}{m},\frac{i}{m}]} \left| \left(s - \frac{i}{m} \right) \right| \\ &= \frac{1}{m} \max_{\xi\in[U,u]} \|F''(\xi)\|\, \epsilon. \end{aligned} \tag{4.4.54}$$

Taking $m > \dfrac{1}{\epsilon}$ ensures this bound is quadratic in ϵ, and only requires that F be twice continuously differentiable. Arguing as above, we obtain a bound for $\|\bar{\phi} - \phi_m\|$:

$$\begin{aligned} \|A_m^{-\top} - (A_m^\top + E)^{-1}\| &\leq \|A_m^{-\top}\|\, \|(A_m^\top + E)^{-1}\|\, \|E\| \\ &\leq \frac{C}{m} \sup_{\xi\in[U,u]} \|F''(\xi)\|\epsilon \\ &\leq C \sup_{\xi\in[U,u]} \|F''(\xi)\|\epsilon^2, \qquad m > \frac{1}{\epsilon}. \end{aligned} \tag{4.4.55}$$

Note further that

$$\begin{aligned}
\|\phi_m - \tilde{\phi}\| &= \left\| \left(\sum_{i=1}^{m} \Phi_\psi \left(U + \frac{i}{m}(u-U) \right) \right) - \Phi_\psi(U) \right\| \\
&= \left\| \frac{1}{m} \left(\sum_{i=1}^{m} \Phi_\psi \left(U + \frac{i}{m}(u-U) \right) - \Phi_\psi(U) \right) \right\| \\
&\leq \frac{1}{m} \sum_{i=1}^{m} \left\| \Phi_\psi \left(U + \frac{i}{m}(u-U) \right) - \Phi_\psi(U) \right\| \\
&\leq \frac{1}{m} \sum_{i=1}^{m} \sigma\epsilon = \sigma\epsilon.
\end{aligned} \tag{4.4.56}$$

Putting (4.4.55) and (4.4.56) together, we get

$$\|\bar{\phi} - \tilde{\phi}\| \leq \|\bar{\phi} - \phi_m\| + \|\phi_m - \tilde{\phi}\| \leq C\epsilon^2 + \epsilon\sigma, \qquad m > \frac{1}{\epsilon}. \tag{4.4.57}$$

Again, the constant C depends on $(A_m^\top)^{-1}$ and E and the previous comments apply.

We see again that the dominant singular value of the map Φ_ψ is the primary factor in the estimate. We conclude that even though dual solutions of the form $\Phi(U + \epsilon w)$ cannot exactly reproduce the dual solution corresponding to linearizing around the correct integral average, even for small ϵ, the discrepancy is not worse than quadratic in ϵ and can be safely ignored for small ϵ.

4.5 Green's function approach

In this section, we use the Green's function for the solution of the linearized dual problem to derive an explicit formula for $\Phi'_\psi(U)$. This reveals how the derivative depends on the coefficients of the forward problem as well as the dual solution and some of its derivatives.

4.5.1 Review of Green's functions

We give a brief description of the Green's function. The reason is to give an overview, without delving into the rigor required for a complete exposition. In particular, we do not discuss distribution theory in any detail. Much can be found on that subject in the original references [35, 90], as well as [53, 76, 77, 100, 102]. Additionally, full treatments of Green's functions for ordinary differential equations can be found in [54, 71, 87, 93].

If L is an ordinary differential operator acting on a space of functions, we can describe a differential equation by

$$Lu = f, \tag{4.5.1}$$

together with some initial or boundary conditions. We solve the equation by finding the inverse L^{-1} and applying it to (4.5.1) to recover u. When the inverse exists, it takes the form of an integral operator. The kernel of the integral operator is called the *Green's function* for L [87]. Formally, applying L^{-1} to both sides of (4.5.1) gives

$$\begin{aligned} u(x) &= L^{-1}f(x) \\ &= \int G(x,\xi)f(\xi)\,d\xi. \end{aligned} \tag{4.5.2}$$

The Green's function $G(x,\xi)$ satisfies the differential equation with δ–function data, centered at ξ:

$$LG(x,\xi) = \delta(x-\xi). \tag{4.5.3}$$

From this we see that G is the kernel of an integral operator that inverts L. From (4.5.2),

$$Lu(x) = \int LG(x,\xi)f(\xi)\,d\xi = \int \delta(x-\xi)f(\xi)\,d\xi = f(x). \tag{4.5.4}$$

In the next subsection, we derive a formula for the derivative of the map $\Phi(U)$ defined in §4.4.1 that is based on the Green's function for the linearized dual problem (4.3.28). We also derive a formula for the adjoint, with a view to constructing a power method algorithm for recovering the dominant singular value and left singular vector for a discrete representation of $\Phi'_\psi(U)$, relative to some discretization.

4.5.2 A formula using the Green's function

Denote the adjoint problem (4.3.29) by

$$F(U)^\top \phi = \psi, \text{ i.e. } \phi = F(U)^{-\top}\psi, \tag{4.5.5}$$

so that ϕ is the dual solution corresponding to linearization around U. Then, using the Green's function associated with the linear dual problem to represent $F(U)^{-\top}$, we have

$$\phi(x) = F(U)^{-\top}\psi = \int_0^1 G_U(x,y)\psi(y)\,dy. \tag{4.5.6}$$

We attach the subscript U to G to indicate that this is the Green's function for the dual problem corresponding to linearization around U.

Now, let ϕ_1 and ϕ_2 be dual solutions corresponding to linearization around U_1 and U_2, respectively:

$$\begin{cases} -a(U_1)\phi_1'' - b(U_1)\phi_1' - f'(U_1)\phi_1 = \psi, \\ \phi_1(0) = \phi_1(1) = 0, \quad x \in [0,1], \end{cases} \tag{4.5.7}$$

and

$$\begin{cases} -a(U_2)\phi_2'' - b(U_2)\phi_2' - f'(U_2)\phi_2 = \psi, \\ \phi_2(0) = \phi_2(1) = 0, \quad x \in [0,1]. \end{cases} \tag{4.5.8}$$

Subtracting these equations and adding some terms to both sides gives

$$\begin{aligned} &-[a(U_1)\phi_1'' - a(U_1)\phi_2''] - [b(U_1)\phi_1' - b(U_1)\phi_2'] \\ &\qquad - [f'(U_1)\phi_1 - f'(U_1)\phi_2] \\ &= -a(U_2)\phi_2'' + a(U_1)\phi_2'' - b(U_2)\phi_2' + b(U_1)\phi_2' \\ &\qquad - f'(U_2)\phi_2 + f'(U_1)\phi_2, \end{aligned} \tag{4.5.9}$$

which can be written as

$$\begin{aligned} &-a(U_1)\Delta\phi'' - b(U_1)\Delta\phi' - f'(U_1)\Delta\phi \\ &\qquad = \tilde{a}(U_1 - U_2)\phi_2'' + \tilde{b}(U_1 - U_2)\phi_2' + \tilde{f}(U_1 - U_2)\phi_2, \end{aligned} \tag{4.5.10}$$

where

$$\begin{aligned} \Delta\phi &:= \phi_1 - \phi_2, \\ \tilde{a} &:= \int_0^1 a'(sU_1 + (1-s)U_2)\, ds, \\ \tilde{b} &:= \int_0^1 b'(sU_1 + (1-s)U_2)\, ds, \\ \tilde{f} &:= \int_0^1 f''(sU_1 + (1-s)U_2)\, ds. \end{aligned} \tag{4.5.11}$$

In terms of the operator F, we have

$$F(U_1)^{\top}\Delta\phi = \tilde{a}(U_1 - U_2)\phi_2'' + \tilde{b}(U_1 - U_2)\phi_2' + \tilde{f}(U_1 - U_2)\phi_2. \tag{4.5.12}$$

Now,

$$\Delta\phi = F(U_1)^{-\top}\left\{\tilde{a}(U_1 - U_2)\phi_2'' + \tilde{b}(U_1 - U_2)\phi_2' + \tilde{f}(U_1 - U_2)\phi_2\right\}. \tag{4.5.13}$$

If we let $U_1 - U_2 = \epsilon Z$, then

$$\begin{aligned} \frac{\Delta\phi}{\epsilon} &= \frac{1}{\epsilon}\left(F(U_1)^{-\top}\left\{\tilde{a}\epsilon Z\phi_2'' + \tilde{b}\epsilon Z\phi_2' + \tilde{f}\epsilon Z\phi_2\right\}\right) \\ &= F(U_1)^{-\top}\left\{\tilde{a}Z\phi_2'' + \tilde{b}Z\phi_2' + \tilde{f}Z\phi_2\right\} \\ &= \int_0^1 G_{U_1}(x,y)\left\{\tilde{a}Z(y)\phi_2''(y) + \tilde{b}Z(y)\phi_2'(y) + \tilde{f}Z(y)\phi_2(y)\right\} dy \end{aligned} \tag{4.5.14}$$

Now as $\epsilon \to 0$, we have $U_2 \to U_1$ and $\phi_2 \to \phi_1$, and, assuming sufficient smoothness,

$$\begin{aligned}
\tilde{a} &= \int_0^1 a'(sU_1 + (1-s)U_2)\,ds \to a'(U_1), \\
\tilde{b} &= \int_0^1 b'(sU_1 + (1-s)U_2)\,ds \to b'(U_1), \\
\tilde{f} &= \int_0^1 f''(sU_1 + (1-s)U_2)\,ds \to f''(U_1).
\end{aligned} \tag{4.5.15}$$

Finally, we get

$$\left(F(U_1)^{-\top}\right)'(Z(x)) = \lim_{\epsilon \to 0}\left(\frac{\Delta\phi}{\epsilon}\right), \tag{4.5.16}$$

so we can express the derivative as

$$\int_0^1 G_{U_1}(x,y)\Big\{a'\left(U_1(y)\right)\phi_1''(y) + b'\left(U_1(y)\right)\phi_1'(y) + f''\left(U_1(y)\right)\phi_1(y)\Big\}Z(y)\,dy. \tag{4.5.17}$$

Note that for $\epsilon = 0$, $\phi_2 = \phi_1 = F(U_1)^{-\top}\psi$ depends on $U_1 = U_2$, the point at which the derivative is to be evaluated. This way of expressing the derivative of makes it clear that its properties depend on the properties of the functions a, b and f.

We can connect this expression for the derivative of the mapping between the forward solution and linearized dual solution with the general formula (4.4.12) that we derived in §4.4.1. Recall the forward problem (4.3.17):

$$\begin{cases} -(a(u)u')' + b(u)u' = f(u), \\ u(0) = u(1) = 0. \end{cases} \tag{4.5.18}$$

We can write this as $F(u) = 0$, where

$$F(u) := -(a(u)u')' + b(u)u' - f(u). \tag{4.5.19}$$

The linearized dual problem is

$$\begin{cases} F'(U)^\top \phi = \psi, \\ \phi(0) = \phi(1) = 0, \end{cases} \tag{4.5.20}$$

where

$$F'(U)^\top \equiv -a(U)\frac{d^2}{dx^2} - b(U)\frac{d}{dx} - \frac{\partial f}{\partial U}, \tag{4.5.21}$$

and U is the approximate solution to the forward problem. In this context, $\Phi_\psi(U)$ is defined by

$$\Phi_\psi(U) := F'(U)^{-\top}\psi = \int_0^1 G_U(x,y)\psi(y)\,dy. \tag{4.5.22}$$

As in §4.4.1, we can write the dual problem as

$$F'(U)^\top \Phi_\psi(U) = \psi. \tag{4.5.23}$$

We derived formula (4.4.12) for Φ'_ψ by differentiating this expression implicitly, and we can employ the same method here. Differentiating (4.5.23) with respect to U, we get

$$\left[F'(U)^\top\right]' \Phi_\psi(U) + F'(U)^\top \Phi'_\psi(U) = 0, \tag{4.5.24}$$

which gives

$$\Phi'_\psi(U) = -F'(U)^{-\top}\left[\left(F'(U)^\top\right)'\phi\right], \tag{4.5.25}$$

with $\Phi'_\psi(U)$ denoting the derivative of $\Phi_\psi(U)$ with respect to U. Recall that

$$F'(U)^\top = -a(U)\frac{d^2}{dx^2} - b(U)\frac{d}{dx} - \frac{\partial f}{\partial U}, \tag{4.5.26}$$

so

$$\left(F'(U)^\top\right)' = -\frac{\partial a}{\partial U}\frac{d^2}{dx^2} - \frac{\partial b}{\partial U}\frac{d}{dx} - \frac{\partial^2 f}{\partial U^2}. \tag{4.5.27}$$

Therefore,

$$\begin{aligned}\Phi'_\psi(U) &= -F'(U)^{-\top}\left[\left(-\frac{\partial a}{\partial U}\frac{d^2}{dx^2}-\frac{\partial b}{\partial U}\frac{d}{dx}-\frac{\partial^2 f}{\partial U^2}\right)\phi\right] \\ &= \int_0^1 G_U(x,y)\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}dy.\end{aligned}$$

The action of $\Phi'_\psi(U)$ on a function $z(x)$ is given by

$$\Phi'_\psi(U)z(x) = \int_0^1 G_U(x,y)\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y) + \frac{\partial^2 f}{\partial U^2}\phi(y)\right\}z(y)\,dy. \quad (4.5.28)$$

This is the same formula we derived by computing $\lim_{\epsilon\to 0}\left(\frac{\Delta\phi}{\epsilon}\right)$.

4.5.3 A formula for the adjoint

We denote the operator $\Phi'_\psi(U)$ in (4.5.28) by L, and seek a formula for the adjoint L^*. We have

$$\begin{aligned}\langle Lz, w\rangle &= \int_0^1\left[\int_0^1 G_U(x,y)\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}z(y)\,dy\right]w(x)\,dx \\ &= \int_0^1\int_0^1 G_U(x,y)\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}w(x)\,dx\,z(y)\,dy \\ &= \int_0^1\int_0^1 G_U(x,y)w(x)\,dx\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}z(y)\,dy \\ &= \langle z, L^*w\rangle.\end{aligned}$$

Thus,

$$L^*w = \left[\int_0^1 G_U(x,y)w(x)\,dx\right]\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi'(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}. \quad (4.5.29)$$

So, while the operator L can be characterized by "multiply the Green's function by $\left\{\frac{\partial a}{\partial U}\phi''(y)+\frac{\partial b}{\partial U}\phi(y)+\frac{\partial^2 f}{\partial U^2}\phi(y)\right\}$ and integrate against the data," we see here that the dual operator is characterized by "integrate the Green's

function against the data with respect to the other variable, then multiply by $\left\{ \frac{\partial a}{\partial U}\phi''(y) + \frac{\partial b}{\partial U}\phi(y) + \frac{\partial^2 f}{\partial U^2}\phi(y) \right\}$. By numerically reconstructing the appropriate quantities in both L and L^*, we can apply them in succession and create a power method for the symmetric operator LL^*. This way, we can estimate the dominant singular value and left singular vector for $\Phi'_\psi(U)$. We discuss this implementation in §4.7.5.

4.6 Conditioning for nonlinear equations

The condition number of a matrix is a measure of the sensitivity of the solution to a linear system of equations. In [85], Rheinboldt extends the notion of condition number to nonlinear equations on normed linear spaces. In this setting, the condition number depends on the domain, and Rheinboldt shows that the condition number of the nonlinear problem asymptotically reduces to that of the derivative at a point. In this section we examine the map Φ defined in (4.4.2) in this context, and show that the quantity $\|\Phi'_\psi(U)\|$ is an appropriate measure of the sensitivity for changes in the dual solution ϕ to (4.3.28) corresponding to perturbations of the forward solution U to (4.3.16).

For real normed linear spaces X, Y and the space $L(X, Y)$ of bounded linear operators from X to Y, Rheinboldt defines the quantities

$$\mu(F, C) = \sup\{t \in [0, \infty); \|Fx - Fy\| \geq t\|x - y\| \ \forall x, y \in C\}, \qquad (4.6.1)$$

and

$$\nu(F, C) = \inf\{t \in [0, \infty]; \|Fx - Fy\| \leq t\|x - y\| \ \forall x, y \in C\}, \qquad (4.6.2)$$

for any mapping $F : D \subset X \to Y$ and any closed subset $C \subset D$. Note that the value $t = \infty$ is allowed in (4.6.2). The condition number of the

mapping F on the set C is defined as

$$\kappa(F,C) = \begin{cases} \dfrac{\mu(F,C)}{\nu(F,C)}, & \text{if } 0 < \mu(F,C), \nu(F,C) < \infty, \\ \infty, & \text{otherwise.} \end{cases} \tag{4.6.3}$$

This reduces to the standard definition $\kappa(F) = \|F\|\|F^{-1}\|$ in case $F \in L(\mathbb{R}^n, \mathbb{R}^n)$. Rheinboldt examines solutions to nonlinear equations

$$Fx = b, \tag{4.6.4}$$

and

$$Gx = c, \tag{4.6.5}$$

with mappings F, G from X to Y. Here, he takes the maps F and G "close to each other in the sense that on some set C the difference mapping

$$E : C \subset X \to Y,\ Ex = Fx - Gx\ \forall\ x \in C \tag{4.6.6}$$

has a sufficiently small Lipschitz norm." In other words, the map G can be viewed as a perturbation of F.

Rheinboldt shows that if $C_1 \subset C_2$, then

$$\kappa(F, C_1) \leq \kappa(F, C_2), \tag{4.6.7}$$

which suggests that we must consider the limit as the domain shrinks to a point. For $F : D \subset X \to Y$, $C \subset D$ (C closed), and $z \in C$, he defines the localized bounds

$$\mu^0(F, C, z) = \sup\{t \in [0, \infty);\ \|Fx - Fz\| \geq t\|x - y\|\ \forall x \in C\}, \tag{4.6.8}$$

and

$$\nu^0(F, C, z) = \inf\{t \in [0, \infty];\ \|Fx - Fz\| \leq t\|x - y\|\ \forall x \in C\}, \tag{4.6.9}$$

and the corresponding localized condition number

$$\kappa^0(F,C) = \begin{cases} \dfrac{\mu^0(F,C)}{\nu^0(F,C)} & \text{if } 0 < \mu(F,C), \nu(F,C) < \infty \\ \infty, & \text{otherwise.} \end{cases} \tag{4.6.10}$$

We state two of the theorems in [85] that are relevant. They appear as Theorem 3.2 and Theorem 3.3 in that paper.

Theorem 4.6.1. *Suppose that the continuous mapping $F : D \subset X \to Y$ has a Fréchet derivative $F'(z) \in L(X,Y)$ at $z \in \text{int}(D)$ for which the inverse $F'(z)^{-1} \in L(Y,X)$ exists. Then for any sufficiently small $\epsilon > 0$ there is a $\delta > 0$ such that $C = \{x \in X; \|x - z\| \leq \delta\} \subset D$ and*

$$|\nu^0(F,C,z) - \|F'(z)\|| \leq \epsilon, \tag{4.6.11}$$

and

$$|\mu^0(F,C,z) - \|F'(z)^{-1}\|^{-1}| \leq \epsilon. \tag{4.6.12}$$

From (4.6.11) and (4.6.12), it follows that

$$-\frac{\epsilon}{\|F'(z)^{-1}\|^{-1} + \epsilon} \leq \frac{\kappa^0(F,C,z) - \kappa(F'(z))}{1 + \kappa(F'(z))} \leq \frac{\epsilon}{\|F'(z)^{-1}\|^{-1} - \epsilon}, \tag{4.6.13}$$

where $\kappa(F'(z)) = \|F'(z)\|\|F'(z)^{-1}\|$ is the condition number of the derivative on X.

Theorem 4.6.2. *Let $F : D_F \subset X \to Y$ be such that $\kappa^0(F,C,x^*) < \infty$ on some closed set $C \subset D_F$ and that (4.6.4) has a solution $x^* \in C$. Then, for any $G : D_G \subset X \to Y$, $C \subset D_G$, for which (4.6.5) has a solution $y^* \in C$, we have*

$$\frac{\|x^* - y^*\|}{\|x^* - x^0\|} \leq \kappa^0(F,C,x^*)\left[\frac{\|b - c\|}{\|b - Fx^0\|} + \frac{\|G - F\|_C}{\|b - Fx^0\|}\right]. \tag{4.6.14}$$

Here, $x^0 \in C$, $x^0 \neq x^*$ is any point, and $\|G - F\|_C = \sup\{\|Fx - Gx\| \; \forall \; x \in C\}$.

In practice, we analyze the map Φ by producing an approximate solution U to the forward problem, and then add perturbations of a given size to U and examine the corresponding changes in the dual solution ϕ. In this situation, equations (4.6.4) and (4.6.5) become

$$\begin{cases} F\phi = U \\ F\phi = U + \epsilon w; \quad \epsilon \ll 1, \ \|w\| = 1. \end{cases} \tag{4.6.15}$$

Here, F represents the inverse Φ^{-1}, and we assume the hypotheses of the inverse function theorem hold on the ball $B_\epsilon(U)$. Using the notation introduced in §(4.4.1), we let ϕ denote the solution to the dual problem corresponding to linearization around U, and $\phi_{\epsilon w}$ the dual solution corresponding to linearization around $U + \epsilon w$. In this case, equation (4.6.14) in Theorem 4.6.2 is

$$\frac{\|\phi - \phi_{\epsilon w}\|}{\|\phi - \phi^0\|} \leq \kappa^0(F, C, \phi)\left[\frac{\epsilon}{\|U - F\phi^0\|}\right]. \tag{4.6.16}$$

Here, C is a closed set contained in $B_\epsilon(U)$, which can be taken to be the intersection of the closed set in Theorem 4.6.2 and the closed set given by the inverse function theorem, if necessary. We can write (4.6.16) as

$$\|\phi - \phi_{\epsilon w}\| \leq \kappa^0(F, C, \phi)\left[\frac{\|\phi - \phi^0\|}{\|U - F\phi^0\|}\right]\epsilon, \tag{4.6.17}$$

and, denoting by U^0 the forward solution corresponding to the dual solution ϕ^0, we can further write this as

$$\|\phi - \phi_{\epsilon w}\| \leq \kappa^0(F, C, \phi)\left[\frac{\|\Phi(U) - \Phi(U^0)\|}{\|U - U^0\|}\right]\epsilon. \tag{4.6.18}$$

Note that as $\epsilon \to 0$, $\dfrac{\|\Phi(U) - \Phi(U^0)\|}{\|U - U^0\|} \to \|\Phi'(U)\|$ and $\kappa^0(F, C, \phi) \to \kappa(F'(\phi))$ by Theorem 4.6.1. Using the inverse function theorem,

$$\begin{aligned} \kappa(F'(\phi)) &= \|F'(\phi)\|\|F'(\phi)^{-1}\| \\ &= \|\Phi'(U)^{-1}\|\|\Phi'(U)\| \\ &= \kappa(\Phi'(U)). \end{aligned} \tag{4.6.19}$$

These results imply that

$$\|\phi - \phi_{\epsilon w}\| \leq \|\Phi'(U)\|\epsilon, \tag{4.6.20}$$

which we have seen already from the direct linearization of the map Φ in §4.4.1.

In theory, we wish to examine the problem

$$\begin{cases} F\phi = U \\ G\phi = U, \end{cases} \tag{4.6.21}$$

where G represents the inverse of the map corresponding to solving the dual problem around the correct integral average of u and U given in (4.3.14). Letting $\phi_G = G^{-1}(U)$, equation (4.6.14) reads

$$\frac{\|\phi - \phi_G\|}{\|\phi - \phi^0\|} \leq \kappa^0(F, C, \phi)\left[\frac{\|G - F\|_C}{\|U - F\phi^0\|}\right]. \tag{4.6.22}$$

Using the same estimates as before, we see asymptotically as $\epsilon \to 0$ that

$$\|\phi - \phi_G\| \leq \|\Phi'(U)\|\|G - F\|_C. \tag{4.6.23}$$

It is worthwhile to compare equations (4.6.20) and (4.6.23): the quantities in the latter describe exactly the sensitivity to linearization, but since we never know the exact solution u, the operator G is unknown, and the dual solution ϕ_G is impossible to obtain. We give up a direct attack on $\|G - F\|_C$ in (4.6.20) by replacing G with F and examining perturbations to the known data U. This is not unreasonable in light of the difficult nature of determining G. If we have more information about the linearized dual operator, then it may be possible to say more. For instance, if we know that the derivative of the operator f in the forward problem $f(u) = 0$

is Lipschitz continuous with constant L, then

$$\begin{aligned}\left\| \int_0^1 f'(su+(1-s)U)\,ds - f'(U) \right\| &\leq \int_0^1 \|f'(su+(1-s)U) - f'(U)\|\,ds \\ &\leq \int_0^1 Ls\|u-U\|\,ds \\ &= \frac{L}{2}\|u-U\|. \end{aligned} \tag{4.6.24}$$

Thus, if we know that the error $e = u - U$ is small for a sufficiently fine discretization, we may take $\epsilon < \frac{L}{2}\|e\|$ in (4.6.20) when we make computations. Regardless, the appearance of the quantity $\|\Phi'_\psi(U)\|$ in both expressions suggests its value as a sensitivity measure. The results in Rheinboldt's paper [85] verify the role played here by the derivative $\Phi'_\psi(U)$ of the nonlinear map Φ_ψ between the approximate forward solution U and the dual problem corresponding to linearization around U that we defined in §4.4.1.

4.7 Implementation

In this section, we describe the programs that we use to investigate the linearization issue for two–point boundary value problems.

4.7.1 Solution of the forward problem

Recall the two–point boundary value problem (4.3.17):

$$\begin{cases} -((a(u)u')' + b(u)u'(x) = f(u), \\ u(0) = u(1) = 0, \end{cases} \tag{4.7.1}$$

The weak form of the differential equation is obtained by multiplying (4.7.1) by a test function $v(x)$, with $v(0) = v(1) = 0$ and integrating the second order term by parts. We seek $u \in V$ such that

$$\begin{aligned} -\int_0^1 (au')'v\,dx + \int_0^1 bu'v\,dx &= \int_0^1 fv\,dx \\ \Rightarrow \int_0^1 (au'v' + bu'v)\,dx &= \int_0^1 fv\,dx \quad \forall v \in V, \end{aligned} \tag{4.7.2}$$

where $V = H_0^1[0,1]$, the Sobolev space of functions on $L_2[0,1]$ whose weak first derivatives are also in $L^2[0,1]$. Given a partition $0 = x_0 < x_1 < \ldots < x_{M+1} = 1$ with $h_i = x_i - x_{i-1}$, we let $V_h^{(1)}$ be the space of continuous linear functions with value zero at $x = 0$ and $x = 1$. $V_h^{(1)}$ is an M dimensional subspace of $H_0^1[0,1]$, and a basis is given by

$$\phi_j = \begin{cases} 0, & x \notin [x_{i-1}, x_{i+1}] \\ \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, x_i] \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}}, & x \in [x_i, x_{i+1}], \end{cases} \tag{4.7.3}$$

for $i = 1 \ldots M$. We discretize (4.7.2) using the continuous Galerkin (cG(1)) method to compute $U \in V_h^{(1)}$ such that

$$\int_0^1 (a(U)U'v' + b(U)U'v)\,dx = \int_0^1 f(U)v\,dx \tag{4.7.4}$$

for all $v \in V_h^{(1)}$, where $U = \sum_{j=1}^{M} U_j\phi_j$. Choosing $v = \phi_i$ for $1 \leq i \leq M$ leads to the system of equations

$$\sum_{j=1}^{M} U_j \int_0^1 a(U_i)\phi_j'\phi_i' + b(U_i)\phi_j'\phi_i\,dx = \int_0^1 f(U_i)\phi_i\,dx \qquad 1 \leq i \leq m. \tag{4.7.5}$$

Each basis function is supported on $[x_{i-i}, x_{i+1}]$, so the integrals vanish except when $i = j - i$, $i = j$, and $i = j + 1$. This leads to the discrete system

of equations

$$
\begin{aligned}
&-\left[\frac{(a(U_i,x_i)+a(U_{i-1},x_{i-1}))}{2h_i}\right]U_{i-1}\\
&\quad+\left\{\left[\frac{(a(U_{i-1},x_{i-1})+a(U_i,x_i))}{2h_i}\right]+\left[\frac{(a(U_i,x_i)a(U_{i+1},x_{i+1}))}{2h_{i+1}}\right]\right\}U_i\\
&\qquad-\left[\frac{(a(U_i,x_i)+a(U_{i+1},x_{i+1}))}{2h_{i+1}}\right]U_{i+1}\\
&\qquad+\left[\frac{b(U_i,x_i)}{2}\right](U_{i+1}-U_{i-1})\\
&\qquad=\frac{h_i+h_{i+1}}{2}f(U_i,x_i). \quad (4.7.6)
\end{aligned}
$$

We solve this nonlinear system of equations by defining $F_i(U)$ as the difference between the left and right hand sides of (4.7.6) and solving $F(U)=0$ using Newton's method with damping. Damping is used when the initial guess for the method is not good. We view the computation of a root of $F(U)=0$ as trying to minimize $\|F(U)\|$. It is possible that a Newton step causes $\|F(U^{(i)})\| > \|F(U^{(i-1)})\|$, for example when $U^{(i-1)}$ is far from a root, or when the Jacobian $F'(U^{(i-1)})$ is nearly singular. Now, the i^{th} Newton change

$$\delta_i = F'(U^{(i-1)})^{-1}F(U^{(i-1)}) \qquad (4.7.7)$$

is a direction of decrease for $\|F(U)\|$ from $U^{(i-1)}$, so $\|F(U)\| < \|F(U^{(i-1)})\|$ for all U sufficiently close to $U^{(i-1)}$ lying in the direction of δ_i. "Damping" means that we move in the direction of the Newton change, but not as far as indicated by Newton's method. With damping, we try to find an iterate in the direction of δ_i from the previous iterate that is relatively large, but for which the successive norm decreases. The algorithm performs a bisection search in the direction of the Newton change in order to find an iterate that decreases the norm. The algorithm is

Algorithm 4.7.1. *(Damped Newton Method)*

1. $U_{old} = U_0$
2. *For* $i = 1$ *to MAXIT*
   - *Solve* $F'(U_{old})\delta = F(U_{old})$
   - *set* $U_{new} = U_{old} + \delta$
   - *IF* $\|\delta\| < TOL$ *break*
   - *IF* $\|F(U_{new})\| > \|F(U_{old})\|$
     - *set* $U_{new} = \dfrac{U_{new} + U_{old}}{2}$
     - *WHILE* $\|F(U_{new})\| > \|F(U_{old})\|$
     - $U_{new} = \dfrac{U_{new} + U_{old}}{2}$
     - *end WHILE*
   - *end IF*
   - $U_{old} = U_{new}$
3. *end*

This algorithm produces a sequence $\{U^{(0)}, U^{(1)}, U^{(2)}, \ldots\}$ of Newton iterates that converge to the root of the equation $F(U) = 0$. The solution of the equation $F'(U_{old})\delta = F(U_{old})$ is accomplished by constructing the Jacobian $F'(U_{old})$ as a sparse, banded matrix and solving the linear system. The matrix is banded and has the same sparsity structure as the discretized

problem. The entries in the Jacobian are given by

$$\begin{aligned}
\frac{\partial F_i(U)}{\partial U_{i-1}} &= -\left[\frac{(a(U_i,x_i)+a(U_{i-1},x_{i-1})+a'(U_{i-1},x_{i-1})}{2h_i}\right](U_i-U_{i-1}) \\
&\quad -\frac{b(U_{i+1},x_{i+1})}{2}; \\
\frac{\partial F_i(U)}{\partial U_i} &= \left[\frac{a(U_{i-1},x_{i-1}}{2h_i}+\left(\frac{1}{2h_i}+\frac{1}{2h_{i+1}}\right)a(U_i,x_i)+\frac{a(U_{i+1},x_{i+1})}{2h_{i+1}}\right] \\
&\quad +\frac{(U_i-U_{i-1})}{2h_i}a'(U_i,x_i)-\frac{(U_i+U_{i-1})}{2h_{i+1}}a'(U_i,x_i) \\
&\quad +\frac{(U_{i+1}+U_{i-1})}{2}b'(U_i,x_i)-\frac{h_i-h_{i+1}}{2}f'(U_i,x_i); \\
\frac{\partial F_i(U)}{\partial U_{i+1}} &= -\left[\frac{(a(U_i,x_i)+a(U_{i+1},x_{i+1})+a'(U_{i+1},x_{i+1})}{2h_{i+1}}\right](U_i-U_{i-1}) \\
&\quad +\frac{b(U_{i-1},x_{i-1})}{2}. \qquad (4.7.8)
\end{aligned}$$

As before, the notation $a'(u,x)$ etc., denotes differentiation with respect to u and not to x. Additionally, in this thesis we only consider uniform discretizations, so $h_i = h_{i+1}$ for all i.

In some examples that we consider later on, the Newton algorithm does not converge to the correct solution. In cases where we know the exact solution u, we circumvent Newton's method and use values of the true solution to construct a solution U in an appropriate space of approximate solutions. Typically, we take the nodal values of the true solution and construct a piecewise linear interpolant, which serves as the approximate solution about which we linearize the coefficients of the dual problem. In this way, we can control the size of the error in the forward solution and examine how it affects change in the dual solution in cases where the forward solution process does not work.

4.7.2 Solution of the dual problem

Recall the form of the dual solution used in the finite element method is

$$\begin{cases} -\left(a(U)\phi'\right)' + \left(a'(U)U' - b(U)\right)\phi' - f'(U)\phi = \psi, \\ \phi(0) = \phi(1) = 0. \end{cases} \tag{4.7.9}$$

After the Newton algorithm produces the approximate solution U to the forward problem (4.7.1), we evaluate the coefficients in the dual problem, specify the dual data ψ, and solve the dual problem. The code produces three different finite element solutions to the dual problem. The first uses the same basis functions (4.7.3), the second uses quadratic Lagrange basis elements (see [42], page 200), and the third uses cubic Hermite elements (see [99], page 58). We compute the dual solution in these spaces of smoother functions to examine properties of the error estimates that require more smoothness. For example, solving the dual problem in the same space as the forward solution (e.g. $\phi \in V_h^{(1)}$) makes analysis easier, since in this case the domain and range of the map $\Phi : U \to \phi$ have the same dimension. Solving the dual problem in $V_h^{(2)}$ is done for examining error estimates that give second derivative information, e.g. $\phi - \pi_h\phi$. We use the cubic Hermite elements to construct dual solutions with smooth second derivatives, in order to facilitate examining the formulas for $\Phi'_\psi(U)$ that are discussed in §4.5. See equation (4.5.17), for example.

4.7.3 The nonlinear power method

With the computed forward and dual solutions in hand, we can implement the NLPM in a matrix–free way as set forth in Algorithm 2.4.1. While the algorithm itself is relatively straightforward, there are some subtleties

with the manner in which we compute the sequence

$$\Phi(U + \delta_i) - \Phi(U). \tag{4.7.10}$$

In particular, if we solve the forward problem $U \in V_h^{(1)}$ but the dual solution $\phi = \Phi(U) \in V_h^{(2)}$, then the input and output spaces for Φ are not the same, so it is not simply a matter of feeding the previous iterate back into the loop. In case U is in a "smaller" space than ϕ, we attach the appropriate nodal values to U to "lift" U into the same space as ϕ. In the case where $U \in V_h^{(1)}$, adding midpoint values between the nodes is sufficient. This is not an issue when the forward and dual problems are in the same space. Also, some of the coefficients depend on the derivative of the forward solution, and this information must be computed numerically. The values for the coefficients of the dual problem are expressed on a grid of gauss nodes by forming linear combinations of the appropriate basis functions, which are defined by the code and evaluated on the gauss nodes.

4.7.4 Constructing $\Phi'_\psi(U)$

Once we obtain the linearized dual solution ϕ in a finite dimensional vector space, we construct a finite-dimensional approximation to the operator $\Phi'_\psi(U)$ by solving a sequence of dual problems: we fix a small value for the parameter ϵ, successively scale the basis functions e_j, perturb the forward solution, and compute

$$\frac{\Phi(U + \epsilon e_j) - \Phi(U)}{\epsilon}, \quad j = 1 \ldots M, \tag{4.7.11}$$

where M is the dimension of the space for the dual problem. In this way, we determine the action of the matrix $\Phi'_\psi(U)$ on the basis, and concatenate these to form the matrix. With the matrix in hand, we can use MATLAB

to find eigenvalues and eigenvectors, and to perform a singular value decomposition. We can then compare these results with those of algorithms (2.4.1) and (2.4.2), as well as the codes based on equation (4.5.17) in §4.5.6.

4.7.5 Approximate Green's function

In §4.5, we discussed an approach to $\Phi'_\psi(U)$ that utilizes the Green's function for the linearized dual problem. Using equations (4.5.17) for $\Phi'_\psi(U)$ and (4.5.29) for its adjoint, we can construct an algorithm that performs a power method for $\Phi'_\psi(U)^\top \Phi'_\psi(U)$, thereby providing an alternate route to the dominant singular value and singular vector.

We approximate the Green's function for the linearized dual problem by inverting the stiffness matrix for the system of discrete equations that arises in the finite element formulation of the linearized dual problem. We can do this because the Green's function satisfies the dual problem with delta function data. The load vector for the dual problem with a delta function centered at the j^{th} node is given by

$$\int_0^1 \delta_{x_j}(x)\phi_i(x)\,dx = \phi_i(x_j) = \begin{cases} 1 & i = j, \\ 0 & \text{otherwise}, \end{cases} \tag{4.7.12}$$

since the ϕ_i are Lagrange basis functions. From this we see that the load vector corresponding to the i^{th} equation is the standard basis vector e_i. Therefore, the matrix that represents the Green's function corresponding to solving dual problems with delta data centered at the nodes is

$$SG = I \Rightarrow G = S^{-1}, \tag{4.7.13}$$

where S is the stiffness matrix for the finite element that produces the dual solution corresponding to linearization around the approximate forward solution U.

To demonstrate, we numerically reconstruct the Green's function for the problem

$$-u'' = f(x), \quad f(0) = f(1), \ x \in [0,1]. \tag{4.7.14}$$

We take a uniform partition of $[0,1]$ into 100 subintervals of length $h = .01$ and build the stiffness matrix for the cG(1) finite element formulation. Now, the Green's function for this problem is given by [93]

$$G(x,\xi) = \begin{cases} (1-\xi)x, \ 0 \leq x < \xi \\ (1-x)\xi, \ \xi \leq x < 1. \end{cases} \tag{4.7.15}$$

We plot $G(x,\xi)$ and the inverse of the stiffness matrix on $[0,1] \times [0,1]$, and their difference in figure 4.4. Once the code constructs the Green's function and computes the other quantities in the kernels of equations (4.5.17) and (4.5.29), it begins with an initial function $z(x)$, computes the value of equation (4.5.17) using gaussian quadrature, then uses that value as the function on which the adjoint operator acts, as per (4.5.29). The intermediate iterates are saved in order that we may compare these with the NLPM. Note that the evaluation of the kernel requires second derivatives of the computed dual solution. We solve the dual problem using cubic Hermite basis elements in order to produce a sufficiently smooth approximation to ϕ for this purpose.

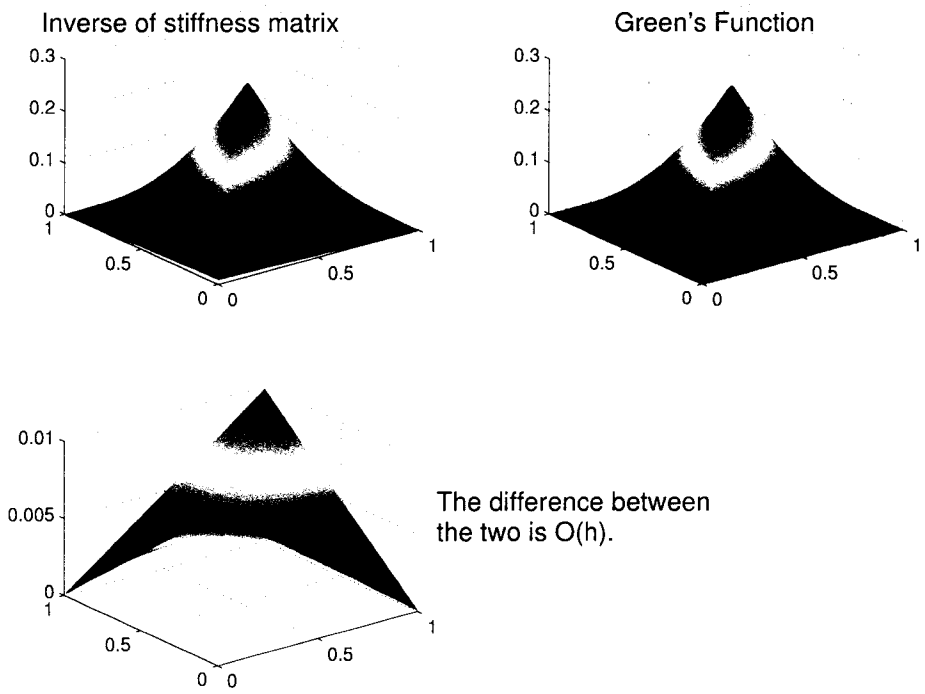

Figure 4.4: Inverted cG(1) stiffness matrix, Green's function, and their difference.

Chapter 5

EXAMPLES

5.1 Numerical examples

We present a series of numerical examples to illustrate the use of the nonlinear power method and to demonstrate the codes based on the algorithms for the NLPM and the Green's function integral representation for $\Phi'_{\psi}(U)$.

Example 5.1.1. We begin by examining the one–dimensional version of the classical Bratu problem

$$\begin{cases} -u''(x) = \lambda e^{u(x)}, & 0 \leq x \leq 1, \\ u(0) = u(1) = 0. \end{cases} \tag{5.1.1}$$

This is the steady–state problem corresponding to the evolution problem

$$u_t - \Delta u = \lambda e^u, \quad x \in D, \quad t > 0, \tag{5.1.2}$$

and is a model for solid fuel ignition [74, 13]. Equation (5.1.1) is simply (4.3.16) with $a \equiv 1$, $b \equiv 0$, and $f = \lambda e^{u(x)}$. The exact solution is given by [5, 21, 31]

$$u(x) = -2 \ln \left(\frac{\cosh\left((x - \frac{1}{2})\frac{\theta}{2}\right)}{\cosh\left(\frac{\theta}{4}\right)} \right), \tag{5.1.3}$$

where θ solves

$$\theta = \sqrt{2\lambda}\cosh\left(\frac{\theta}{4}\right). \tag{5.1.4}$$

There is a unique solution for $\lambda_c = 3.513830719$ (see [21]), and there are two solutions for $\lambda < \lambda_c$. We take $\lambda = 1$, which corresponds to two solutions given by the values

$$\theta_1 = 1.5171645 \quad \text{and} \quad \theta_2 = 10.9387028. \tag{5.1.5}$$

Taking $U_1 = x(1-x)$ as the initial iterate, Newton's method converges to the solution corresponding to θ_1. Using dual data $\psi \equiv 1$, the dual problem corresponding to linearization around the approximate solution U is

$$-\phi''(x) - e^U\phi = 1. \tag{5.1.6}$$

Figure 5.1 displays the forward solutions, error, and dual solution. The thick black curve is the true solution, and the cG(1) finite element solution is the thin yellow curve.

Figure 5.2 displays the results for the matrix reconstruction of $\Phi'_\psi(U)$ (see §4.7.4). $\Phi'_\psi(U)$ is shown in the upper left corner, and the matrix corresponding to $(I - \pi_h)\Phi'_\psi(U)$ is shown in the upper right corner. The dominant left singular vector for this matrix gives the perturbation to U that causes the largest change in $\phi - \pi_h\phi$, where π_h is the nodal interpolant that projects ϕ from $V_h^{(2)}$ into $V_h^{(1)}$. This projection is important for studying *a-posteriori* estimates involving the dual weights $\phi - \pi_h\phi$. Note that the singular vectors shown in the lower left of Figure 5.2 are not the same, indicating that two different perturbations are required to bound estimates involving both ϕ and $\phi - \pi_h\phi$. Since the main focus is the map $\Phi : U \to \phi$, henceforth we only consider $\Phi'_\psi(U)$.

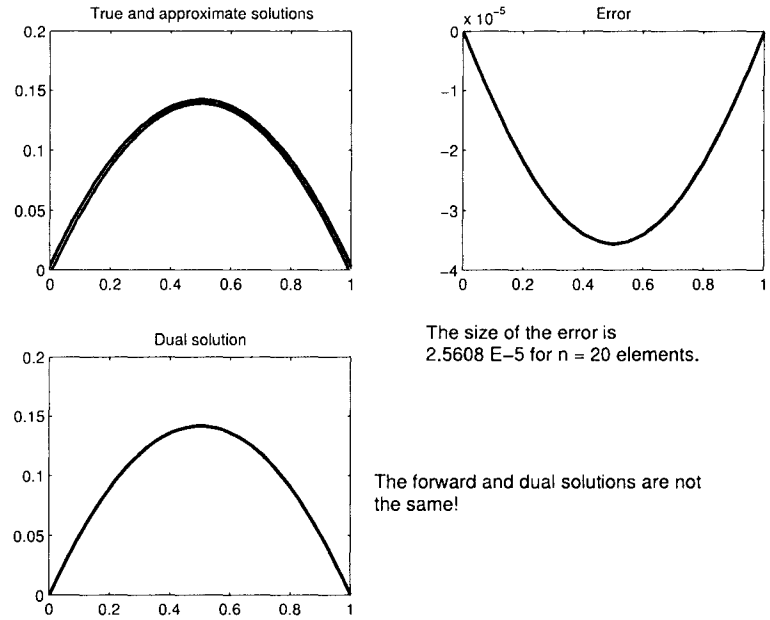

Figure 5.1: Solutions for the 1–d Bratu problem.

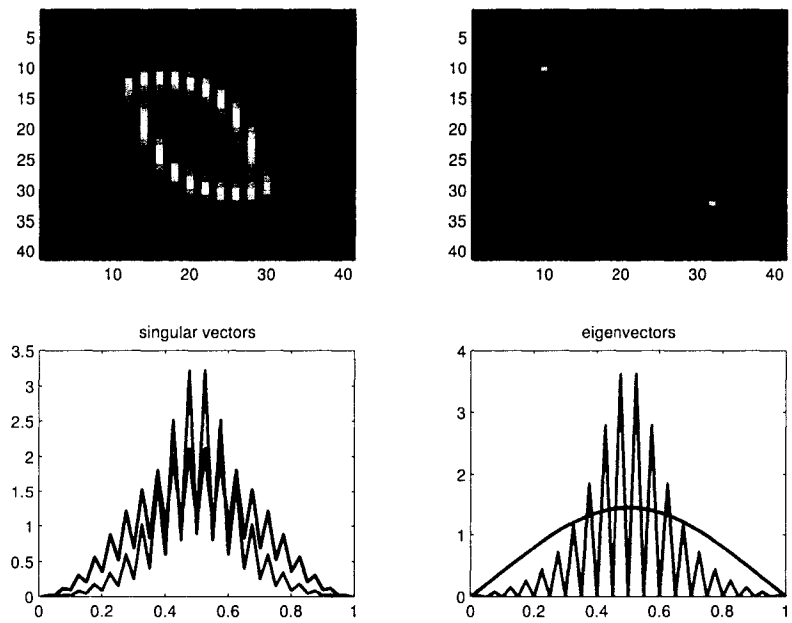

Figure 5.2: $\Phi'_\psi(U)$ and $(I - \pi_h)\Phi'_\psi(U)$, singular vectors, and eigenvectors.

Example 5.1.2. We consider the two–point boundary value problem (4.3.16), with the following coefficients and data:

$$\begin{aligned} a(u,x) &= 1+u \\ b(u,x) &= u \\ f(u,x) &= 2u + u(1-2x) - (1-2x)^2 + 2. \end{aligned}$$

The data f was chosen so that $u(x) = x(1-x)$ solves the problem exactly. We choose dual data

$$\psi(x) = -4x^3 + 6x^2 - 2x + 2, \tag{5.1.7}$$

which makes $\phi(x) = x(1-x)$ the solution to the dual problem corresponding to linearization around the true forward solution u. This way, we can test both the forward and dual solves to determine whether they are producing meaningful approximations. We solve the problem on a very coarse mesh ($n = 4$ elements), and on a finer mesh ($n = 40$ elements) and display the results in Figures 5.3–5.8. Figures 5.3 and 5.4 show plots of the approximate and true forward and dual solutions, as well as numerical approximations of the Green's function for the linearized dual problem, and a reconstruction of the matrix $\Phi'_\psi(U)$. Figures 5.5 and 5.6 display the results of the nonlinear power method; the left hand figure shows the final iterate of the method (vectors are normalized to have L_2 norm equal to one) to determine the dominant eigenvector and dominant left singular vector. Figures 5.7 and 5.8 again show $\Phi'_\psi(U)$, using the basis functions for the space $V_h^{(2)}$ of piecewise quadratic functions, and the basis functions for $V_h^{(1)}$, the space of piecewise linear functions. Notice the "banding" effect that is caused by solving relative to the basis for $V_h^{(2)}$: this is because there are two different

"shape" functions in that basis, and their perturbations lead to qualitatively different dual solutions. The appearance is smoother in $V_h^{(1)}$ since the basis "hat" functions are identical at each node.

Table 5.1 displays the results of some error computations. Since the true solution is known, we can compute the quantities $\|e\| = \|u - U\|$ and $|\langle e, \psi \rangle|$ exactly, and compare these with *a-posteriori* estimates based on the dual solution.

# of elements	$\|e\|$	$\|\langle e, \psi \rangle\|$	estimate 4.3.38	error ratio
$n = 4$	5.738×10^{-3}	1.109×10^{-2}	1.116×10^{-2}	1.00522
$n = 40$	5.644×10^{-5}	1.042×10^{-4}	1.042×10^{-4}	1.00005

Table 5.1: Errors for Example (5.1.2).

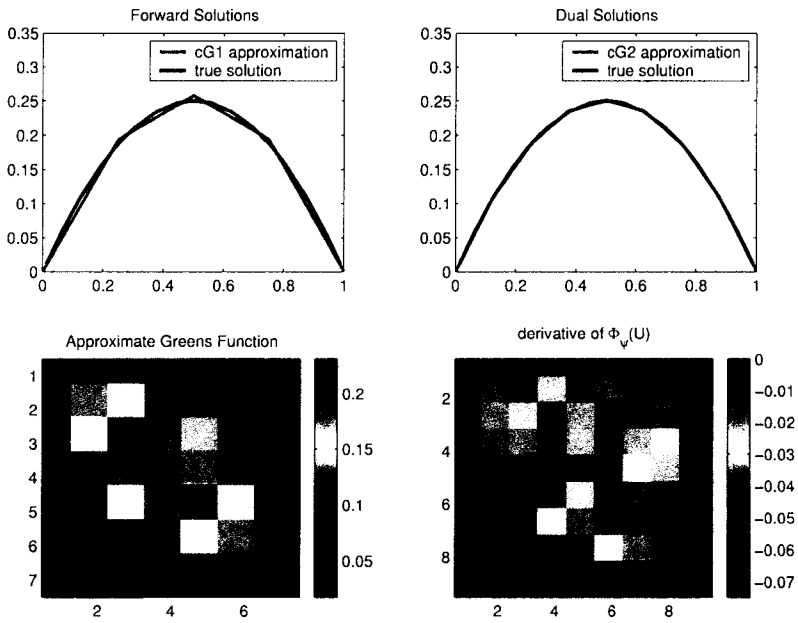

Figure 5.3: Solutions for Example (5.1.2) with $n = 4$ elements.

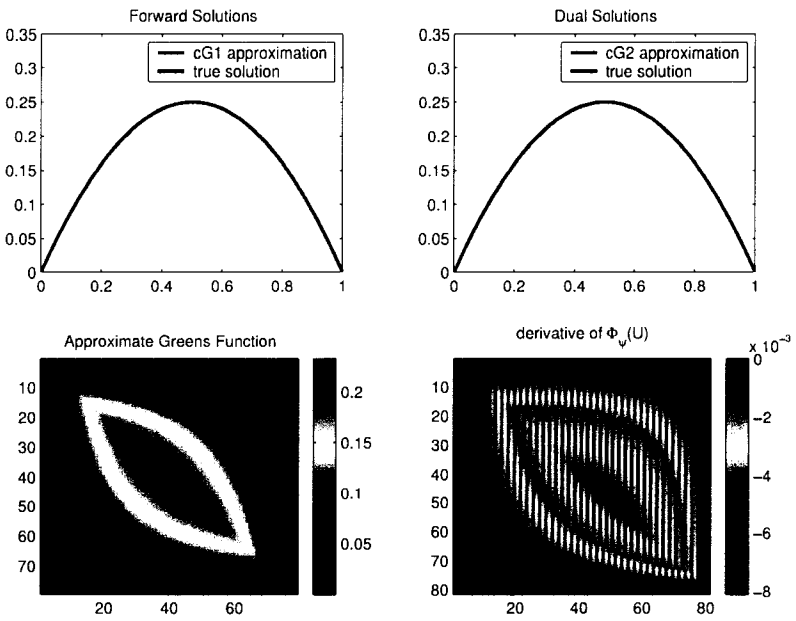

Figure 5.4: Solutions for Example (5.1.2) with $n = 40$ elements.

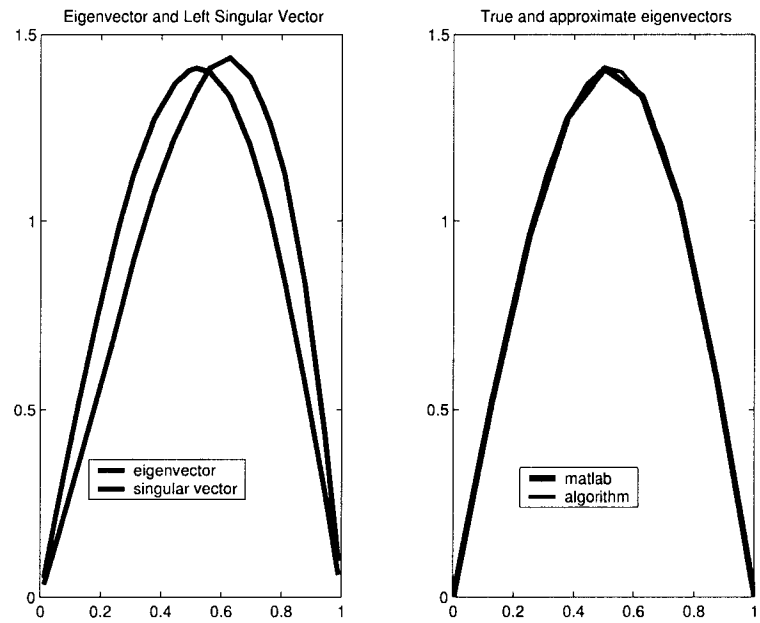

Figure 5.5: NLPM results for Example (5.1.2) with $n = 4$ elements.

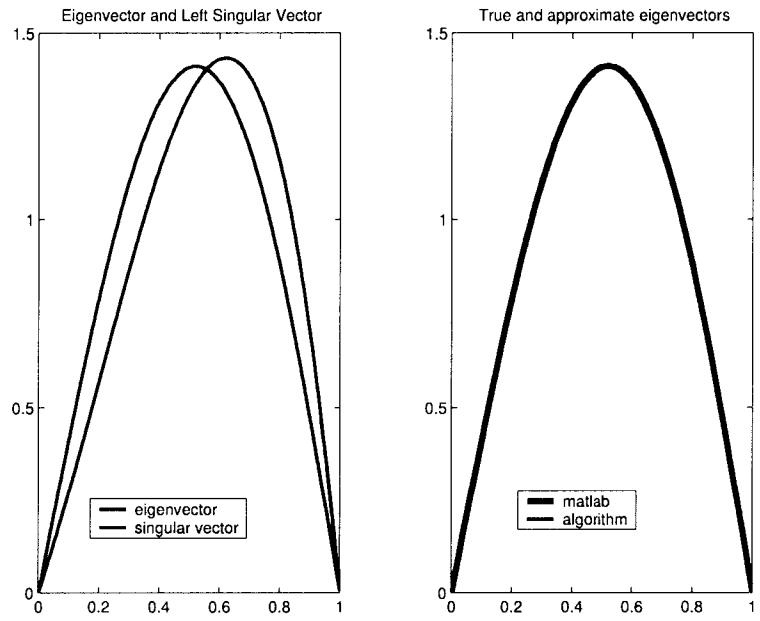

Figure 5.6: NLPM results for Example (5.1.2) with $n = 40$ elements.

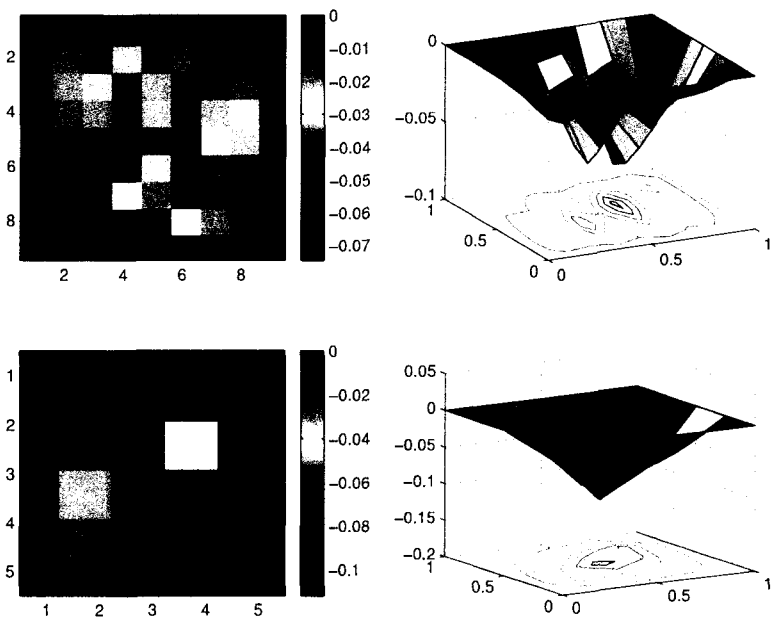

Figure 5.7: $\Phi'_{\psi}(U)$ with cG(2) and cG(1) basis functions, $n = 4$ elements.

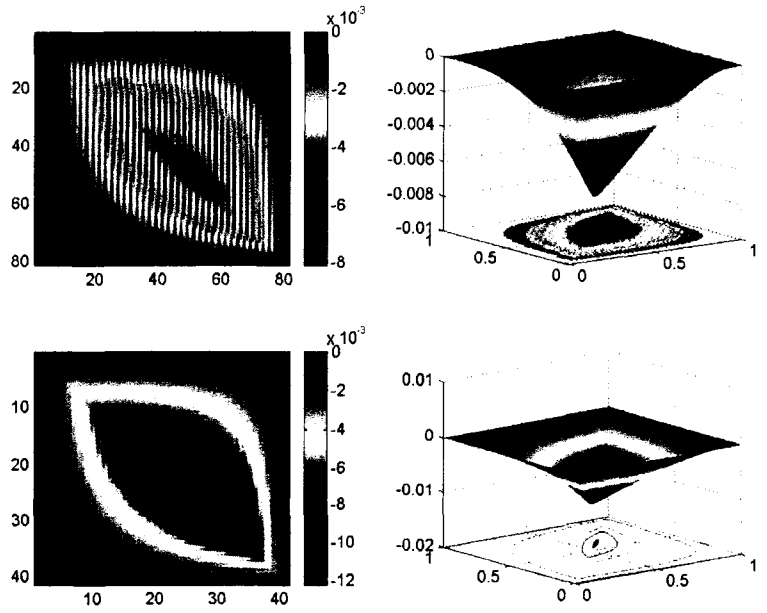

Figure 5.8: $\Phi'_{\psi}(U)$ with cG(2) and cG(1) basis functions, $n = 40$ elements.

Example 5.1.3. In this example, we consider the boundary value problem (4.3.16) with these coefficients:

$$\begin{aligned} a(u,x) &= 1+u \\ b(u,x) &= 0 \\ f(u,x) &= 1+6x-6x^2. \end{aligned}$$

With this choice of f, the exact solution is $u(x) = x(1-x)$. In the first part of this example, we use the dual data

$$\psi(x) = \beta^2\pi^2(1+x-x^2)\sin(\beta\pi x). \tag{5.1.8}$$

This choice makes $\phi(x) = \sin(\beta\pi x)$ the dual solution corresponding to linearizing around u. We begin by taking $\beta = 5$ and show some results based on Algorithm (2.4.1), as well as an algorithm based on formula (4.5.17), utilizing the integral representation for $\Phi'_\psi(U)$ based on the Green's function. Since the true dual solution is known, we are able to check the accuracy of the cG(1), cG(2), and Hermite finite element methods used to solve the dual problem. In Figure 5.9, we display the results for the dual solution corresponding to linearization around the finite element solution. Clearly these plots indicate that the dual solution codes are producing accurate results. Figure 5.10 displays the results of the nonlinear power method, based on Algorithm (2.4.1). The initial starting vector is shown in black, and the final iterate in red. In this example, we used the sequence $\alpha_j = (.75)^j$ for the linearization, and ran the power method until the norm of the perturbations to U became as small as $\sqrt{\epsilon_0}$, where ϵ_0 is the machine number.

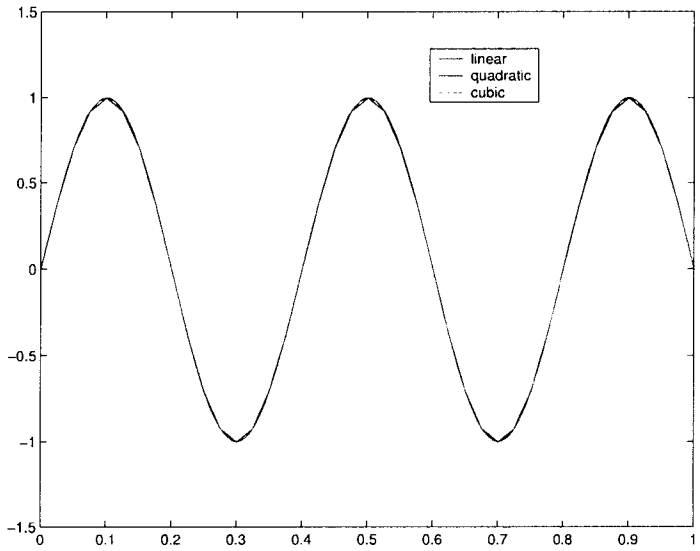

Figure 5.9: Linear, quadratic, and cubic dual solutions for Example (5.1.3), $\beta = 5$.

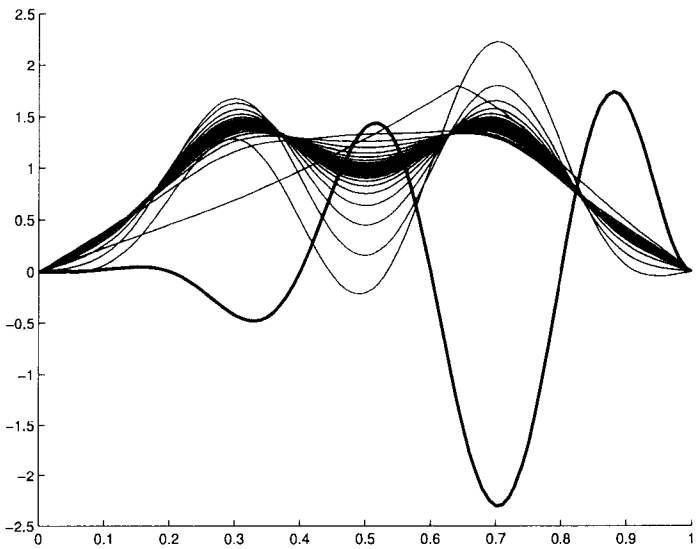

Figure 5.10: Iterates of the NLPM for Example (5.1.3), $\beta = 5$.

Figures 5.11–5.13 compare the final iterates from the NLPM and Green's function integral formula with the eigenvector produced by MATLAB for the matrix reconstruction of $\Phi'_{\psi}(U)$.

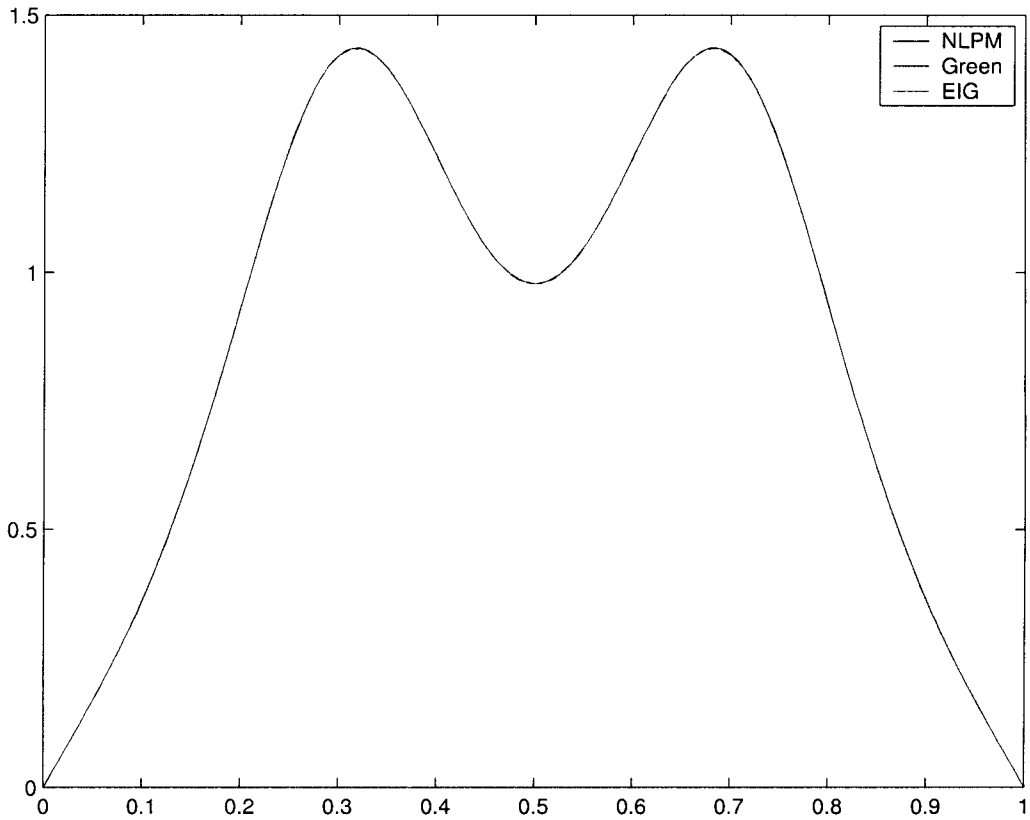

Figure 5.11: Dominant eigenvector for $\Phi'_{\psi}(U)$: NLPM, Green's function formula, and MATLAB.

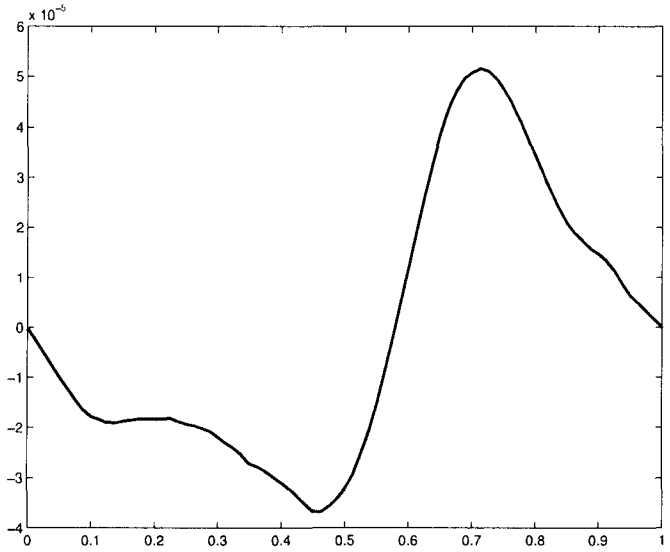

Figure 5.12: Difference between NLPM and MATLAB.

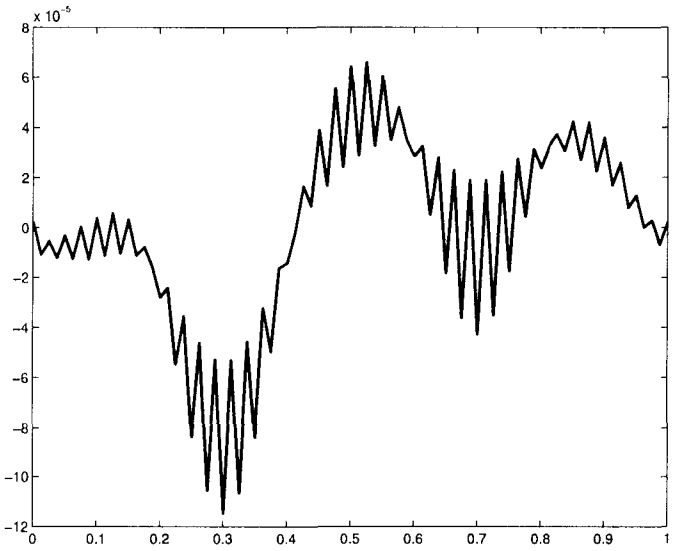

Figure 5.13: Difference between Green's function formula and MATLAB.

Table 5.2 shows the approximations to the dominant eigenvalue for the NLPM, the Green's function formula, and MATLAB's eigen–decomposition of the matrix representation for $\Phi'_\psi(U)$. We also show the approximation to $\|\Phi'_\psi(U)\|$ based on the Green's function formula coupled with the formula (4.5.29), as well as from the SVD for the matrix $\Phi'_\psi(U)$.

	NLPM	Green's	MATLAB
eigenvalue	3.306488	3.306171	3.305899
singular value		14.54759	15.32457

Table 5.2: Dominant eigenvalue and singular value.

Example 5.1.4. We continue with the problem given in Example 5.1.3, with some differences. In the previous example, the dual data was not normalized, making it inappropriate for the purposes of error estimation. Here, we use the same formula for the dual data, but scaled to have unit norm: we take

$$p(x) = \beta^2\pi^2(1 + x - x^2)\sin(\beta\pi x) \tag{5.1.9}$$

and then use $\psi(x) = \dfrac{p(x)}{\|p(x)\|}$ as the dual data. This has the effect of scaling the dual solution by approximately $(\pi\beta)^{-2}$, so the size of the dual solution decreases fairly rapidly with increasing β. We vary the parameter β and examine what effects this may have on the sensitivity of the problem to linearization. From equation (4.5.28), we conjecture that dual solutions with large second derivatives may indicate a sensitivity to linearization.

We take $n = 40$ elements and solve both the forward and dual problems in the space $V_h^{(1)}$ of piecewise linear polynomials. The error $\epsilon \approx \|u - U\| = 1.55 \times 10^{-4}$ for this discretization. We construct the matrix representation of the operator $\Phi_\psi(U)$ and use MATLAB to perform both singular value and

eigen–decompositions and display results in Table 5.3. Recall that $\Phi(U)$ and $\Phi(u)$ are the dual solutions corresponding to linearization around the approximate and true solutions, respectively, while $\Phi(U + \epsilon v)$ is the dual solution corresponding to linearization around a perturbation of size ϵ in the direction of a unit vector v. In particular, if v is the dominant left singular vector for $\Phi'_\psi(U)$, we are linearizing in the direction of maximum possible change for $\Phi(U)$.

Note that the size of the changes in $\|\Phi(U+\epsilon v)-\Phi(U)\|$ are not altered significantly with increases in β, but remain about one order of magnitude smaller than the forward error. However, it is significantly larger than the corresponding change $\|\Phi(u)-\Phi(U)\|$, in which the linearization is perturbed in the direction of the true solution. Additionally, the values for $\|\Phi'_\psi(U)\|$ do not change considerably with β. These results indicate that the linearization is not sensitive to perturbations in the forward solution. Moreover, the bound based on the norm of the derivative is considerably larger (as much as three orders of magnitude when $\beta = 15$) than the effect of linearization in the direction of the true solution, as indicated by comparing the second and third columns in Table 5.3.

Differences become more apparent when we examine the relative change in the dual solution. The fifth column in the table lists the size of the change in the dual solution relative to the size of the dual solution obtained by linearizing around the approximate solution U. The column in the table labelled "ratio" gives the ratio of the relative change in ϕ to the relative change in U. Since we solve the forward problem on the same mesh for each value of β, the relative change in U is constant, so the final two columns in the table are directly proportional. In the situation where the size of the

perturbation to U is the size of the forward error, we have

$$\begin{aligned} \text{ratio} &= \left(\frac{\|\Phi(U+\epsilon v)-\Phi(U)\|}{\|\Phi(U)\|}\right)\left(\frac{\|U\|}{\|u-U\|}\right) \\ &= \left(\frac{\|\Phi(U+\epsilon v)-\Phi(U)\|}{\epsilon}\right)\left(\frac{\|U\|}{\|\Phi(U)\|}\right) \\ &\approx \|\Phi'_\psi(U)\|\frac{\|U\|}{\|\Phi(U)\|}. \end{aligned} \tag{5.1.10}$$

We see that $\|\Phi'_\psi(U)\|$ is the relative condition number for Φ_ψ. It relates the relative size of the forward and dual solutions to the ratio of relative changes in the forward and dual solutions. Since the relative error in the forward solution is constant, and $\|\Phi'_\psi(U)\|$ does not change significantly, and the increasing ratios in the last column of Table 5.3 just reflect the fact that the size of the dual solution decreases with increasing β.

β	$\|\Phi(U+\epsilon v)-\Phi(U)\|$	$\|\Phi(u)-\Phi(U)\|$	$\|\Phi'_\psi(U)\|$	$\frac{\|\Phi(U+\epsilon v)-\Phi(U)\|}{\|\Phi(U)\|}$	ratio
1	1.28×10^{-5}	3.78×10^{-6}	8.29×10^{-2}	1.55×10^{-4}	.182
5	1.11×10^{-5}	2.06×10^{-7}	7.03×10^{-2}	3.19×10^{-3}	3.76
10	1.11×10^{-5}	3.70×10^{-8}	6.77×10^{-2}	1.28×10^{-2}	15.1
15	1.10×10^{-5}	1.68×10^{-8}	6.35×10^{-2}	2.87×10^{-2}	33.7

Table 5.3: Quantities of interest for Example (5.1.4).

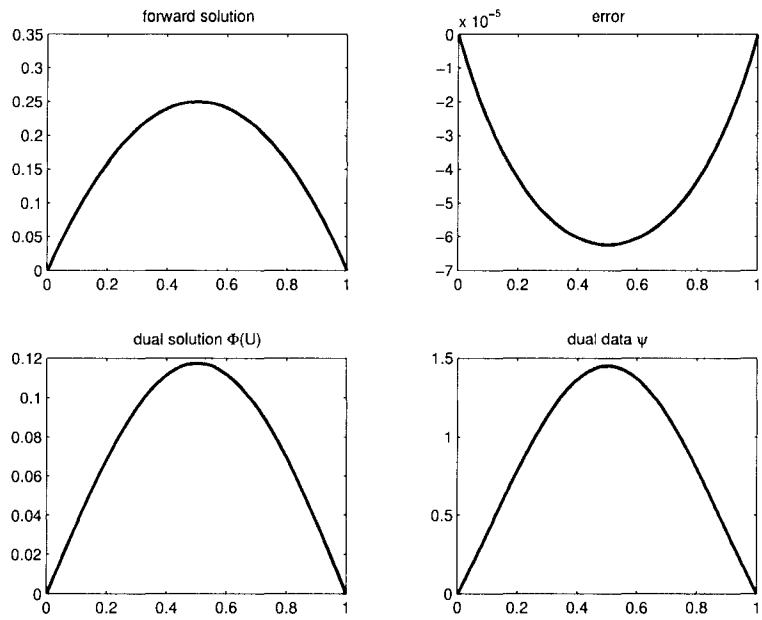

Figure 5.14: Example (5.1.4), $\beta = 1$.

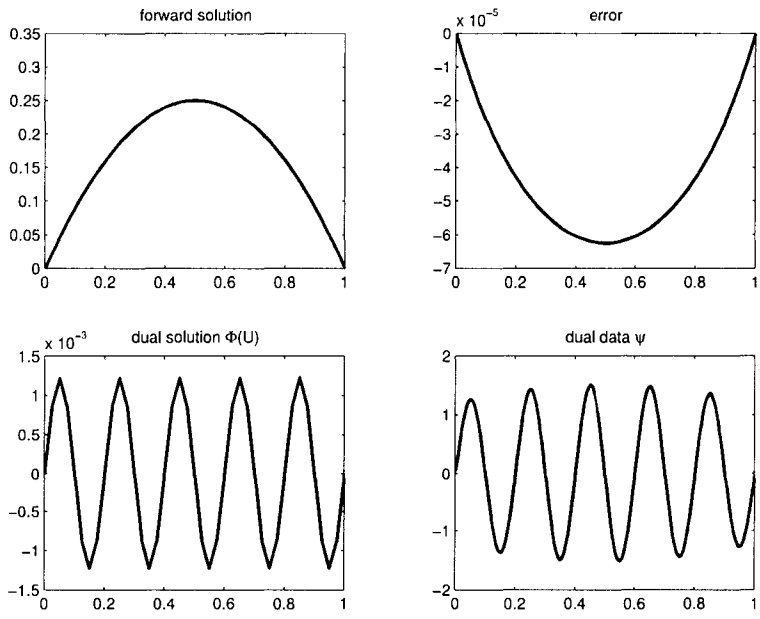

Figure 5.15: Example (5.1.4), $\beta = 10$.

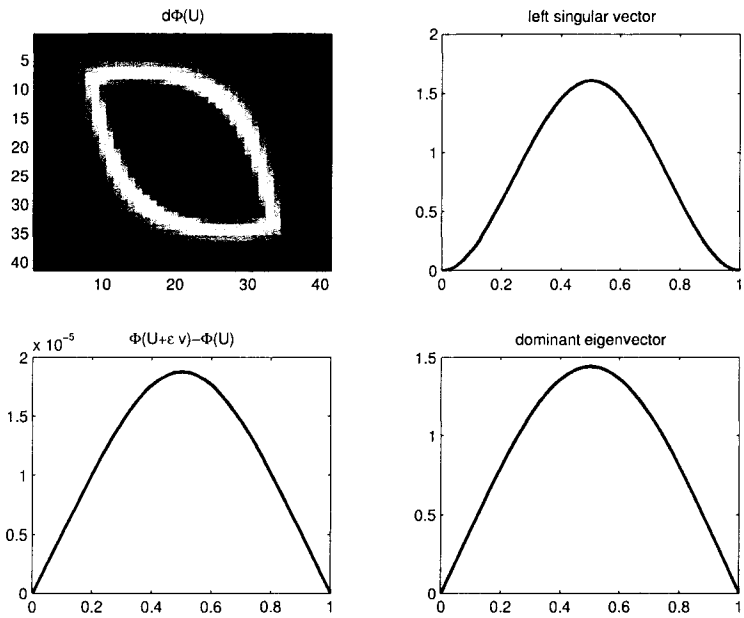

Figure 5.16: Example (5.1.4), $\beta = 1$.

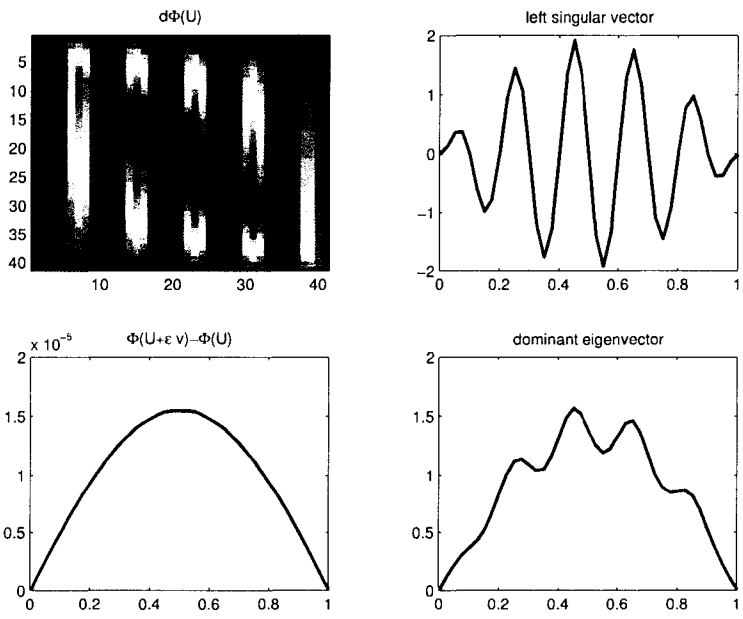

Figure 5.17: Example (5.1.4), $\beta = 10$.

Example 5.1.5. In this example, we consider the boundary value problem (4.3.16) with these coefficients:

$$\begin{aligned} a(u,x) &= \beta \tan^{-1}(u) + \frac{\beta\pi}{2} + \alpha \\ b(u,x) &= 0 \\ f(u,x) &= -\frac{4\pi^2\beta B^2 \cos(2\pi x)^2}{1 + B^2 \sin(2\pi x)^2} \\ &\quad +4\pi^2\beta B \sin(2\pi x)\left(\tan^{-1}(B\sin(2\pi x)) + \frac{\beta\pi}{2}\right) - 4\pi^2\alpha B \sin(2\pi x). \end{aligned}$$

The true solution for this problem is $u = -B\sin(2\pi x)$. We can alter the parameters B, β and α and examine their effects on linearization. The parameters B and β influence the steepness of the graphs of $a(u,x)$ and $u(x)$. In particular, larger values for these parameters make a large, sharp spike in the convection term for the linearized dual problem. The term $\frac{\beta\pi}{2}$ ensures that $a(u,x)$ remains bounded away from zero, and the constant α linearly adjusts $a(u,x)$, in order that we can examine the effect of the proximity of $a(u,x)$ to zero.

In this example, the relatively bad behavior of the coefficient $a(u,x)$ and the solution $u(x)$ for large values of the parameters make the forward problem difficult to solve. We find that Newton's method does generally not converge, so instead we construct a piecewise linear approximation to the true solution by simply using the nodal values of $u(x)$ and interpolating linearly. This is evidenced by the "oscillatory" error plots in Figure 5.19.

Results for various values of B and β are displayed in Table 5.4 for $n = 20$ elements and $\alpha = 0$. Plots for coefficients, solutions, and $\Phi'_\psi(U)$ are displayed in Figures 5.18–5.21 for the values $\beta = 5$ and $B = 10$. Note the term $A^{-\top}\psi$ in the fifth column of Table 5.4. This refers to the dual solution corresponding to linearization around the appropriate average of u and U

(see equation 4.3.14). We construct this average by writing the true dual problem (4.3.27) in weak form and using gaussian quadrature to compute the coefficients $\tilde{a}, \tilde{\alpha}, \tilde{b}$, and $\tilde{f}$.

(β, B)	$\|u - U\|$	$\|\Phi(U + \epsilon v) - \Phi(U)\|$	$\|\Phi(u) - \Phi(U)\|$	$\|A^{-\top}\psi - \Phi(U)\|$	$\|\Phi'_\psi(U)\|\epsilon$
$(2, 3)$	1.91×10^{-2}	6.18×10^{-4}	3.54×10^{-4}	1.76×10^{-4}	6.55×10^{-4}
$(3, 2)$	1.27×10^{-2}	2.59×10^{-4}	1.45×10^{-4}	7.25×10^{-5}	2.75×10^{-4}
$(5, 10)$	6.35×10^{-2}	8.78×10^{-4}	5.11×10^{-4}	2.55×10^{-4}	9.39×10^{-4}
$(10, 5)$	3.18×10^{-2}	2.15×10^{-4}	1.24×10^{-4}	6.21×10^{-5}	2.29×10^{-4}

Table 5.4: Quantities of interest for Example (5.1.5).

We emphasize that the proximity of the true and approximate solutions is needed to ensure the bound (4.4.5) holds. With $\beta = 2, B = 20, \alpha = 0$ and $n = 20$ elements, the error is $\|u - U\| = 0.127$, and the appropriately scaled norm of $\Phi'_\psi(U)$ does not bound the difference $\|\Phi(U + \epsilon v) - \Phi(U)\|$. Refining the mesh brings the true and approximate solutions closer together, and we see in Table 5.5 that the bound holds.

n	(β, B)	$\epsilon = \|u - U\|$	$\|\Phi(U + \epsilon v) - \Phi(U)\|$	$\|\Phi'_\psi(U)\|\epsilon$
20	$(2, 20)$	1.27×10^{-1}	6.84×10^{-3}	6.71×10^{-3}
60	$(2, 20)$	1.42×10^{-2}	5.05×10^{-4}	5.38×10^{-4}

Table 5.5: Equation (4.4.5) is an asymptotic result.

The distance from $a(u, x)$ to zero appears to have a fairly small effect on linearization as well. We again "solve" the forward problem by using the piecewise linear interpolant of the true solution on $n = 40$ elements. We solve the linearized dual problem with $\alpha = 0$ and $\alpha = 10$ and record the results in Table 5.6. Even when the coefficient $a(u, x)$ is close to zero, the dual solution changes by less than one percent.

α	$\min(a(U,x))$	$\|\Phi(U+\epsilon v)-\Phi(U)\|$	% change	$\|A^{-\mathrm{T}}\psi-\Phi(U)\|$	% change
0	.05	2.19×10^{-3}	.33	6.47×10^{-4}	.09
10	10.05	3.68×10^{-6}	.04	2.29×10^{-8}	.0028

Table 5.6: Effect of distance from $a(u,x)$ to zero.

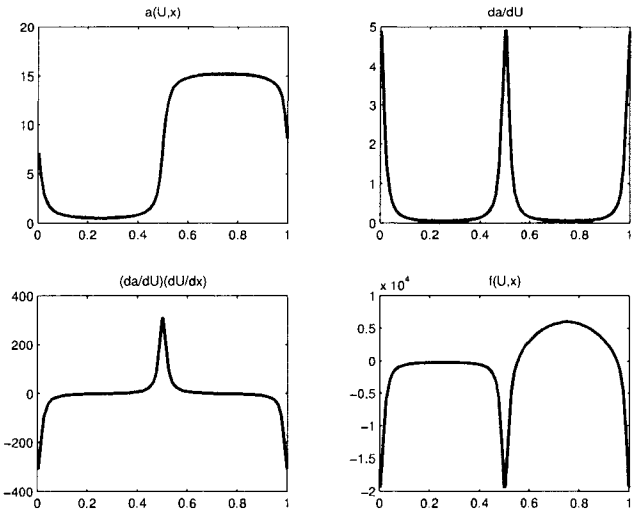

Figure 5.18: Coefficients for Example (5.1.5); $\beta=5, B=10$.

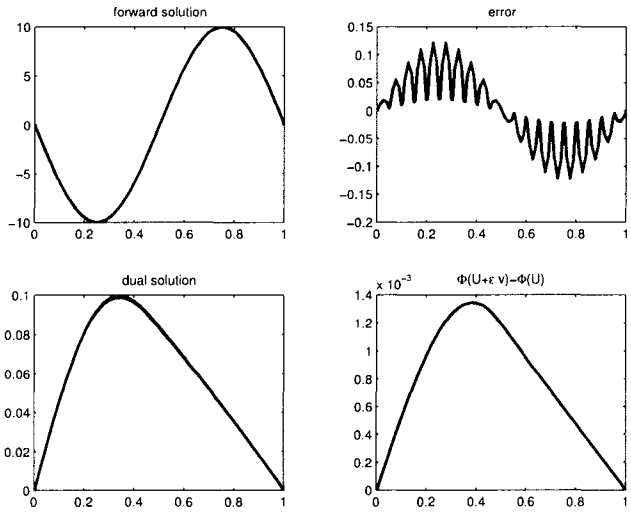

Figure 5.19: Solutions for (5.1.5); $\beta=5, B=10$.

Figure 5.20 displays images of the reconstructed matrix $\Phi'_\psi(U)$. the left-hand column shows the result when $\Phi : V_h^{(2)} \rightarrow V_h^{(2)}$. In this situation, we use the fact that $U \in V_h^{(1)} \subset V_h^{(2)}$ and we express U as a function in $V_h^{(2)}$ by adding appropriate nodal values. The "banding" in the matrix representation for $\Phi'_\psi(U)$ is due to the basis functions that we use for the finite element solution in $V_h^{(2)}$. We also believe this is the reason for the jagged shape of the dominant singular vector shown in the lower left–hand figure. The right–hand column shows the result when we solve both the forward and dual problems in $V_h^{(1)}$.

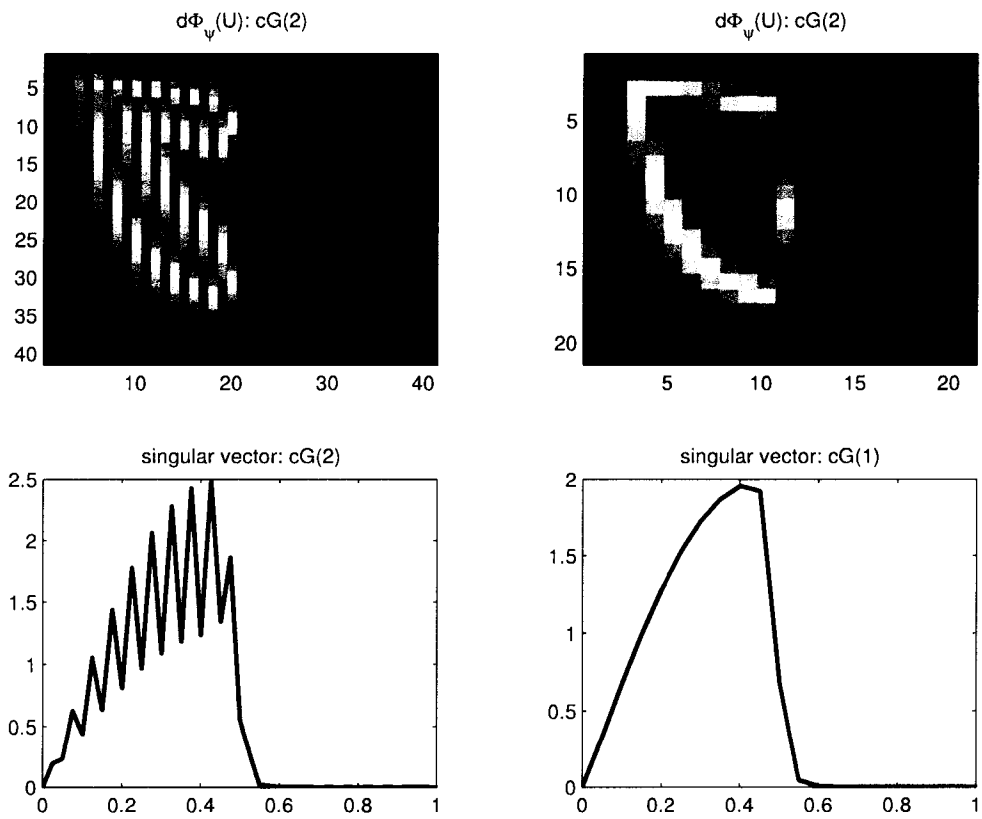

Figure 5.20: $\Phi'_\psi(U)$ and singular vectors, cG(2) and cG(1) methods.

Figure 5.21 shows how the dual solution changes when we perturb the forward solution in various directions. All of the perturbations are equally scaled. The thick black line curve the change when U is perturbed in the direction of the dominant singular vector for $\Phi'_\psi(U)$. The blue curve is for a perturbation in the direction from U to the true solution U, and the red curve indicates the "true" error of linearization, based on $A^{-\top}\psi$ described earlier. We also show the result of adding a "random" smooth perturbation to the forward solution in green. For this example, U was perturbed in the direction $x(1-x)\sin(3\pi x)$.

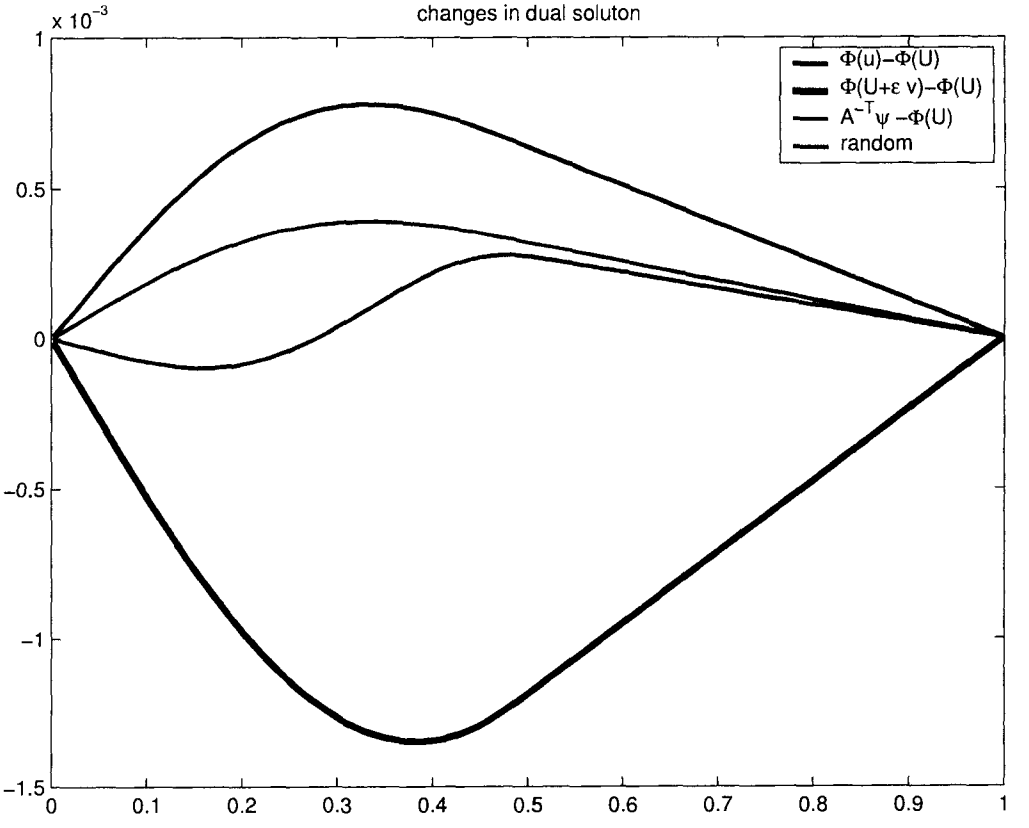

Figure 5.21: Changes in dual solution for Example (5.1.5).

Example 5.1.6. This is an example of a problem with a solution that has a steep layer. We take

$$\begin{aligned}
a(u,x) &= 1+u,\\
b(u,x) &= 0,\\
f(u,x) &= -(1-x)\tanh(\beta\,(x-\alpha)\\
&\quad +x\tanh(\beta(x-\alpha))+x(1-x)(\beta-\beta\tanh(\beta(x-\alpha))^2)^2\\
&\quad -(1+u)(-2\tanh(\beta(x-\alpha))\\
&\quad +2\beta(1-x)(1-\tanh(\beta(x-\alpha)))^2\\
&\quad -2x\beta(1-\tanh(\beta(x-\alpha))^2)\\
&\quad -2x\beta^2(1-x)\tanh(\beta(x-\alpha))(1-\tanh(\beta\,(x-\alpha)))^2.
\end{aligned}$$

With these coefficients and data, the true solution is

$$u(x) = x(1-x)\tanh(\beta(x-\alpha)). \tag{5.1.11}$$

The parameters α and β determine the position and steepness of the layer in the solution. We use $n = 40$ elements and solve the problem for $\beta = 5, 10, 15$ and 20. We take $\alpha = 0.5$ in each case. Figure 5.22 depicts the true forward solution in black, and the finite element solutions as dotted green lines for $\beta = 5, 10$ and 15, and a solid red line for $\beta = 20$. It is evident that Newton's method has difficulty solving the forward problem with increasing β, and the final iterate for $\beta = 20$ is a very inaccurate approximation to the true solution.

In Figure 5.23, we plot the dual solutions corresponding to the forward solves in Figure 5.22 with dual data $\psi \equiv 1$. The thick black curve is the dual solution $\Phi(U)$ corresponding to linearization around U, the red curve

corresponds to $\Phi(U + \epsilon v)$, where $\epsilon = \|u - U\|$ is the size of the error, and v is the normalized, dominant left singular vector for $\Phi'_{\psi}(U)$. The green curves show the dual solution corresponding to linearization around the true integral average of u and U, as given by (4.3.14).

Table 5.7 displays the changes in the forward and dual solutions. Here, we linearized in the dominant singular direction as well as the theoretically correct direction. The maximum value of the derivative increases linearly with β. Since Newton's method did not converge for large β, we constructed the approximate solution U for $\beta = 100$ by evaluating the true solution at the nodes and randomly perturbing the values near the layer. The perturbations are roughly one order of magnitude smaller than the values of U, and the difference between u and U is shown in Figure 5.25. For $\beta = 100$, we used 100 elements. Notice that the change in the dual solution remains about one order of magnitude smaller than $\|u - U\|$ when we linearize in the direction of the dominant singular vector, but is considerably smaller when linearized around the integral average. This is fairly striking evidence that computational errors due to steep layers in the solution have very little effect on linearization.

β	$\lvert u'(\alpha)\rvert$	$\|u - U\|$	$\|\Phi(U + \epsilon v) - \Phi(U)\|$	$\|A^{-\top}\psi - \Phi(U)\|$
5	2.5	3.68×10^{-6}	6.47×10^{-4}	2.29×10^{-8}
10	1.25	1.97×10^{-4}	2.23×10^{-5}	8.99×10^{-6}
15	3.75	1.91×10^{-3}	2.36×10^{-4}	1.48×10^{-4}
20	5	1.58×10^{-1}	5.65×10^{-2}	1.64×10^{-2}
100	25	2.28×10^{-3}	3.01×10^{-4}	4.29×10^{-6}

Table 5.7: Changes for Example (5.1.6).

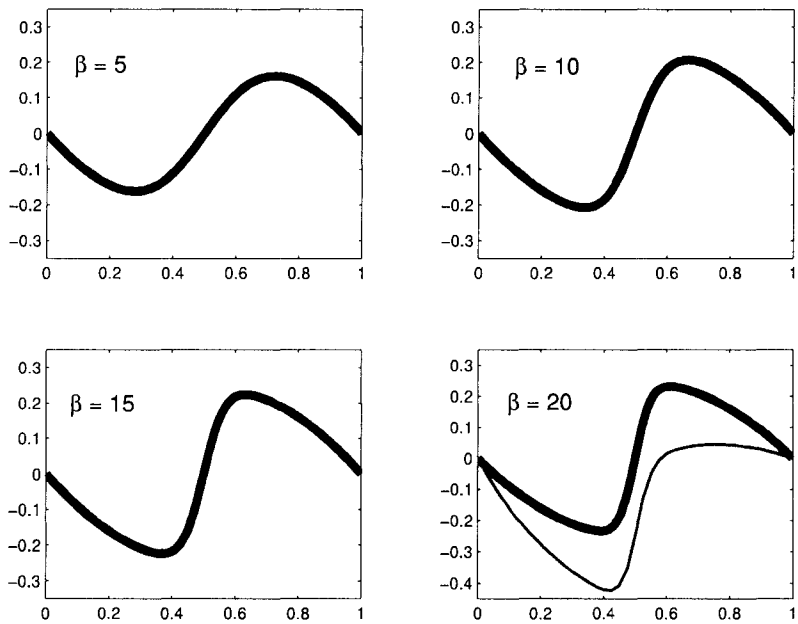

Figure 5.22: Forward solutions for Example (5.1.6).

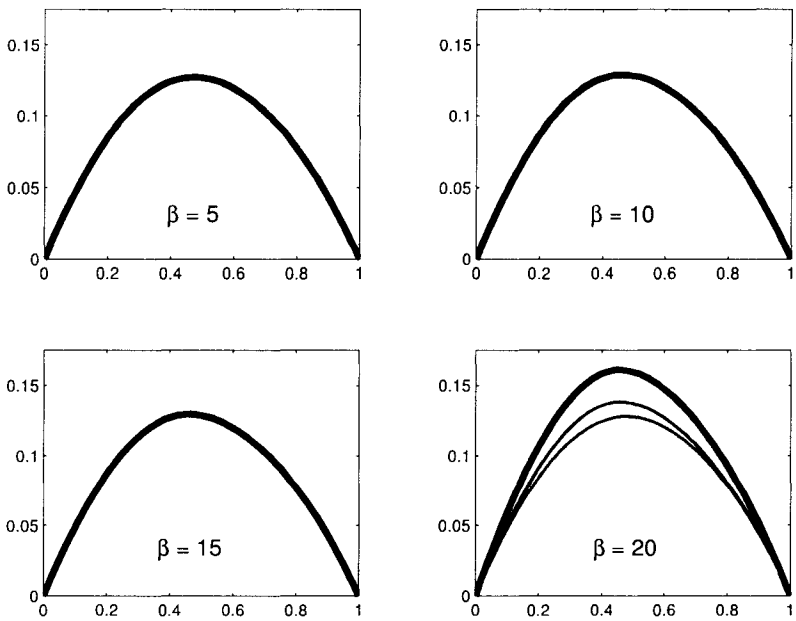

Figure 5.23: Dual solutions for Example (5.1.6).

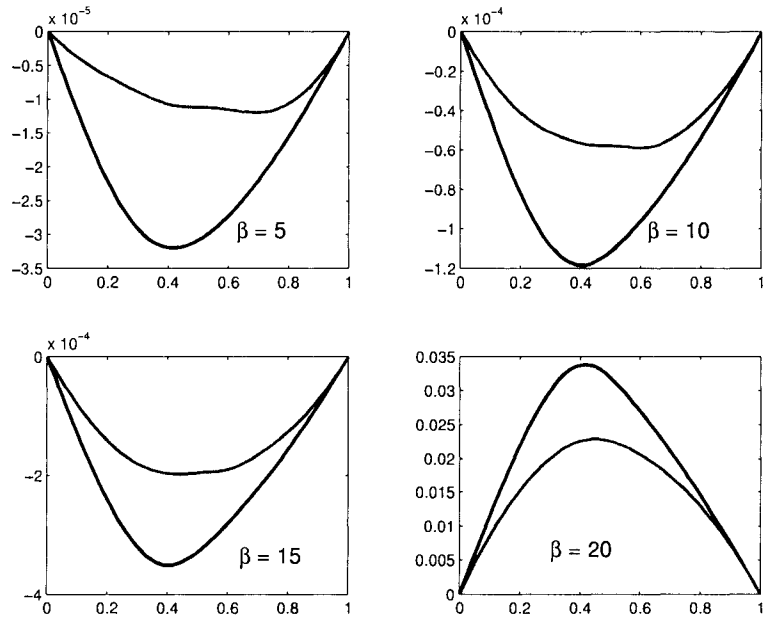

Figure 5.24: Changes in the dual solutions for Example (5.1.6).

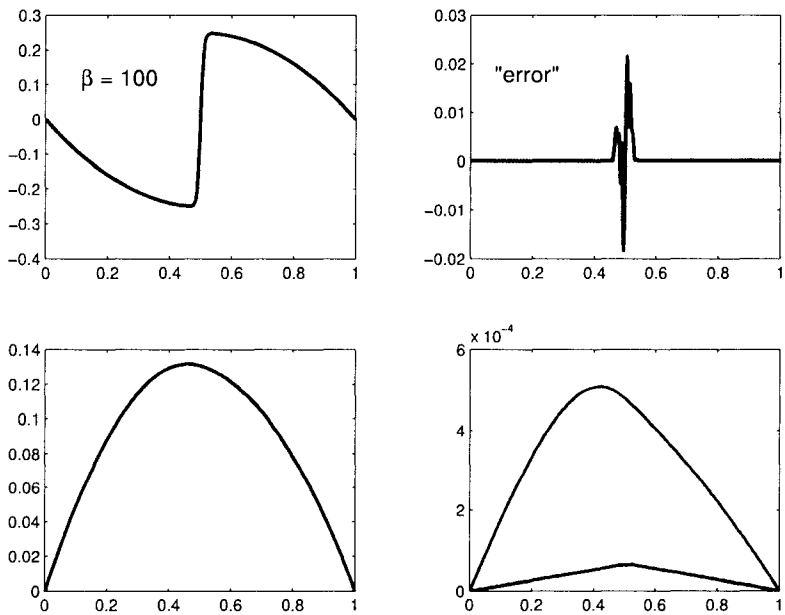

Figure 5.25: Contrived solution for $\beta = 100$.

Example 5.1.7. The preceding examples indicate that linearization error is not a serious issue for many elliptic two-point boundary value problems. Dual solutions for problems with ill-behaved coefficients and steep or oscillatory forward solutions do not exhibit much sensitivity to perturbations in the approximate solution around which we linearize. This was confirmed by the small values of $\|\Phi'_\psi(U)\|$ that we computed in each example. We now give a very simple example of a problem for which the dual is highly sensitive to perturbations in the forward solution. This 2×2 example is nearly identical to Example 4.4.2 in §4.4.1. In that example, we considered the nonlinear 2×2 system of equations $f = 0$, where f was given by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \frac{x_1^2}{2} - \frac{1}{2} \\ \frac{x_2^3}{3} - \frac{1}{3} \end{bmatrix}, \tag{5.1.12}$$

for which $x = [1, 1]^\top$ was an exact solution. Recall $\Phi'_\psi(X)$ for that example:

$$\Phi'_\psi(X) = \begin{bmatrix} -\frac{\psi_1}{x_1^2} & 0 \\ 0 & -\frac{2\psi_2}{x_2^3} \end{bmatrix}. \tag{5.1.13}$$

It is clear why this problem is not sensitive to linearization: the components of $\Phi'_\psi(X)$ do not change drastically with small changes in x_1 and x_2 when x_1 and x_2 are both near 1. Next, we consider the nonlinear equation $f = 0$, but modify the function f so that

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \frac{(x_1 - .9)^2}{2} - \frac{1}{200} \\ \frac{x_2^3}{3} - \frac{1}{3} \end{bmatrix}. \tag{5.1.14}$$

Again, $[1, 1]^\top$ solves the system, but now

$$\Phi'_\psi(X) = \begin{bmatrix} -\frac{\psi_1}{(x_1 - .9)^2} & 0 \\ 0 & -\frac{2\psi_2}{x_2^3} \end{bmatrix}, \tag{5.1.15}$$

and it is apparent that the dual problem is sensitive to perturbations in the first component of the approximate solution. For instance, if we compute an approximate solution $X = [.95\ .95]^\top$ and use dual data to estimate the average error, we find that the dual solution $\|\Phi(X)\| = 14.16$, and $\|\Phi'_\psi(X)\| = 282.85$. Thus, while the dual solutions for boundary value problems are generally much smaller than the corresponding forward solutions, here the opposite is true. It is interesting to note the role of the dual data: if we were only interested in the error in the second component, we would choose $\psi = [0, 1]^\top$, and in this case $\|\Phi'_\psi(X)\| = 2.33$, since this direction is "blind" to the ill–behavior in the first component.

Chapter 6

CONCLUSIONS

The power method is a standard technique in numerical linear algebra for approximating the dominant eigenpair of a matrix. Algorithms for implementing the power method are simple and efficient, since the only requirement is knowledge of the matrix action. In this thesis, we have extended the power method to approximate the dominant eigenpair of the linearization of a nonlinear map. Our implementation is also matrix–free, since we approximate the action of the derivative by computing the function differences

$$F(x+d) - F(x) \approx F'(x)d. \tag{6.0.1}$$

Thus, it is not necessary to explicitly compute the derivative $F'(x)$ in order to estimate the dominant eigenpair. This is a great advantage when the dimension of the problem is large, since computing function differences is relatively inexpensive, while computing $F'(x)$ may be impractical or even impossible. We developed the NLPM by observing that the higher order term in the definition of the linearization can be viewed as a perturbation to the derivative, which decreases to zero as the iteration proceeds:

$$\begin{aligned} F(x+d) - F(x) &= F'(x)d + o(\|d\|) && (6.0.2) \\ &= [F'(x) + E_d]d, && (6.0.3) \end{aligned}$$

where $E_d \to 0$ as $\|d\| \to 0$. In Chapter 2, we developed the link between the perturbed power method and the nonlinear power method and showed that the process converges to the dominant eigenpair, regardless of the dimension. Chapter 3 was an application of the NLPM to determining the stability of a nonlinear system of ordinary differential equations. We included examples of the NLPM that incorporated inverse iteration and spectral enhancement. The NLPM is well suited to applications in which the sign of the dominant eigenvalue determines the stability. Additionally, the analysis of the method is relatively straightforward because the relationship between the system under consideration and the system that we solve is fairly direct.

The idea for the NLPM initially arose in the context of *a-posteriori* error estimates for nonlinear differential equations. Given a problem (in this thesis, we examined nonlinear elliptic problems of the type given by (4.3.16)), the idea was to implement a matrix–free algorithm for finding the perturbation to a solution U that caused the largest possible change in the solution ϕ to the dual problem corresponding to linearization around U. In this way, we could bound the effect of linearization on the problem. We defined the map Φ_ψ between the approximate solution and corresponding linearized dual solution in Chapter 4 with a view to understanding how linearization affects the dual solution. We saw that the linearization issue is related to the derivative $\Phi'_\psi(U)$, and computing the norm of this map requires the dominant singular value and left singular vector rather than the dominant eigenpair. In case $\Phi'_\psi(U)$ is nonnormal, we cannot recover $\Phi'_\psi(U)$ in a matrix–free fashion, since this requires information about the operator and its transpose. However, by numerically reconstructing a matrix representation for $\Phi'_\psi(U)$, we are still able to estimate $\Phi'_\psi(U)$.

It is interesting to note that the map Φ_ψ is quite complicated: its calculation requires the solution of two differential equations, and its properties depend on the coefficients and data for both problems. Even in cases where both the forward and dual problems are relatively simple, Φ_ψ and its derivative $\Phi'_\psi(U)$ are not. The map $\Phi'_\psi(U)$ is nonnormal in general, regardless of whether the forward and/or dual problems are self–adjoint. For instance, any forward problem (4.3.16) with $a'(U)U' = b(U)$ will have a linearized dual problem which is self–adjoint (see equation (4.3.29)), yet the map $\Phi'_\psi(U)$ is still nonnormal. Consider the equation

$$-\big((1+u)u'\big)' + (1-2x)u' + 2u - 2 = 0; \quad u(0) = u(1) = 0. \qquad (6.0.4)$$

The exact solution is $u = x(1-x)$, and the dual problem is self–adjoint. However, the plots in Figure 6.1 show that the dominant singular vector and dominant eigenvector are different. For these plots we used $n = 20$ elements, and compute $\|\Phi'_\psi(U)\| = 0.1276$, while the dominant eigenvalue $|\lambda| = 0.1208$. Now, the linearized dual problem is not self–adjoint (the dual problem corresponding to linearization around the true solution u is self–adjoint). If $\Phi'_\psi(U)$ were normal, we would expect to see the singular vector and eigenvector (as well as $\|\Phi'_\psi(U)\|$ and the dominant eigenvector) coalesce as we refine the mesh. However, this is not the case. For this problem, a refinement to $n = 70$ elements gave $\|\Phi'_\psi(U)\| = 0.1274$ and $|\lambda| = 0.1206$. The 3–d plot labelled $G - G^\top$ in the lower left–hand corner shows the difference between the reconstructed matrix for the Green's function and its transpose. Note that the scale is on the order of 10^{-4}.

An interesting aspect of the reconstructed operator $\Phi'_\psi(U)$ is the structure of the dominant singular vectors (see figure 5.20). Relative to the basis

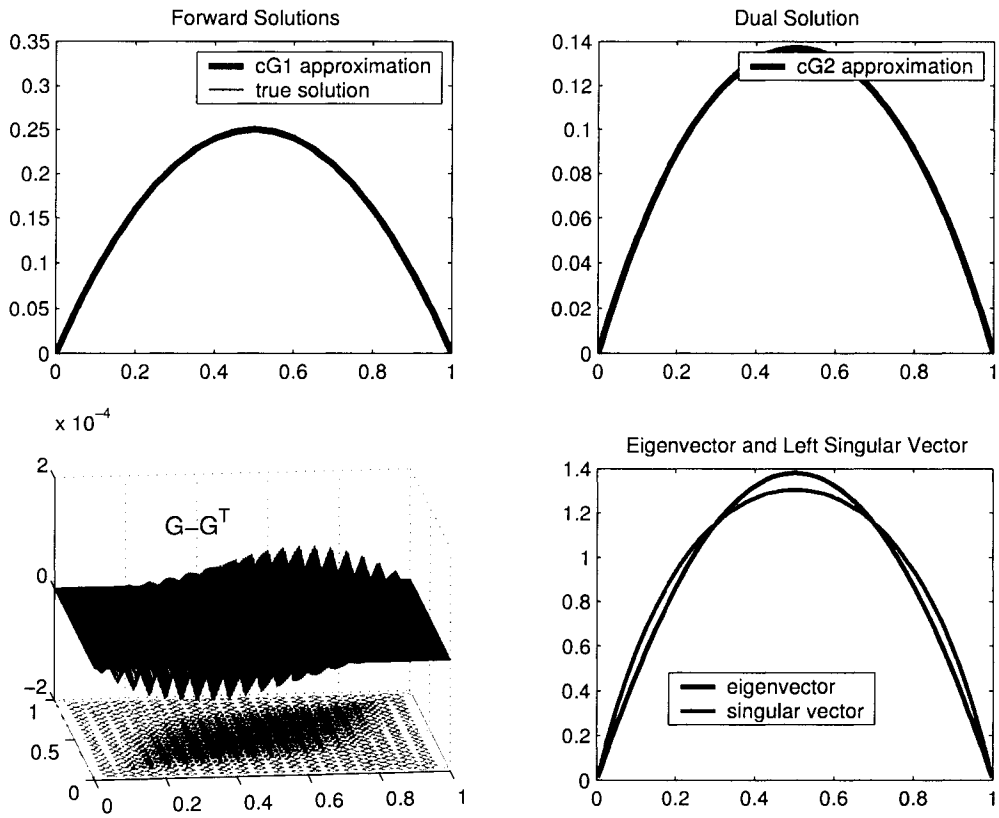

Figure 6.1: A self–adjoint dual problem doesn't mean $\Phi'_\psi(U)$ is normal.

for $V_h^{(2)}$, the singular vectors have a "jagged" shape, which is surely related to the "banding" that we see in the matrix reconstruction of $\Phi'_\psi(U)$. The singular vectors appear smoother relative to $V_h^{(1)}$. In fact, if we use only the nodes common to both spaces, the (normalized) vectors are the same. The power method that we implement for $\Phi'_\psi(U)^\top \Phi'_\psi(U)$ based on the Green's function converges to this "smoother" singular vector.

The above discussion indicates that there is still work to be done in analyzing the map $\Phi'_\psi(U)$ and the role that it plays in *a-posteriori* error estimation. Our results involving $\Phi'_\psi(U)$ combine both *a-posteriori* results

(our computations only involve the approximate solution) and *a-priori* results (the bounds involve the norm). Deriving purely *a-posteriori* bounds remains to be done. Additionally, the map $\Phi'_\psi(U)$ consists of many parts, some of which deserve further exploration. Some possibilities:

1. What analytic information can be gleaned from the Green's function formula (4.5.28) for $\Phi'_\psi(U)$?

2. How can we better understand the nature of $\Phi'_\psi(U)$ relative to the structure of both the forward and dual problems?

3. What is the significance of the structure of the singular vectors relative to different bases?

4. Is it possible to analyze the role of the dual data ψ on the bounds?

5. Is it possible to determine a class of problems for which $\Phi'_\psi(U)$ is normal, and a systematic way to measure the departure from normality?

6. It is not necessary to use the same mesh to solve the forward and dual problems when constructing $\Phi'_\psi(U)$. Is it worthwhile to solve the dual problem on a very coarse mesh and estimate $\|\Phi'_\psi(U)\|$?

Finally, the ideas in this thesis should be extended to time–dependent PDE. In that setting, matrix–free methods are essential, and the effects of linearization error are surely more dramatic.

Bibliography

[1] A. Abrosetti and G. Prodi. *A Primer of Nonlinear Analysis.* Cambridge University Press, Great Britain, 1993.

[2] Y. Achdou, C. Bernardi, and F. Coquel. A Priori and A Posteriori Analysis of Finite Volume Discretizations of Darcy's Equations. *Numerische Mathematik*, 96:17–42, 2003.

[3] A.C. Aitken. Studies in Practical Mathematics, II. The Evaluation of the Latent Roots and Latent Vectors of a Matrix. *Proceedings of the Royal Society of Edinburgh*, 57:269–304, 1937.

[4] Nicholas D. Alikakos and Giorgio Fusco. A dynamical systems proof of the Kreĭn-Rutman theorem and an extension of the Perron theorem. *Proceedings of the Royal Society of Edinburgh. Section A. Mathematics*, 117(3-4):209–214, 1991.

[5] U.M. Ascher, R.M.M. Mathcij, and R.D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[6] Kendall E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley and Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1989.

[7] I. Babuška and W.C. Rheinboldt. A-Posteriori Error Estimates for the Finite Element Method. *International Journal for Numerical Mehods in Engineering*, 12:1597–1615, 1978.

[8] I. Babuška and W.C. Rheinboldt. Error Estimates for Adaptive Finite Element Computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[9] I. Babuška and W.C. Rheinboldt. Analysis of Optimal Finite-Element Meshes in $\mathbb{R}^1$. *Mathematics of Computation*, 33(146):435–463, 1979.

[10] I. Babuška and W.C. Rheinboldt. A-Posteriori Error Analysis of Finite Element Solutions for One-Dimensional Problems. *SIAM Journal on Numerical Analysis*, 18(3):565–589, 1981.

[11] R.E. Bank and A. Weiser. Some A Posteriori Error Estimators for Elliptic Partial Differential Equations. *Mathematics of Computation*, 44(170):283–301, 1985.

[12] Eberhard Bänsch and Kunnibert G. Siebert. A Posteriori Error Estimation for Nonlinear Problems by Duality Techniques. Institut für Angewandte Mathematik, Freiburg, December 1995.

[13] J. Bebernes and D. Eberly. *Mathematical Problems from Combustion Theory*. Springer-Verlag, New York, 1989.

[14] Christine Bernardi. An Improved inf-sup Condition for the Spectral Discretization of the Stokes Problem in a Cylinder. *Computer Methods in Applied Mechanics and Engineering*, 175:267–280, 1999.

[15] Christine Bernardi and Frédéric Hecht. Error Indicators for the Mortar Finite Element Discretization of the Laplace Equation. *Mathematics of Computation*, 71(240):1371–1403, 2001.

[16] E. Bodewig. Bericht über die Methoden zur numerischen Lösung von algebraischen Eigenwertproblemen. I. *Atti del Seminario Matematico e Fisico dell'Università di Modena*, 4:133–193, 1950.

[17] E. Bodewig. Bericht über die Methoden zur numerischen Lösung von algebraischen Eigenwertproblemen. II. *Atti del Seminario Matematico e Fisico dell'Università di Modena*, 5:3–39, 1951.

[18] E. Bodewig. A Practical Refutation of the Iteration Method for the Algebraic Eigenvalue Problem. *Mathematical Tables and Other Aids to Computation*, 8(48):237–240, 1954.

[19] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, Berlin, Heidelberg, New York, 2002.

[20] Lia Bronsard and Robert V. Kohn. On the Slowness of Phase Boundary Motion in One Space Dimension. *Communications on Pure and Applied Mathematics*, 43(8):983–997, 1990.

[21] Ron Buckmire. On Exact and Numerical Solutions of the One-Dimensional Planar Bratu Problem. http://faculty.oxy.edu/ron/research. Preprint, 2003.

[22] Richard L. Burden and J. Douglas Faires. *Numerical Analysis.* PWS-Kent Publishing Company, Boston, 1993.

[23] Yang Cao and Linda Petzold. A Posteriori Error Estimation and Global Error Control for Ordinary Differential Equations by the Adjoint Method. *SIAM Journal of Scientific Computing*, 26(2):359–374, 2004.

[24] J. Carr and R Pego. Metastable Patterns in $u_t = \epsilon^2 u_{xx} - f(u)$. *Comm. Pure Appl. Math*, 42:523–576, 1989.

[25] J. Carr and R. L. Pego. Very Slow Phase Separation in One Dimension. In *PDEs and continuum models of phase transitions (Nice, 1988)*, volume 344 of *Lecture Notes in Phys.*, pages 216–226. Springer, Berlin, 1989.

[26] Carsten Carstensen, Sören Bartels, and Stephan Jansche. A Posteriori Error Estimates for Nonconforming Finite Element Methods. *Numerische Mathematik*, 92:233–256, 2002.

[27] Zhiming Chen and Ricardo H. Nochetto. Residual Type A Posteriori Error Estimates for Elliptic Obstacle Problems. *Numerische Mathematik*, 84:527–548, 2000.

[28] Fehmi Cirak and Ekkehard Ramm. A-Posteriori Error Estimation and Adaptivity for Elastoplasticity Using the Reciprocal Theorem. *International Journal for Numerical Methods in Engineering*, 47(1-3):379–393, 2000. Richard H. Gallagher Memorial Issue.

[29] E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations.* McGraw-Hill, New York, Toronto, London, 1955.

[30] John B. Conway. *A Course in Functional Analysis, 2nd ed.* Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong, 1990.

[31] E. Deeba, S.A. Khuri, and S. Xie. An Algorithm for Solving Boundary Value Problems. *Journal of Computational Physics*, 159:125–138, 2000.

[32] Luca Dieci. Jacobian Free Computation of Lyapunov Exponents. *Journal of Dynamics and Differential Equations*, 13(3):697–717, 2002.

[33] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics.* Society for Industrial and Applied

Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.

[34] Jean Descloux and Jaques Rappaz. A Nonlinear Inverse Power Method With Shift. *SIAM Journal on Numerical Analysis*, 20(6):1147–1152, 1983.

[35] Paul Dirac. *Principles of Quantum Mechanics.* Clarendon Press, Oxford, 1947.

[36] Paul DuChateau and David Zachmann. *Applied Partial Differential Equations.* Dover Publications, New York, 1989.

[37] Todd Dupont. Mesh Modification for Evolution Equations. *Mathematics of Computation*, 39(159):85–107, 1982.

[38] P. J. Eberlein. On Measures of Non-Normality for Matrices. *American Mathematical Monthly*, 72:995–996, 1965.

[39] C.M. Elliott and A.M. Stuart. The Global Dynamics of Discrete Semilinear Parabolic Equations. *SIAM Journal on Numerical Analysis*, 30(6):1622–1663, 1993.

[40] L. Elsner. On the Variation of the Spectra of Matrices. *Linear Algebra and its Applications*, 47:127–138, 1982.

[41] L. Elsner and M. H. C. Paardekooper. On Measures of Nonnormality of Matrices. *Linear Algebra and its Applications*, 92:107–123, 1987.

[42] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Computational Differential Equations.* Cambridge University Press, Great Britain, 1996.

[43] Kenneth Eriksson. Finite Element Methods of Optimal Order for Problems with Singular Data. *Mathematics of Computation*, 44(170):345–360, 1985.

[44] Kenneth Eriksson. Improved Accuracy by Adapted Mesh-Refinements in the Finite Element Method. *Mathematics of Computation*, 44(170):321–343, 1985.

[45] Kenneth Eriksson and Claes Johnson. Adaptive Finite Element Methods for Parabolic Problems i: A Linear Model Problem. *SIAM Journal on Numerical Analysis*, 28(1):43–77, 1991.

[46] Donald Estep. An Analysis of Numerical Approximations of Metastable Solutions of the Bistable Equation. *Nonlinearity*, 7:1445–1462, 1994.

[47] Donald Estep. A Posteriori Error Bounds and Global Error Control for Approximation of Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 32(1):1–48, 1995.

[48] Donald Estep. The Pointwise Computability of the Lorenz System. *Mathematical Models and Methods in Applied Sciences*, 8(8):1277–1305, 1998.

[49] Donald Estep, Michael Holst, and Mats Larson. Generalized Green's Functions and the Effective Domain of Influence. *SIAM Journal of Scientific Computing*, To appear.

[50] Donald Estep, Michael Holst, and Duane Mikulencak. Accounting for Stability: A Posteriori Error Estimates Based on Residuals and Variational Analysis. *Communications in Numerical Methods in Engineering*, 18(1):15–30, 2002.

[51] Donald J. Estep, Dewey H. Hodges, and Michael Warner. The Solution of a Launch Vehicle Trajectory Problem by an Adaptive Finite-Element Method. *Computer Methods in Applied Mechanics and Engineering*, 190(35-36):4677–4690, 2001.

[52] Donald J. Estep, Mats G. Larson, and Roy D. Williams. Estimating the Error of Numerical Solutions of Systems of Reaction-Diffusion Equations. *Memoirs of the American Mathematical Society*, 146(696):viii–109, 2000.

[53] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, Rhode Island, 1998.

[54] Gerald Folland. *Fourier Analysis and its Applications*. Wadsworth and Brooks/Cole, Pacific Grove, CA, 1992.

[55] Kurt Georg. On the Convergence of an Inverse Iteration Method for Nonlinear Elliptic Eigenvalue Problems. *Numerische Mathematik*, 32:69–74, 1979.

[56] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 1983.

[57] Philip Hartman. A Lemma in the Theory of Structural Stability of Differential Equations. *Proceedings of the American Mathematical Society*, 11(4):610–620, 1960.

[58] Philip Hartman. *Ordinary Differential Equations.* John Wiley & Sons, Baltimore, 1973.

[59] Peter Henrici. Bounds for Iterates, Inverses, Spectral Variation and Fields of Values of Non-Normal Matrices. *Numerische Mathematik*, 4:24–40, 1962.

[60] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 1996.

[61] Nicholas J. Higham and Philip A. Knight. Matrix Powers in Finite Precision Arithmetic. *SIAM J. Matrix Anal. Appl.*, 16(2):343–358, 1995.

[62] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, New York and London, 1974.

[63] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Cambridge University Press, Cambridge, 1985.

[64] Howard Hotelling. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24:447–441 and 498–520, 1933.

[65] Alston S. Householder. *The Theory of Matrices in Numerical Analysis.* Blaisdell Publishing Company, New York, Toronto, London, 1964.

[66] T.E. Hull, W.H. Enright, B.M. Fellen, and A.E. Sedgwick. Comparing Numerical Methods for Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[67] Claes Johnson. Error Estimates and Adaptive Time-Step Control for a Class of One-Step Methods for Stiff Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 25(4):908–926, 1988.

[68] Philip A. Knight, Michael Grinfeld, and Harbir Lamba. Nonnormality and Finite Precision Arithmetic in Power Method Dynamics. Technical Report 1997/6, University of Strathclyde, Glasgow, Scotland, February 1997.

[69] M. G. Krein and M. A. Rutman. Linear Operators Leaving Invariant a Cone in a Banach Space. *American Mathematical Society Translations*, 1950(26):128, 1950.

[70] Erwin Kreyszig. *Introductory Functional Analysis with Applications.* Wiley Classics Library, New York, Chichester, Brisbane, Toronto, Singapore, 1978.

[71] Cornelius Lanczos. *Linear Differential Operators.* Dover, Mineola, New York, 1997.

[72] Steven L. Lee. A Practical Upper Bound for Departure from Normality. *SIAM Journal on Matrix Analysis and Applications*, 16(2):462–468, 1995.

[73] John Locker. *Functional Analysis and Two-Point Differential Operators.* Pitman Research Notes In Mathematics Series. Longman Scientific & Technical, England, 1986.

[74] J. David Logan. *An Introduction to Nonlinear Partial Differential Equations.* John Wiley & Sons, New York, Chicester, Brisbane, Toronto, Singapore, 1994.

[75] Charalambos Makridakis and Ricardo H. Nochetto. Elliptic Reconstruction and A Posteriori Error Estimates for Parabolic Problems. *SIAM Journal on Matrix Analysis and Applications*, 41:1585–1594, 2003.

[76] Robert McOwen. *Partial Differential Equations, Methods and Applications.* Prentice-Hall, Upper Saddle River, NJ, 1996.

[77] Kenneth S. Miller. *Linear Differential Equations in the Real Domain.* W.W. Norton & Company, New York, 1963.

[78] R.H. Nochetto, A. Schmidt, and C. Verdi. A Posteriori Error Estimation and Adaptivity for Degenerate Parabolic Problems. *Mathematics of Computation*, 69(229):1–24, 1999.

[79] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Academic Press, New York, 1970.

[80] A. M. Ostrowski. *Solution of equations and systems of equations.* Second edition. Pure and Applied Mathematics, Vol. 9. Academic Press, New York, 1966.

[81] Jeffrey S. Ovall. *Duality-Based Adaptive Refinement for Elliptic PDEs.* PhD thesis, University of California, San Diego, 2004.

[82] B.N. Parlett and W.G. Jr. Poole. A Geometric Theory for the QR, LU, and Power Iterations. *SIAM Journal on Numerical Analysis*, 10(2):389–412, 1973.

[83] Lawrence Perko. *Differential Equations and Dynamical Systems.* Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[84] Oskar Perron. Zur Theorie der Matrices. *Mathematische Annalen*, 64:248–263, 1907.

[85] Werner C. Rheinboldt. On Measures of Ill-Conditioning for Nonlinear Equations. *Mathematics of Computation*, 30:104–111, 1976.

[86] Werner C. Rheinboldt. On a Theory of Mesh-Refinement Processes. *SIAM Journal on Numerical Analysis*, 17(6):766–778, 1980.

[87] G.F. Roach. *Green's Functions.* Cambridge University Press, Great Britain, 1982.

[88] Axel Ruhe. On the Closeness of Eigenvalues and Singular Values for Almost Normal Matrices. *Linear Algebra and its Applications*, 11:87–94, 1975.

[89] Martin Schechter. *Principles of Functional Analysis.* American Mathematical Society, Providence, Rhode Island, 2002.

[90] Laurent Schwartz. *Théorie des Distributions (2 vols.).* Hermann, Paris, 1950–1951.

[91] L.F. Shampine, H.A. Watts, and S.M. Davenport. Solving Nonstiff Ordinary Differential Equations–The State of the Art. *SIAM Review*, 18(3):376–411, 1976.

[92] Joel Smoller. *Shock Waves and Reaction–Diffusion Equations.* Springer–Verlag, New York, Heidelberg, Berlin, 1983.

[93] Ivar Stakgold. *Green's Functions and Boundary Value Problems.* Wiley Interscience, New York, Chichester, Brisbane, Toronto, 1979.

[94] G. W. Stewart. Error Bounds for Approximate Invariant Subspaces of Closed Linear Operators. *SIAM Journal on Numerical Analysis*, 8:796–808, 1971.

[95] G. W. Stewart. Error and Perturbation Bounds for Subspaces Associated with Certain Eigenvalue Problems. *SIAM Review*, 15:727–764, 1973.

[96] G. W. Stewart. *Matrix algorithms. Vol. I.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Basic Decompositions.

[97] G. W. Stewart. *Matrix algorithms. Vol. II.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Eigensystems.

[98] G. W. Stewart. On the Powers of a Matrix with Perturbations. *Numerische Mathematik*, 96(2):363–376, 2003.

[99] Gilbert Strang and George J. Fix. *An Analysis of the Finite Element Method.* Prentice Hall Series in Automatic Computation, Englewood Cliffs, N.J., 1973.

[100] Robert Strichartz. *A Guide to Distribution Theory and Fourier Transforms.* CRC Press, Boca Raton, 1994.

[101] Steven H. Strogatz. *Nonlinear Dynamics and Chaos.* Addison Wesley Publishing Company, Reading, Massachusetts, 1994.

[102] Silviu Teleman. *Applications of the Theory of Distributions.* John Wiley and Sons, London, New York, Sydney, Toronto, 1973.

[103] Roger Thelwell. *Adjoint Approach to Parameter Identification With Application to the Richards Equation.* PhD thesis, Colorado State University, 2004.

[104] L. N. Trefethen. Pseudospectra of Matrices. In *Numerical analysis 1991 (Dundee, 1991)*, volume 260 of *Pitman Res. Notes Math. Ser.*, pages 234–266. Longman Sci. Tech., Harlow, 1992.

[105] Bart Van der Rotten. *Limited Memory Broyden Methods.* PhD thesis, Leiden, 2003.

[106] Rüdiger Verfürth. A Posteriori Error Estimates for Nonlinear Problems. Finite Element Discretizations of Elliptic Equations. *Mathematics of Computation*, 62(206):445–475, 1994.

[107] Rüdiger Verfürth. *A Review of A-Posteriori Error Estimation and Adaptive Mesh Refinement Techniques.* Wiley Teubner, Chichester, New York, Brisbane, Toronto, Singapore, Stuttgart, Leipzig, 1996.

[108] R. von Mises. Praktische Methoden zur Gleichungsauflösung. *Zeitschrift fur Angewandte Mathematik und Mechanik*, 9:62–77, 1927.

[109] David S. Watkins. *Fundamentals of Matrix Computations.* John Wiley & Sons Inc., New York, 1991.

[110] J.H. Wilkinson. The Use of Iterative Methods for Finding the Latent Roots and Vectors of Matrices. *Mathematical Tables and Other Aids to Computation*, 9(52):184–191, 1955.

[111] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[112] J.H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation, vol II, Linear Algebra.* Springer-Verlag, New York, 1971.

[113] Kôsaku Yosida. *Functional Analysis.* Springer-Verlag, Berlin, Heidelberg, New York, 1980.

[114] Eberhard Zeidler. *Nonlinear Functional Analysis and its Applications I, Fixed–Point Theorems.* Springer–Verlag, New York, Berlin, Heidelberg, Tokyo, 1986.

[115] J.Z. Zhu and O.C. Zienkiewicz. Adaptive Techniques in the Finite Element Method. *Communications in Applied Numerical Methods*, 4:197–204, 1988.

Appendix A

CODE FOR STABILITY CALCULATIONS

We present here the computer programs used to examine the ideas in Chapter 3. Note that the limited memory Broyden methods alluded to in Chapter 3 are not implemented in these codes.

A.1 MATLAB codes

```
% FUNCTION power_broy
% COMPUTES DOMINANT EIGENVALUE FOR LINEARIZED SYSTEM OF ODE
% USING BROYDEN'S METHOD AND THE NONLINEAR POWER METHOD
% INPUTS:
%    guess: user provides an initial guess for
%           the equilibrium solution
% OUTPUTS:
%    init_and_final: initial guess and equilibrium solution
%    eig_val: result of nonlinear power method
%    sP_sF_usF: a vector with three columns-
%               the results of Matlab's calculation of the
%               eigenvalues of the matrices corresponding
%               to the shifted problem and its inverse
%    vec_results: first column is components of approximate
```

```
%                 eigenvector from power method, second column
%                 is eigenvector from Matlab
%    eig_val: shows the approximate eigenvalue at each iteration
function [init_and_final,eig_val,sP_sF_usF,vec_results,lambda]=...
                                                power_broy(guess)
global n
n=length(guess);
v1=ones(1,n-1);
v2=ones(1,n);
global lam
lam=0.1;
global B;
% FINITE DIFFERENCE MATRIX
B=diag(v1,-1)+diag(-2*v2,0)+diag(v1,1);
  B(1,1)=-1; % neumann bdy. conds.
  B(n,n)=-1;
epsilon=sqrt(eps);
TOL=epsilon;
maxit=100;
d=zeros(length(guess),1);
phi_0=broyden(guess,d,epsilon,maxit,TOL); % u for Phi(0)=f(u)=0
delta=rand(length(guess),1);
delta(:,1)=delta/norm(delta);
vect(:,1)=delta;
lambda(1)=1;
rate=2;
```

```
global shift
shift=0.1;
shiftedphi=phi_0;
ITER=floor(-log(eps)/(2*log(2)));
for i=2:ITER
    delta(:,i)=broyden2(guess,-shift*phi_0+delta(:,i-1),...
                                    epsilon,maxit,TOL)-shiftedphi;
    lambda(i)=(delta(:,i)'*delta(:,i-1))/norm(delta(:,i-1))^2;
    vect(:,i)=delta(:,i)/norm(delta(:,i));
    delta(:,i)=vect(:,i)/rate^(i-1);
end
lambda=lambda';
vector=vect(:,end);
eig_val=lambda(end);
% COMPUTE NUMERICAL JACOBIAN FOR COMPARISON
H=epsilon*eye(n);
for i=1:n
    cent_num_jac(:,i)=...
       (f((phi_0+H(:,i)),d)-f((phi_0-H(:,i)),d))/(2*epsilon);
end
init_and_final=[guess phi_0];
% COMPUTE INVERSE FOR COMPARISON
nj_inv=inv(cent_num_jac-shift*eye(n));
[v,e]=eig(nj_inv);
e=diag(e);
eigs_shifted_Phi_zero=e;
```

```
eigenvalues_shifted_F_prime=1./e;
eigenvalues_F_prime=(1./e)+shift;
sP_sF_usF=[eigs_shifted_Phi_zero...
           eigenvalues_shifted_F_prime eigenvalues_F_prime];
[p,ip]=max(abs(e));
domin_vec=v(:,ip);
vec_results=[vector domin_vec];
```

```
% FUNCTION broyden2
% Broyden's method for solving f(x)-d=0
% INPUTS:
%   x: initial guess for broyden iteration
%   d: the shift to f(x): f(w)-d=0 so f(w)=d
%   epsilon: size of perturbation in initial approx. Jacobian
%   maxit: maximum number of broyden iterations
%   TOL: stopping criteria
% OUTPUT:
%   out: result of broyden iteration
function out=broyden2(x,d,epsilon,maxit,TOL)
global n % power_broy.m makes the length of the guess global
H=epsilon*eye(n);
y=f2(x,d);
A=-eye(n);
% BROYDEN ALGORITHM
out(:,1)=x;
for i=2:maxit
    s=A\-f2(x,d);
    x_new=x+s;
    y=f2(x_new,d)-f2(x,d);
    A=A+(y-A*s)*s'/(s'*s);
    out(:,i)=x_new;
    check = norm(s);
    if check <= TOL
        break
```

```
    end
    x=x_new;
end
if i==maxit
    fprintf('broyden didnt converge ...
    (did %.4g, the max number of iterations) \n', maxit);
else
    fprintf('did %.4g broyden iterations\n',i);
end
out=out(:,end);
```

```
% FUNCTION f
%   for nonlinear power method using power_broy.m and broyden2.m
%   the m-file "power_broy.m" defines the global variables lam, B, n
%   lam is the parameter in the equation u'=Bu+lam*f
%   B is the finite difference approximation to the laplacian
%   n is the dimension of the problem
% INPUTS u and d
% OUTPUT function evaluation
function y=f(u,d)
global lam; global B; global n;
for i=1:n
    nonlin(i,1)=u(i)-u(i)^3;
end
nonlin=lam*nonlin;
y=(B*u+nonlin)-d;
% FUNCTION f2
%    f2 incorporates the shift
function y=f2(u,d)
global lam; global B; global n; global shift;
for i=1:n
    nonlin(i,1)=u(i)-u(i)^3;
end
nonlin=lam*nonlin;
y=(B*u+nonlin)-d-shift*u;
```

Appendix B

CODE FOR *A-POSTERIORI* CALCULATIONS

This appendix includes codes used for is for analyzing the error of linearization for computational error estimates. We include here the codes used to produce the examples of the 2×2 systems of nonlinear algebraic equations in Chapters 4 and 5, followed by the codes used to examine the two–point boundary value problem (4.3.16) and corresponding linearized dual problem (4.3.29).

B.1 Code for 2×2 problems

B.1.1 Main program

```
% This is the driver program for analyzing f(x)=0
% this code constructs a grid of points around an
% exact solution to f(x)=0: you need to define the exact sol'n
% and specify the grid.
% REQUIRED FILES:
%    comp1.m comp2.m comp3.m comp4.m - COMPONENTS OF JACOBIAN
%    preliminaries.m f.m fprime.m Phi.m Phi_deriv.m
%    ppower3.m
%    eigen_calculatioin.m perturbations.m
```

```
clear;
% preliminary data set-up and function calls
preliminaries;
[phihat,Ahat,Atrue,lam,dom_vec] = ...
      eigen_calculation(x,y,M,N,exact,psi);
[Aperts,approx_dual,perturbed_dual,change_in_phi,...
     pert_val,largest_change,d_o_c]=...
          perturbations(perts,Atrue,Ahat,cent,psi);
appr_dom_eig=lam(y_val,x_val);
dominant_eigenvector=...
     [dom_vec(y_val,x_val,1) dom_vec(y_val,x_val,2)];
eigendirection=q*dominant_eigenvector/norm(dominant_eigenvector);
%calculate dual solutions corresp to F'(X+ew) and true A
max_dual=Phi([cent(1)+eigendirection(1)...
         cent(2)+eigendirection(2)],psi)';
true_dual=(inv(Atrue(:,:,x_val,y_val))'*psi)';
 % IP is inner product of residual with the approx. dual
 %      solution corresponding to linearization around f'(X)
 % eRlam is epsilon*||R||*|lambda|
 % maxIP is the projection at the maximally perturbed dual
IP=abs(R(:,:,1).*phihat(:,:,1)+R(:,:,2).*phihat(:,:,2));
maxIP=abs(R(y_val,x_val,1)*max_dual(1)+...
      R(y_val,x_val,2)*max_dual(2));
eRlam=epsilon.*resnorm.*lam;
RHS=eRlam+IP;
pd=Phi_deriv(cent,psi);
```

```
[phi_vect,phi_val]=eig(pd);
true_dom_eig=max(max(abs(phi_val)));
diff=abs(true_dom_eig-abs(appr_dom_eig));
[u,s,v]=svd(pd); %singular value decomp
rtsingmax=q*v(:,1); %dominant right singular vector
lftsingmax=q*u(:,1);  %dominant left singular vector
rtsingmin=q*v(:,2); %dominant right singular vector
lftsingmin=q*u(:,2);  %dominant left singular vector
temp=[eigendirection' rtsingmax lftsingmax rtsingmin lftsingmin];
for i=1:length(temp)
 [th1(i),r1(i)]=cart2pol(temp(1,i),temp(2,i));
  if th1(i) <= 0
     th1(i) = 2*pi+th1(i);
  end
 th1(i)=th1(i)/(2*pi);
 th1(i)=round(th1(i)*(length(perts)));
end
[th,r]=cart2pol(eigendirection(1),eigendirection(2));
if th <= 0
    th = 2*pi+th;
end
th=th/(2*pi);
th=round(th*(length(perts)));
bound=true_dom_eig*q*ones(length(pert_val),1);
truediff=norm(true_dual-approx_dual);
TD=truediff*ones(length(pert_val),1);
```

```
% PLOT COMMANDS
figure
 plot([cent(1),exact(1)],[cent(2),exact(2)],'k');
 hold on;
 plot([cent(1),perts(d_o_c,1)],...
      [cent(2),perts(d_o_c,2)],'k-*','LineWidth',2);
 hold on;
  % plot dominant eigenvector
 plot([cent(1),cent(1)+eigendirection(1)],...
 [cent(2),cent(2)+eigendirection(2)],'r--+','LineWidth',2);
 hold on;
 %plot "dominant right singular vector"
 plot([cent(1),cent(1)+rtsingmax(1)],...
 [cent(2),cent(2)+rtsingmax(2)],'g--o','LineWidth',2);
 hold on;
 %plot "dominant left singular vector"
 plot([cent(1),cent(1)+lftsingmax(1)],...
 [cent(2),cent(2)+lftsingmax(2)],'m--+','LineWidth',2);
 hold on;
 % plot perturbations of approx. soln
 plot(perts(:,1),perts(:,2));
 legend('direction to exact soln',...
 'direction of largest change','dominant eigendirection')
 axis square;
figure
 plot(d_o_c,pert_val(d_o_c),'k*',th1(1),pert_val(th1(1)),'rs');
```

```
 hold on;
 title('how dual solution changes with perturbations')
 legend('maximum perturbation','pert in dir. of dom. eigenvector');
 hold on;
 plot(pert_val);
  plot(bound);
 hold on;
 plot(TD);
figure
  surf(xx,yy,errproj)
  title('projection of error in direction of...
                          dual data-exact and approx.');
  hold on
  surf(xx,yy,IP)
  hold on
  surf(xx,yy,RHS)
figure
 subplot(1,2,1)
 surf(xx,yy,resnorm)
 subplot(1,2,2)
 surf(xx,yy,epsilon)
 pv=pert_val(d_o_c);
 th=th1(1);
 pvth=pert_val(th1(1));
 % PRINT statements
 fprintf('the exact solution is:')
```

```
disp(exact')
fprintf('the approx. solution you chose was:')
disp(cent')
fprintf('the dual data is a unit vector corresponding to:')
disp(psi')
fprintf('the derivative of the dual...
          solution operator Phi_psi is\n')
disp(pd)
fprintf('the perturbed power method calculates the...
      dominant eigenvalue to be %.9g\n',appr_dom_eig )
fprintf('and it calculates the dominant eigenvector to be\n')
disp(dominant_eigenvector')
fprintf('matlabs eig routine calculates the ...
      dominant eigenvalue to be %.9g\n',true_dom_eig)
fprintf('and matlab calculates the eigenvectors to be\n')
disp(phi_vect)
fprintf('the difference between pert. power...
      and matlab routine is %.5g\n\n',diff)
fprintf('the dual operator f-prime(X)^t is:\n')
disp(Ahat(:,:,x_val,y_val));
fprintf('the true linearized dual operator given...
                              by integral MVT is:\n')
disp(Atrue(:,:,x_val,y_val));
fprintf('the dual solution corresponding to...
      linearization around X is:\n\n')
disp(approx_dual);
```

```
fprintf('the dual solution in the direction...
      of maximum perturbation of size...
       %.5g is:\n\n',epsilon(y_val,x_val))
disp(max_dual);
fprintf('the dual solution corresponding to...
                    the MVT operator is:\n\n')
disp(true_dual);
fprintf('the true projection of the...
      error |(e,psi)| is %.9g\n\n',errproj(y_val,x_val))
fprintf('the projection |(R,phihat)|...
      corresp. to linearization around f'(X) is...
                       %.9g\n\n',IP(y_val,x_val))
fprintf('the projection |(R,phihatmax)|...
      corresp. to linearization at f'(X+ew) is %.9g\n\n',maxIP)
fprintf('the quantity e*||R||*|lambda| is ...
                  %.9g\n\n',eRlam(y_val,x_val))
fprintf('norm of difference between true and...
                       approximate duals is:\n\n')
disp(norm(true_dual-approx_dual))
fprintf('norm of difference between...
      maximally perturbed and approx duals is\n\n')
disp(norm(max_dual-approx_dual))
table=[maxIP IP(y_val,x_val) eRlam(y_val,x_val)...
      IP(y_val,x_val)+eRlam(y_val,x_val)]
```

B.1.2 Secondary programs and functions

```
% PRELIMINARIES - define and certain quantities involved
% in solving f(x)=0
% the function f is defined in another file called f.m
% user defined quantities:
%   exact -- the exact solution to f(x)=0
%   psi -- vector for dual data
%   x,y -- coordinates of approximate solutions
%          near exact soln
%   x_val, y_val -- user chooses a particular
%                   approximate solution
%   scale, cent -- scale factor for size of
%                   perturbations about  cent-
%                   which is the approximate solution corresp.
%                   to x_val and y_val
% quantities computed:
%   perts and perts 2 -- perturbations of chosen
%                        approximate solution
%   error -- difference between exact solution
%            and each approximate
%            solution in the grid- each component of
%            error is a page in a multi-dimensional array
%   R -- Residuals - has same form as error
% define exact solution to the test problem f(x)=0
exact=[1 ; 1];
psi=input('enter dual data ');
```

```
psi=psi/norm(psi);
% define grid around exact solution
% to be used for "approximate" solutions to f(x)=0
x=[.9:.05/4:1.1];
y=x;
M = length(x);
N = length(y);
[xx,yy]=meshgrid(x,y); % makes grid for plotting
                       % can only have functions of 2 variables
 x_val=input('enter x_val '); % x_val and y_val need to be
                              % integers between 1 and N or M
 y_val=input('enter y_val '); % these determine which gridpoint
                              % is used as the center
% this computes the error vector- each page of
% the multi-dimensional array contains the
% components of the true error
error(:,:,1)=exact(1)-xx;
error(:,:,2)=exact(2)-yy;
epsilon=sqrt(error(:,:,1).^2+error(:,:,2).^2);
% errproj is |(e,psi)| -> the exact projection of the error
% on the dual data psi
errpsi(:,:,1)=error(:,:,1)*psi(1);
errpsi(:,:,2)=error(:,:,2)*psi(2);
errproj=abs(errpsi(:,:,1)+errpsi(:,:,2));
% this calculates residuals
% R is a mult-dim array holding the components of the residuals-
```

```
% resnorm is a matrix whose (j,i) entry
%is the norm of the residual at [x(i),y(j)]
for i=1:M
    for j=1:N
        scratchy=f(x(i),y(j));
        R(j,i,1)=scratchy(1);
        R(j,i,2)=scratchy(2);
        resnorm(j,i)=norm([scratchy(1) scratchy(2)]);
    end
end
% approx. soln about which perts are formed:
cent=[x(x_val);y(y_val)];
q=norm(exact-cent); % q is the distance from exact to approx soln
t=[0:pi/32:2*pi]';
% perturbations of size q:
perts=[cent(1)+q*cos(t) cent(2)+q*sin(t)];
```

```
% FUNCTION eigen_calculation
% this function calculates important quantities based on
% information passed to it which comes from the file
% "preliminaries" - which sets up some initial data
% this double loop computes the following quantities:
%   phihat - the approximate dual solution, corresponding to
%            linearization around f'(X)
%   lambda - a cell array containing eigenvalues
%            and eigenvectors as computed by "ppower3"-
%            the perturbed power method
%   Ahat - dual operator corresponding to
%          linearization around f'(X)
%   Atrue - dual operator as given by  mean value theorem
%            computes the integral of f'(X+s(x-X)) from 0 to 1
%            using the matlab routine "quadl" for quadrature
function [phihat,Ahat,Atrue,lam,dom_vec] =...
                eigen_calculation(x,y,M,N,exact,psi)
for i = 1:N     % y-loop
    for j = 1:M % x-loop
     scratch=Phi([x(j);y(i)],psi);
     phihat(i,j,1)=scratch(1); % 1st comp. of approx. dual sol'n
     phihat(i,j,2)=scratch(2); % 2nd comp. of approx. dual sol'n
     lambda(i,j,:)=ppower3([x(j);y(i)],psi); % apprx. e-vals
                                             % for Phi-prime(X)
     % lambda is a CELL ARRAY- lambda(:,:,1) is the dom. eig. value
     % and lambda(:,:,2) contains the dominant eigenvector
```

```
      % Ahat, the "perturbed" dual operator-this is f'(X)^T
      Ahat(:,:,j,i)=fprime([x(j) y(i)])';
      % The exact A from int. MVT -using quadrature
      Atrue(:,:,j,i)=...
        [quadl('comp1',0,1,[],[],[x(j);y(i)],exact)...
         quadl('comp2',0,1,[],[],[x(j);y(i)],exact);
         quadl('comp3',0,1,[],[],[x(j);y(i)],exact)...
         quadl('comp4',0,1,[],[],[x(j);y(i)],exact)];
      end
   end
% convert first page of lambda cell array to a regular matrix
L1=lambda(:,:,1);
L2=lambda(:,:,2);
for i=1:N
    for j=1:M
        lam(i,j)=L1{i,j}; %lam is the matrix of eigenvalues
        dom_vec(i,j,:)=L2{i,j};
    end
end
```

```
% FUNCTION perturbations.m - create a sequence of perturbations
% to a chosen approximate solution
% inputs perts, cent and psi all come from the file
%  'preliminaries.m'
% inputs Atrue and Ahat are generated by the function
%  'eigen_calculation'
% this function computes the solution of the perturbed
% dual problem f'(X+d)phi = psi where X is an
% approximate solution, d is a
% perturbation and psi is the dual data
% the function then computes the dual solution at f'(X)
% and finds the perturbation to X that causes
% the largest change in the dual solution
function [Aperts,approx_dual,perturbed_dual,...
              change_in_phi,pert_val,largest_change,d_o_c]=...
                   perturbations(perts,Atrue,Ahat,cent,psi)
% this loop computes f'(X+d) where X=cent
% and d are the perturbations
for j=1:length(perts)
    Aperts(:,:,j)=fprime([perts(j,1);perts(j,2)])';
end
Aperts(:,:,j+1)=Atrue(:,:,1);
Aperts(:,:,j+2)=Ahat(:,:,1);
 % solution to the dual problem corresponding to linearization
 % around the approximate solution- phi = f'(X)^-T*psi
 % (it's transposed to make a row vector- to be compatible with
```

```
 % perturbed duals below-
approx_dual=Phi([cent(1),cent(2)],psi);
approx_dual=approx_dual';
 % this loop creates the dual solutions at the perturbed values
 % of the approximate solution- the rows are the comps. of the dual
 % change_in_phi gives the difference between the dual sol'n at X
 % and the dual solution at X+perts
 % pert_val gives the size of the change in phi
for i=1:length(perts)
    r1=Phi([perts(i,1),perts(i,2)],psi);
    perturbed_dual(i,1)=r1(1);
    perturbed_dual(i,2)=r1(2);
    change_in_phi(i,:)=perturbed_dual(i,:)-approx_dual;
    pert_val(i,1)=norm(change_in_phi(i,:));
end
 % largest_change gives the size of the biggest
 % change in the approximate dual solution
 % and d_o_c (direction of change) marks which
 % perturbation causes this change
[largest_change,d_o_c]=max(pert_val);
```

```
function y=f(x1,x2)
y=[x1.^2/2-1/2  x2.^3/3-1/3];
% comp1 is the first component of the Jacobian for 2x2 problem
% THIS WILL ONLY WORK FOR 2x2 PROBLEMS!
% X and x are 2x2 vectors- the approx. and exact solutions
% for example if the (1,1) entry of the jacobian is (xy^2)
% then you have to define the function by
    % y= (X(1)+s*(x(1)-X(1)))*(X(2)+s*(x(2)-X(2)))^2;
    %        this is x             this is y^2
function y=comp1(s,X,x)
y=X(1)+s*(x(1)-X(1));
function y=comp2(s,X,x)
y=0*s;
function y=comp3(s,X,x)
y=0*s;
function y=comp4(s,X,x)
%y=(a+s*(b-a)).^2;
y=(X(2)+s*(x(2)-X(2))).^2;
% Phi maps the approximate solution to the forward problem
% to the approximate dual problem formed by linearizing around X
% x1 and x2 are components of vector, psi is the data vector
% the matrix defined here is the Jacobian of the function f(x)
function y=Phi(x,psi)
y=inv([x(1) 0 ; 0 x(2).^2]')*psi;
function y=Phi_deriv(x,psi)
y=[-psi(1)/x(1)^2 0 ; 0 -2*psi(2)/x(2)^3];
```

B.2 Two–point boundary value problems

B.2.1 Main program

```
n=19; % specify number of interior nodes
gaussrule=5; % specify 2, 3 or 5 point gaussrule for quadratures
global param gamma lambda
param =1;
gamma = .2; %for prexam4 also for burgers
lambda = 1.5; %for bratu
Ntol=1e-10;
Nmax=20;
[outer,inner,steps,mids,dualmesh,dm,gaussgrid,gaussweights]=...
                                          gridsetup(n,gaussrule);
% COMPUTE APPROXIMATE SOLUTION U
uold=newtonguess(inner); %initial guess for newton solve
U = newton(uold,inner,steps,Ntol,Nmax);
wholeU=[0;U;0];  % give U the values 0 at the endpoints
fullU=interp1(outer,wholeU,dualmesh);
% LINEARLY INTERPOLATE FORWARD SOLUTION TO GAUSSGRID
lingaussu=interp1(outer,wholeU,gaussgrid);
[lins,dlins]=linbasis(gaussgrid,outer,steps);
for j=1:length(wholeU)
    dnu(:,j)=wholeU(j)*dlins(:,j);
end
dlingaussu=sum(dnu')'; %gets U' on gaussgrid
linpert=outer.^3.*(1-outer);
```

```
linpert=linpert/l2norm(outer,outer(2:end-1));
pert=dualmesh.^3.*(1-dualmesh);
pert=pert/l2norm(dualmesh,pert(2:end-1));
[quads,dquads]=quadbasis(gaussgrid,outer,steps,mids,dualmesh);
% EXPRESS FORWARD SOLUTION IN BASIS OF PIECEWISE QUADRATICS
for i=1:dm
    qu(:,i)=fullU(i)*quads(:,i);
    dqu(:,i)=fullU(i)*dquads(:,i);
    pu(:,i)=pert(i)*quads(:,i);
    dpu(:,i)=pert(i)*dquads(:,i);
end
gaussu=sum(qu')';
dgaussu=sum(dqu')';
gausspert=sum(pu')';
dgausspert=sum(dpu')';
gausspert=gausspert/l2norm(gaussgrid,gausspert(2:end-1));
% EVALUATE COEFFICIENTS OF LINEARIZED DUAL PROBLEM
afn=a(gaussu,gaussgrid);
bfn=(da(gaussu,gaussgrid).*dgaussu-b(gaussu,gaussgrid));
cfn=-df(gaussu,gaussgrid);
% SOLVE LINEARIZED DUAL PROBLEM.
% "linphi" IS THE cg1 DUAL SOLUTION
linafn=a(lingaussu,gaussgrid);
linbfn=(da(lingaussu,gaussgrid).*dgaussu-b(lingaussu,gaussgrid));
lincfn=-df(lingaussu,gaussgrid);
[linS,linL,linphi]=cG1(gaussgrid,gaussweights,...
```

```
                        outer,lins,dlins,linafn,linbfn,lincfn);
% SOLVE LINEARIZED DUAL PROBLEM.
% "quadphi" IS THE cg2 DUAL SOLUTION
[quadS,quadL,quadphi]=cG2(gaussgrid,gaussweights,...
                            quads,dquads,dualmesh,afn,bfn,cfn);
% EVALUATE HERMITE BASIS FUNCTIONS ON GAUSSGRID
[hermphi,dhermphi,hermeta,dhermeta,globalphi,dglobalphi,...
        globaleta,dglobaleta,ddglobalphi,ddglobaleta]=...
                                      cubicbasis(gaussgrid,outer);
% SOLVE DUAL PROBLEM USING HERMITE BASIS FUNCTIONS
[cubicS,cubicL,cubic_values,cubic_dual_solution,...
 cubic_first_derivative,cubic_second_derivative]=...
  cG3(gaussgrid,gaussweights, hermphi,dhermphi,hermeta,dhermeta,...
   globalphi,dglobalphi,globaleta,dglobaleta,ddglobalphi,...
    ddglobaleta,afn,bfn,cfn,n);
% INFO FOR GREEN ITERATION
green_x=inv(quadS)';
green_y=green_x';
GM_x=zeros(dm,dm);
GM_y=GM_x;
GM_x(2:end-1,2:end-1)=green_x;
GM_y(2:end-1,2:end-1)=green_y;
% PUT EACH COLUMN OF GREEN'S MATRIX ON THE GAUSSGRID
clear i j;
for j=1:dm
    for i=1:dm
```

```
        fp1(:,i)=GM_x(i,j)*quads(:,i);
        fp2(:,i)=GM_y(i,j)*quads(:,i);
    end
    fp1=sum(fp1')';
    fp2=sum(fp2')';
    gaussGx(:,j)=fp1;
    gaussGy(:,j)=fp2;
    fpp(:,j)=quadphi(j)*quads(:,j);
end
gaussphi=sum(fpp')'; %PUTS DUAL SOLUTION ON GAUSSGRID
gaussmult=da(gaussu,gaussgrid).*cubic_second_derivative+...
    db(gaussu,gaussgrid).*cubic_first_derivative+...
    ddf(gaussu,gaussgrid).*gaussphi;
phifirst=interp1(gaussgrid,cubic_first_derivative,...
         dualmesh,'linear','extrap');
phidub=interp1(gaussgrid,cubic_second_derivative,...
       dualmesh,'linear','extrap');
mult=da(fullU,dualmesh).*phidub+db(fullU,dualmesh)...
     .*phifirst+ddf(fullU,dualmesh).*quadphi;
weights=repmat(gaussweights,1,dm);
E=repmat(gaussmult,1,dm);
Z1=repmat(gausspert,1,dm);
W1=Z1;
iter=60;
[eigv,evect,sing_val,svect]=...
  svdroutine(iter,mult,gaussGx,gaussGy,..
```

```
          E,Z1,W1,weights,dm,quads,gaussgrid);
% RECONSTRUCT PHI-PRIME MATRIX HERE
epsilon=.001;
linPhi_p=buildPhipCG1(epsilon,wholeU,gaussgrid,...
   gaussweights,lins,dlins,linphi,outer);
[linV,linEv]=eig(linPhi_p);
linV1=linV(:,1);
lin_eig=max(abs(diag(linEv)));
[linU,linsingvals,linV]=svd(linPhi_p);
linsingular=max(diag(linsingvals));
linsingvect=linV(:,1);
linsingvect=linsingvect/l2norm(outer,linsingvect(2:end-1));
Phi_p = buildPhipCG2(epsilon,fullU,dm,gaussgrid,...
   gaussweights,quads,dquads,dualmesh,quadphi);
[V,ev]=eig(Phi_p);
[V2,ev2]=eig(Phi_p');
all_eigens=diag(ev);
[p,ip]=max(abs(diag(ev)));
ev=sign(ev(ip))*p; % GIVES DOMINANT EIGENVALUE OF Phi_p
[Uvec,singvals,Vvec]=svd(Phi_p);
singular=max(diag(singvals)); % DOMINANT SINGULAR VALUE
singvals=diag(singvals);
sing_and_eig=[singvals all_eigens];
domin_vec=V(:,1); % DOMINANT EIGENVECTOR
domin_vec=domin_vec/l2norm(dualmesh,domin_vec(2:end-1));
vv=Vvec(:,1);
```

```
vv=vv/l2norm(dualmesh,vv(2:end-1));
vv2=Vvec(1:2:end,1);
vv2=vv2/l2norm(outer,vv2(2:end-1));
%experiment with solving Phi(U+ev)-Phi(U) for different v's
sizes=pert_tests(epsilon,wholeU,gaussgrid,gaussweights,...
    lins,dlins,linphi,outer,linsingvect,linV1);
%%%%%%% PLOT COMMANDS %%%%%%%
figure
subplot(2,2,1)
plot(outer,wholeU,gaussgrid,trueu(gaussgrid));
legend('forward soln');
subplot(2,2,2)
plot(dualmesh,quadphi,gaussgrid,cubic_dual_solution);
legend('quad phi','cubic phi')
subplot(2,2,3)
imagesc(green_x);
title('greens function');
colorbar;
subplot(2,2,4)
imagesc(Phi_p);
title('\Phi'_\psi(U)');
colorbar;
figure
subplot(1,2,1)
%plot(gridfortrue,trueu(gridfortrue),'k','LineWidth',2);
plot(gaussgrid,abs(evect(:,end)),gaussgrid,abs(svect(:,end)));
```

```
%legend('eigen','singular')
subplot(1,2,2)
plot(dualmesh,abs(domin_vec),'r','LineWidth',4);
hold on
plot(gaussgrid,abs(evect(:,end)),'k','LineWidth',2);
legend('matlab','algorithm')
eigenvalues=[ev eigv(end)];
singularvalues=[singular linsingular sing_val(end)];
figure
surfc(dualmesh,dualmesh,GM_x-GM_x');
title('diff of green and transpose');
figure
plot(outer,linphi,dualmesh,quadphi,gaussgrid,cubic_dual_solution);
legend('cg1 dual','cg2dual','hermite dual');
figure
 subplot(2,2,1)
  imagesc(Phi_p);
  colorbar;
 subplot(2,2,2)
  surfc(dualmesh,dualmesh,Phi_p);
 subplot(2,2,3)
  imagesc(linPhi_p);
  colorbar;
 subplot(2,2,4)
  surfc(outer,outer,linPhi_p);
fprintf('eigenvalues: matlab and power method: \n');
```

```
disp(eigenvalues);
fprintf('singular values: matlab and power method: \n');
disp(singularvalues);
```

B.2.2 Secondary programs and functions

The following functions are all called by the main program. The functions gausspoints.m, newton.m, newtfun.m, newtjac.m, and bandedsolver.m were written by Donald Estep. I wrote the others.

```
% FUNCTION gridsetup
% This function sets up the meshes for the forward and
% dual solutions as well as calculates gaussweights and
% nodes for quadrature computations.
% INPUTS:
%    n - the number of interior nodes for the forward solve
%    gaussrule - specify number of gausspoints
%                (should be 2, 3, or 5)
% OUTPUTS:
%    outer - mesh for forward solve, including endpoints
%    inner - interior nodes for forward solve only
%    steps - stepsize for forward solve
%    mids  - midpoints of elements for forward solve
%    dualmesh - mesh for cG2 dual solution
%    dm - size of dualmesh
%    gaussgrid, gaussweights - weights and nodes for quadrature

function [outer,inner,steps,mids,dualmesh,...
          dm,gaussgrid,gaussweights]=gridsetup(n,gaussrule)

% SET UP GRID FOR FORWARD SOLUTION
```

```
outer=linspace(0,1,n+2)';
inner=outer(2:end-1);
steps=diff(outer);
mids=outer(1:end-1)+steps/2;

% SET UP GRID FOR cG2 DUAL SOLUTION (FORWARD GRID + MIDPOINTS)
dualmesh=zeros(length(outer)+length(mids),1);
dualmesh(2:2:end)=mids;
dualmesh(1:2:end)=outer;
dm=length(dualmesh);

% GET GAUSS NODES AND WEIGHTS FOR QUADRATURES
[gaussgrid,gaussweights] = gausspoints(inner,steps,gaussrule);
gaussgrid=reshape(gaussgrid',1,prod(size(gaussgrid)))';
gaussweights=reshape(gaussweights',1,prod(size(gaussweights)))';
```

```
% FUNCTION gausspoints
% Computes the composite 2/3/5 point Gauss quadrature
% INPUTS:
%    x - the mesh
%    h - the stepsize
%    quadres - number of gaussnodes (2, 3, or 5)
% OUTPUTS:
%    tau - the gaussnodes
%    omega - the gaussweights
function [tau,omega] = gausspoints(x,h,quadres)
n=length(x);
if quadres == 2
   gamma = 1/(2*3^.5);
   tau(1,1) = x(1)/2-gamma*h(1);
   tau(1,2)= x(1)/2+gamma*h(1);
   omega(1,1) = .5*h(1);
   omega(1,2) = .5*h(1);
   for i = 2: n
      tau(i,1) = (x(i)+x(i-1))/2-gamma*h(i);
      tau(i,2) = (x(i)+x(i-1))/2+gamma*h(i);
      omega(i,1) = .5*h(i);
      omega(i,2) = .5*h(i);
   end
   tau(n+1,1) = (1+x(n))/2-gamma*h(n+1);
   tau(n+1,2) = (1+x(n))/2+gamma*h(n+1);
   omega(n+1,1) = .5*h(n+1);
```

```
    omega(n+1,2) = .5*h(n+1);
    return
elseif quadres == 3
    gamma = (3/5)^.5;
    tau(1,1) = x(1)/2-gamma*h(1)/2;
    tau(1,2) = x(1)/2;
    tau(1,3)= x(1)/2+gamma*h(1)/2;
    omega(1,1) = 5/18*h(1);
    omega(1,2) = 4/9*h(1);
    omega(1,3) = 5/18*h(1);
    for i = 2: n
        tau(i,1) = (x(i)+x(i-1))/2-gamma*h(i)/2;
        tau(i,2) = (x(i)+x(i-1))/2;
        tau(i,3) = (x(i)+x(i-1))/2+gamma*h(i)/2;
        omega(i,1) = 5/18*h(i);
        omega(i,2) = 4/9*h(i);
        omega(i,3) = 5/18*h(i);
    end
    tau(n+1,1) = (1+x(n))/2-gamma*h(n+1)/2;
    tau(n+1,2) = (1+x(n))/2;
    tau(n+1,3) = (1+x(n))/2+gamma*h(n+1)/2;
    omega(n+1,1) = 5/18*h(n+1);
    omega(n+1,2) = 4/9*h(1);
    omega(n+1,3) = 5/18*h(n+1);
    return
elseif quadres == 5
```

```
    xi(1)=-sqrt(245+14*sqrt(70))/21;
    xi(2)=-sqrt(245-14*sqrt(70))/21;
    xi(3)=0;
    xi(4)=sqrt(245-14*sqrt(70))/21;
    xi(5)=sqrt(245+14*sqrt(70))/21;
    w(1)=.23692688505618908752;
    w(2)=.47862867049936646810;
    w(3)=.56888888888888888889;
    w(4)=w(2);
    w(5)=w(1);
    for j = 1: 5
       tau(1,j)=(x(1)+h(1)*xi(j))/2;
       omega(1,j)=w(j)*h(1)/2;
    end
    for i = 2: n
       for j = 1: 5
          tau(i,j)=(x(i)+x(i-1)+h(i)*xi(j))/2;
          omega(i,j)=w(j)*h(i)/2;
       end
    end
    for j = 1: 5
       tau(n+1,j)=(1+x(n)+h(n+1)*xi(j))/2;
       omega(n+1,j)=w(j)*h(n+1)/2;
    end
end
```

```
% FUNCTION newton
% hybrid-Newton iteration for solving (a(u,x)u')'+b(u,x)u'=f(u,x)
% using continuous piecewise linear elements and trapezoidal
% rule to evaluate the integrals
% the matrices are created as column vectors and then
% turned into a sparse banded matrix
% INPUTS:
%    uold - previous Newton iterate
%    x, h - mesh and stepsize
%    Ntol, Nmax - tolerance and maximum iterations
% OUTPUT:
%    u - final solution to the nonlinear system
function u = newton(uold,x,h,Ntol,Nmax)
n=length(x);
Fold=newtfun(uold,x,h);
for newt = 1:Nmax
   JFold=newtjac(uold,x,h);
   Ndel = bandedsolver(JFold,Fold,3);
   u = uold - Ndel;
   Fu=newtfun(u,x,h);
   % stop if Newton change is small or
   % if relative residual is small
   if norm(u-uold) < Ntol
      break
   end
   % If the Newton step increases the residual,
```

```
    % do a bisection search to find a step
    % that decreases the residual
    sc=0;
    while norm(Fu)> norm(Fold)
       Ndel=Ndel/2;
       u = uold - Ndel;
       Fu=newtfun(u,x,h);
       sc=sc+1;
    end
    if sc > 0
        fprintf('      Did %d bisection steps
                              at Newton step %d\n',sc,newt)
    end
    uold = u;
    Fold = Fu;
end
fprintf('   Used %d Newton iterations\n',newt-1)
```

```
% FUNCTION newtfun
% COMPUTES THE FUNCTION VALUE FOR THE NEWTON SOLVE
% INPUTS:
%    u - function values on mesh
%    x - mesh for forward problem
%    h - stepsize
% OUTPUTS:
%    F - function value for Newton solve
function F = newtfun(u,x,h)
n=length(x);
F(1,1) = (1/(2*h(1))*a(0,0) + (1/(2*h(1)))...
   +1/(2*h(2)))*a(u(1),x(1)))...
   +1/(2*h(2))*a(u(2),x(2)))*u(1)...
   -(a(u(1),x(1))+a(u(2),x(2)))/(2*h(2))*u(2)...
   +b(u(1),x(1))*u(2)/2.0...
   -(h(1)+h(2))/2*f(u(1),x(1));
for i = 2: n-1
   F(i,1) = -(a(u(i-1),x(i-1))+a(u(i),x(i)))/(2*h(i))*u(i-1)...
      +(1/(2*h(i))*a(u(i-1),x(i-1))...
      + (1/(2*h(i))+1/(2*h(i+1)))*a(u(i),x(i))...
      +1/(2*h(i+1))*a(u(i+1),x(i+1)))*u(i)...
      -(a(u(i),x(i))+a(u(i+1),x(i+1)))/(2*h(i+1))*u(i+1)...
      +b(u(i),x(i))*(u(i+1)-u(i-1))/2.0...
      -(h(i)+h(i+1))/2*f(u(i),x(i));
end
F(n,1) = -(a(u(n-1),x(n-1))+a(u(n),x(n)))/(2*h(n))*u(n-1)...
```

```
+(1/(2*h(n))*a(u(n-1),x(n-1))...
+ (1/(2*h(n))+1/(2*h(n+1)))*a(u(n),x(n))...
+1/(2*h(n+1))*a(0,1))*u(n)...
-b(u(n),x(n))*u(n-1)/2.0...
-(h(n)+h(n+1))/2*f(u(n),x(n));
```

```
% FUNCTION newtjac
% COMPUTES THE JACOBIAN FOR THE NEWTON ITERATION
% INPUTS:
%    u, x, h: function value u, mesh x, stepsize h
% OUTPUT:
%    Jf: Jacobian of F at u
function JF=newtjac(u,x,h)
n=length(x);
for i = 1: n-1
   JF(i,1) = -(a(u(i+1),x(i+1))+a(u(i),x(i)))/(2*h(i+1))...
      +da(u(i),x(i))/(2*h(i+1))*(u(i+1)-u(i))...
      -b(u(i+1),x(i+1))/2.0;
end
%dummy
JF(n,1)=0;
JF(1,2) = da(u(1),x(1))/(2*h(1))*u(1)...
   +da(u(1),x(1))/(2*h(2))*(u(1)-u(2))...
   +1/(2*h(1))*a(0,0)+(1/(2*h(1))+1/(2*h(2)))*a(u(1),x(1))...
   +1/(2*h(2))*a(u(2),x(2))...
   +db(u(1),x(1))*u(2)/2.0...
   -(h(1)+h(2))/2*df(u(1),x(1));
for i = 2:n-1
   JF(i,2) = da(u(i),x(i))/(2*h(i))*(u(i)-u(i-1))...
      +da(u(i),x(i))/(2*h(i+1))*(u(i)-u(i+1))...
      +1/(2*h(i))*a(u(i-1),x(i-1))+(1/(2*h(i))...
      +1/(2*h(i+1)))*a(u(i),x(i))...
```

```
      +1/(2*h(i+1))*a(u(i+1),x(i+1))...
      +db(u(i),x(i))*(u(i+1)+u(i-1))/2.0...
      -(h(i)+h(i+1))/2*df(u(i),x(i));
end
JF(n,2) = da(u(n),x(n))/(2*h(n))*(u(n)-u(n-1))...
   +da(u(n),x(n))/(2*h(n+1))*u(n)...
   +1/(2*h(n))*a(u(n-1),x(n-1))+(1/(2*h(n))...
   +1/(2*h(n+1)))*a(u(n),x(n))...
   +1/(2*h(n+1))*a(0,1)...
   +db(u(n),x(n))*u(n-1)/2.0...
   -(h(n)+h(n+1))/2*df(u(n),x(n));
%dummy
JF(1,3)=0;
for i = 2:n
   JF(i,3)=da(u(i),x(i))/(2*h(i))*(u(i-1)-u(i))...
      -(a(u(i-1),x(i-1))+a(u(i),x(i)))/(2*h(i))...
      +b(u(i-1),x(i-1))/2.0;
end
```

```
% FUNCTION bandedsolver
% SETS UP A BANDED SPARSE MATRIX USING DIAGONALS
% COMPUTED FROM Ac AND SOLVES THE SYSTEM Ax=b.
% d IS THE # OF DIAGONALS CENTERED AROUND MAIN DIAGONAL
function x = bandedsolver(Ac,b,d)
n=length(b);
d=fix(d/2);
A=spdiags(Ac,-d:d,n,n);
x=A\b;

% FUNCTION l2norm
% compute l2 norm of a function on [0,1] defined at gridpoints
% INPUTS:
%    grid - mesh with endpoints
%    func - function values on interior meshpoints
% OUTPUT:
%   - y is the l2 norm of the function
% NOTE: only for functions which are zero at the endpoints
function y=l2norm(grid,func)
ingrid=grid(2:end-1);
d=diff(grid);
[tau,omega]=gausspoints(ingrid,d,5);
tau=reshape(tau',1,prod(size(tau)))';
omega=reshape(omega',1,prod(size(omega)))';
p=interp1(ingrid,func,tau,'spline');
y=sqrt(sum(omega.*p.^2));
```

```
% FUNCTION linbasis.m
% Evaluate linear Lagrange basis function on grid of gausspoints
% INPUTS
%   g - grid of gausspoints, obtained from  "gausspoints.m"
%   x - grid for forward solution, including the endpoints
%   d - stepsize between x nodes
% OUTPUTS
%   lins,  - Linear Lagrange basis functions
%   dlins  - Derivative of lins
function [lins,dlins]=linbasis(g,x,d);
n=length(x);
       lins(:,1)=((x(2)-g)/d(1)).*(g <= x(2));
       dlins(:,1)=(-1/d(1)).*(g <= x(2));
    for i=2:n-1
        lins(:,i)=(g-x(i-1))/d(i-1).*(g > x(i-1)...
                  & g <= x(i))+(x(i+1)-g)/d(i)...
                  .*(g > x(i) & g <= x(i+1));
        dlins(:,i)=1/d(i-1).*(g > x(i-1) & g <= x(i))...
                  +(-1/d(i).*(g > x(i) & g <= x(i+1)));
    end
       lins(:,n)=(g-x(n-1))/d(n-1).*(g > x(n-1));
       dlins(:,n)=(1/d(n-1)).*(g > x(n-1));
```

```
% FUNCTION quadbasis.m
% Evaluate quadratic Lagrange basis function on gausspoints
% INPUTS
%   g - grid of gausspoints, obtained from  "gausspoints.m"
%   x - grid for the forward solution, including the endpoints
%   mids -  the midpoint values between forward nodes
%   y  - full mesh for dual solution
% OUTPUTS
%   quads, dquads - Lagrange quadratic basis functions and
%   their derivatives, evaluated at the gausspoints
function[quads,dquads]=quadbasis(g,x,d,mids,y);
n=length(mids); m=length(g); p=length(y);
    for i=1:n
        right(:,i)=(2*(g-y(2*i))...
                   .*(g-y(2*i+1))/(y(2*i+1)-y(2*i-1))^2)...
                   .*(g > y(2*i-1) & g <= y(2*i+1));
        dright(:,i)=2*((2*g-y(2*i+1)...
                    -y(2*i))/((y(2*i+1)-y(2*i-1))^2))...
                    .*(g > y(2*i-1) & g <= y(2*i+1));
        middle(:,i)=(4*(y(2*i+1)-g)...
                    .*(g-y(2*i-1))/(y(2*i+1)-y(2*i-1))^2)...
                    .*(g >= y(2*i-1) & g <= y(2*i+1));
        dmiddle(:,i)=(-4*(2*g-y(2*i-1)...
                     -y(2*i+1))/((y(2*i+1)-y(2*i-1))^2))...
                     .*(g >= y(2*i-1) & g <= y(2*i+1));
        left(:,i)=(2*(g-y(2*i))...
```

```
                .*(g-y(2*i-1))/(y(2*i+1)-y(2*i-1))^2)...
                .*(g >= y(2*i-1) & g <= y(2*i+1));
    dleft(:,i)=(2*(2*g-y(2*i-1)...
                -y(2*i))/((y(2*i+1)-y(2*i-1))^2))...
                .*(g >= y(2*i-1) & g <= y(2*i+1));
end
for i=1:n-1
    interior(:,i)=left(:,i)+right(:,i+1);
    dinterior(:,i)=dleft(:,i)+dright(:,i+1);
end
quads=zeros(m,p);
dquads=zeros(m,p);
quads(:,1)=right(:,1);
dquads(:,1)=dright(:,1);
for i=2:2:p-3
    quads(:,i)=middle(:,i/2);
    dquads(:,i)=dmiddle(:,i/2);
    quads(:,i+1)=interior(:,i/2);
    dquads(:,i+1)=dinterior(:,i/2);
end
quads(:,p-1)=middle(:,end);
dquads(:,p-1)=dmiddle(:,end);
quads(:,p)=left(:,end);
dquads(:,p)=dleft(:,end);
```

```
% FUNCTION cubicbasis.m
% evaluate cubic Hermite basis functions on gausspoints
% INPUTS
%   g - grid of gausspoints, obtained from "gausspoints.m"
%   x - grid for forward solution
% OUTPUTS
%   hermphi,dhermphi,hermeta,dhermeta-
%       "local" basis functions and derivatives
%   globalphi,dglobalphi,globaleta,dglobaleta
%      "global" basis functions and derivatives
%   ddglobalphi,ddglobaleta
%      second derivatives of basis functions, to express
%      second derivative of dual solution

function [hermphi,dhermphi,hermeta,dhermeta,...
          globalphi,dglobalphi,globaleta,dglobaleta,...
          ddglobalphi,ddglobaleta]=cubicbasis(g,x)
n=length(x);
for i=1:n-1
p1=(2*(g-x(i+1))./(x(i)-x(i+1)))+1;
p2=(2*(g-x(i))./(x(i+1)-x(i)))+1;
leftphi(:,i) =(((g-x(i)).^2).*p1)./(x(i+1)-x(i)).^2...
              .*(g >= x(i) & g <= x(i+1));
rightphi(:,i)=(((g-x(i+1)).^2).*p2)./(x(i)-x(i+1)).^2...
              .*(g >= x(i) & g <= x(i+1));
dleftphi(:,i)=(2*(g-x(i)).*(2*(g-x(i+1))./(x(i)-x(i+1))+1)...
```

```
        ./((x(i+1)-x(i)).^2)+2*(g-x(i)).^2./((x(i)-x(i+1))...
        .*(x(i+1)-x(i)).^2)).*(g >= x(i) & g <= x(i+1));
drightphi(:,i)=(2*(g-x(i+1)).*(2*(g-x(i))./(x(i+1)-x(i))+1)...
        ./((x (i)-x(i+1)).^2)+2*(g-x(i+1)).^2./((x(i+1)-x(i))...
        .*(x(i)-x(i+1)).^2)).*(g >= x(i) & g <= x(i+1));
ddleftphi(:,i)=6*(2*g-x(i+1)-x(i))/((x(i)-x(i+1))^3)...
                .*(g >= x(i) & g <= x(i+1));
ddrightphi(:,i)=-6*(2*g-x(i+1)-x(i))/((x(i)-x(i+1))^3)...
                 .*(g >= x(i) & g <= x(i+1));
lefteta(:,i)=(((g-x(i+1)).*(g-x(i)).^2)./(x(i+1)-x(i)).^2)...
              .*(g >= x(i) & g <= x(i+1));
righteta(:,i)=(((g-x(i)).*(g-x(i+1)).^2)./(x(i+1)-x(i)).^2)...
               .*(g >= x(i) & g <= x(i+1));
dlefteta(:,i)=((g-x(i)).^2./((x(i+1)-x(i)).^2)+2*(g-x(i+1))...
    .*(g-x(i))./((x(i+1)-x(i)).^2)).*(g >= x(i) & g <= x(i+1));
drighteta(:,i)=((g-x(i+1)).^2./((x(i+1)-x(i)).^2)+2*(g-x(i))...
    .*(g-x(i+1))./((x(i+1)-x(i)).^2)).*(g >= x(i) & g <= x(i+1));
ddlefteta(:,i)=2*(3*g-2*x(i)-x(i+1))/((x(i+1)-x(i))^2)...
                .*(g >= x(i) & g <= x(i+1));
ddrighteta(:,i)=2*(3*g-2*x(i+1)-x(i))/((x(i+1)-x(i))^2)...
                 .*(g >= x(i) & g <= x(i+1));
end
hermphi=[leftphi rightphi];
dhermphi=[dleftphi drightphi];
ddhermphi=[ddleftphi ddrightphi];
hermeta=[lefteta righteta];
```

```
dhermeta=[dlefteta drighteta];
ddhermeta=[ddlefteta ddrighteta];
[r,c]=size(hermphi);
k=c/2;
globalphi(:,1)=hermphi(:,k+1);
dglobalphi(:,1)=dhermphi(:,k+1);
ddglobalphi(:,1)=ddhermphi(:,k+1);
globaleta(:,1)=hermeta(:,k+1);
dglobaleta(:,1)=dhermeta(:,k+1);
ddglobaleta(:,1)=ddhermeta(:,k+1);
for i=2:k
    globalphi(:,i)=hermphi(:,i-1)+hermphi(:,i+k);
    dglobalphi(:,i)=dhermphi(:,i-1)+dhermphi(:,i+k);
    ddglobalphi(:,i)=ddhermphi(:,i-1)+ddhermphi(:,i+k);
    globaleta(:,i)=hermeta(:,i-1)+hermeta(:,i+k);
    dglobaleta(:,i)=dhermeta(:,i-1)+dhermeta(:,i+k);
    ddglobaleta(:,i)=ddhermeta(:,i-1)+ddhermeta(:,i+k);
end
globalphi(:,k+1)=hermphi(:,k);
dglobalphi(:,k+1)=dhermphi(:,k);
ddglobalphi(:,k+1)=ddhermphi(:,k);
globaleta(:,k+1)=hermeta(:,k);
dglobaleta(:,k+1)=dhermeta(:,k);
ddglobaleta(:,k+1)=ddhermeta(:,k);
```

```
% FUNCTION cG1 -
% computes stiffness matrix, load vector and solution
% for the linearized dual problem
%-(a(u)phi')'+a'(u)u'phi'-b(u)phi'-f'(u)=psi
% using lagrange linear basis (hat) functions
% INPUTS:
%    gaussgrid, gaussweights-  gaussnodes and weights,
%    obtained from "gausspoints.m" routine,
%    organized into columns
%    lins, dlins- hat functions and derivatives
%   afn, bfn, cfn - the coefficients of the dual problem:
%   afn = a(U), bfn = (a'(U)U'-b(U)), cfn = -f'(U)
% OUTPUTS:
%   S - Stiffness matrix for cG1 finite element method
%   L - Load vector for cG1 finite element method
%   phi - values of dual solution on nodes
function [S,L,phi]=...
         cG1(gaussgrid,gaussweights,outer,lins,dlins,afn,bfn,cfn)
% CONSTRUCT LOAD VECTOR
p=psi(gaussgrid);
m=length(outer);
for i=1:m
    load(i,1)=sum(gaussweights.*p.*lins(:,i));
end
L=load(2:end-1);
% CONSTRUCT STIFFNESS MATRIX
```

```
 lins=lins(:,2:end-1);
 dlins=dlins(:,2:end-1);
 S=zeros(m-2,m-2);
[r c]=size(lins);
j=1;
for i=j:j+1;
    S(1,i)=sum(gaussweights.*...
           (afn.*dlins(:,i).*dlins(:,j)...
           +bfn.*dlins(:,i).*lins(:,j)...
           +cfn.*lins(:,i).*lins(:,j)));
end
for j=2:c-1;
    for i=j-1:j+1
        S(j,i)=sum(gaussweights.*...
               (afn.*dlins(:,i).*dlins(:,j)...
               +bfn.*dlins(:,i).*lins(:,j)...
               +cfn.*lins(:,i).*lins(:,j)));
    end
end
j=c;
for i=j-1:j;
    S(c,i)=sum(gaussweights.*...
           (afn.*dlins(:,i).*dlins(:,j)...
           +bfn.*dlins(:,i).*lins(:,j)...
           +cfn.*lins(:,i).*lins(:,j)));
end
phi=[0;S\L;0];
```

```
% FUNCTION cG2
% computes stiffness matrix, load vector and solution
% for the linearized dual problem
% -(a(u)phi')'+a'(u)u'phi'-b(u)phi'-f'(u)=psi
% using lagrange quadratic basis functions
% INPUTS:
%    gaussgrid, gaussweights - nodes and weights, obtained from
%    "gausspoints.m" routine, organized into columns
%    quads, dquads - basis functions, evaluated on
%            gauss nodes, obtained from "quadbasis.m" routine
%   dualmesh - the forward grid + midpoints
%   afn, bfn, cfn - the coefficients of the dual problem:
%   afn = a(U), bfn = (a'(U)U'-b(U)), cfn = -f'(U)
% OUTPUTS:
%   S - Stiffness matrix for cG2 finite element method
%   L - Load vector for cG2 finite element method
%   phi - values of dual solution

function [S,L,phi]=...
          cG2(gaussgrid,gaussweights,quads,...
                    dquads,dualmesh,afn,bfn,cfn)
% CONSTRUCT LOAD VECTOR
p=psi(gaussgrid);
m=length(dualmesh);
for i=1:m
    load(i,1)=sum(gaussweights.*p.*quads(:,i));
```

```
end
L=load(2:end-1);
% CONSTRUCT STIFFNESS MATRIX
quads=quads(:,2:end-1);
dquads=dquads(:,2:end-1);
S=zeros(m-2,m-2);
[r c]=size(S);
j=1;
for i=j:j+1;
    S(1,i)=sum(gaussweights.*...
           (afn.*dquads(:,i).*dquads(:,j)...
           +bfn.*dquads(:,i).*quads(:,j)...
           +cfn.*quads(:,i).*quads(:,j)));
end
j=2;
for i=j-1:j+2;
    S(2,i)=sum(gaussweights.*...
           (afn.*dquads(:,i).*dquads(:,j)...
           +bfn.*dquads(:,i).*quads(:,j)...
           +cfn.*quads(:,i).*quads(:,j)));
end
for j=3:m-4;
    for i=j-2:j+2
        S(j,i)=sum(gaussweights.*...
               (afn.*dquads(:,i).*dquads(:,j)...
               +bfn.*dquads(:,i).*quads(:,j)...
```

```
                +cfn.*quads(:,i).*quads(:,j)));
    end
end
j=m-3;
for i=j-2:j+1;
    S(j,i)=sum(gaussweights.*...
            (afn.*dquads(:,i).*dquads(:,j)...
            +bfn.*dquads(:,i).*quads(:,j)...
            +cfn.*quads(:,i).*quads(:,j)));
end
j=m-2;
for i=j-1:j;
    S(j,i)=sum(gaussweights.*...
            (afn.*dquads(:,i).*dquads(:,j)...
            +bfn.*dquads(:,i).*quads(:,j)...
            +cfn.*quads(:,i).*quads(:,j)));
end
phi=[0;S\L;0];
```

```
% FUNCTION cG3.m
% computes stiffness matrix, load vector and solution
% for the linearized dual problem
% using hermite cubic basis functions
% INPUTS:
%    gaussgrid,gaussweights -  gaussnodes and weights
%   hermphi,dhermphi,hermeta,dhermeta,
%   globalphi,dglobalphi,globaleta, dglobaleta
%      - cubic hermite basis functions and their derivatives,
%        evaluated on gauss nodes
%   afn, bfn, cfn - the coefficients of the dual problem:
%   afn = a(U), bfn = (a'(U)U'-b(U)), cfn = -f'(U)
% OUTPUTS:
%   S - Stiffness matrix for cG3 (Hermite) finite element method
%   L - Load vector
%   cubic_values - first column is values of dual solution
%      second column is values of derivative of dual solution
%   cubic_dual_solution - cubic dual expressed on gaussnodes
%   cubic_first_derivative
%      - first derivative of dual on gaussnodes
%   cubic_second_derivative
%      - second derivative of dual on gaussnodes
function [S,L,cubic_values,cubic_dual_solution,...
          cubic_first_derivative,cubic_second_derivative]=...
          cG3(gaussgrid,gaussweights, hermphi,dhermphi,...
          hermeta,dhermeta,globalphi,dglobalphi,globaleta,...
```

```
          dglobaleta,ddglobalphi,ddglobaleta,afn,bfn,cfn,n);
% CONSTRUCT LOAD VECTOR
[r1 c1]=size(hermeta);
[r2 c2]=size(globaleta);
p=psi(gaussgrid);
for i=1:c2
    bL(:,i)=p.*globaleta(:,i);
    tL(:,i)=p.*globalphi(:,i);
end
loadmatrix=[tL bL];
[r3 c3]=size(loadmatrix);
gw=repmat(gaussweights,1,c3);
gaussmatrix=gw.*loadmatrix;
load=sum(gaussmatrix)';
% BUILD STIFFNESS MATRIX
S=zeros(2*(n+2),2*(n+2));
[r c]=size(S);
k=n+2;
j=1;
for i=j:j+1;
    S(1,i)=sum(gaussweights.*...
                (afn.*dglobalphi(:,i).*dglobalphi(:,j)...
                +bfn.*dglobalphi(:,i).*globalphi(:,j)...
                +cfn.*globalphi(:,i).*globalphi(:,j)));
    S(1,i+k)=sum(gaussweights.*...
                (afn.*dglobaleta(:,i).*dglobalphi(:,j)...
```

```
                    +bfn.*dglobaleta(:,i).*globalphi(:,j)...
                    +cfn.*globaleta(:,i).*globalphi(:,j)));
    S(1+k,i)=sum(gaussweights.*...
                    (afn.*dglobalphi(:,i).*dglobaleta(:,j)...
                    +bfn.*dglobalphi(:,i).*globaleta(:,j)...
                    +cfn.*globalphi(:,i).*globaleta(:,j)));
    S(1+k,i+k)=sum(gaussweights.*...
                    (afn.*dglobaleta(:,i).*dglobaleta(:,j)...
                    +bfn.*dglobaleta(:,i).*globaleta(:,j)...
                    +cfn.*globaleta(:,i).*globaleta(:,j)));
end
for j=2:n+1;
     for i=j-1:j+1
        S(j,i)=sum(gaussweights.*...
                    (afn.*dglobalphi(:,i).*dglobalphi(:,j)...
                    +bfn.*dglobalphi(:,i).*globalphi(:,j)...
                    +cfn.*globalphi(:,i).*globalphi(:,j)));
        S(j,i+k)=sum(gaussweights.*...
                    (afn.*dglobaleta(:,i).*dglobalphi(:,j)...
                    +bfn.*dglobaleta(:,i).*globalphi(:,j)...
                    +cfn.*globaleta(:,i).*globalphi(:,j)));
        S(j+k,i)=sum(gaussweights.*...
                    (afn.*dglobalphi(:,i).*dglobaleta(:,j)...
                    +bfn.*dglobalphi(:,i).*globaleta(:,j)...
                    +cfn.*globalphi(:,i).*globaleta(:,j)));
        S(j+k,i+k)=sum(gaussweights.*...
```

```
                (afn.*dglobaleta(:,i).*dglobaleta(:,j)...
                +bfn.*dglobaleta(:,i).*globaleta(:,j)...
                +cfn.*globaleta(:,i).*globaleta(:,j)));
    end
end
j=n+2;
for i=j-1:j;
    S(j,i)=sum(gaussweights.*...
                (afn.*dglobalphi(:,i).*dglobalphi(:,j)...
                +bfn.*dglobalphi(:,i).*globalphi(:,j)...
                +cfn.*globalphi(:,i).*globalphi(:,j)));
        S(j,i+k)=sum(gaussweights.*...
                (afn.*dglobaleta(:,i).*dglobalphi(:,j)...
                +bfn.*dglobaleta(:,i).*globalphi(:,j)...
                +cfn.*globaleta(:,i).*globalphi(:,j)));
        S(j+k,i)=sum(gaussweights.*...
                (afn.*dglobalphi(:,i).*dglobaleta(:,j)...
                +bfn.*dglobalphi(:,i).*globaleta(:,j)...
                +cfn.*globalphi(:,i).*globaleta(:,j)));
        S(j+k,i+k)=sum(gaussweights.*...
                (afn.*dglobaleta(:,i).*dglobaleta(:,j)...
                +bfn.*dglobaleta(:,i).*globaleta(:,j)...
                +cfn.*globaleta(:,i).*globaleta(:,j)));
end
% REMOVE APPROPRIATE ROWS AND COLUMNS OF STIFFNESS
% MATRIX AND LOAD VECTOR SO THAT THE
```

```
% SYSTEM SATISFIES DIRICHLET CONDITIONS
S=[S2:n+1,2:n+1) S(2:n+1,k+1:end);...
      S(k+1:end,2:n+1) S(k+1:end,k+1:end)];
L=[load(2:n+1) ; load(k+1:end)];
solution=S\L;
% MAKE COLUMN FOR FUNCTION VALUES AND
% COLUMN FOR DERIVATIVE VALUES
% ADD ZEROS AT ENDPOINTS FOR DIRICHLET CONDITIONS
cubic_values = [ [0; solution(1:n); 0] solution(n+1:end)];
% EXPRESS SOLUTION ON GAUSSGRID
[r c]=size(cubic_values);
for i=1:r
    cubic(:,i)=cubic_values(i,1)*globalphi(:,i)...
   +cubic_values(i,2)*globaleta(:,i);
    cubic_first(:,i)=cubic_values(i,1)*dglobalphi(:,i)...
   +cubic_values(i,2)*dglobaleta(:,i);
    cubic_sec(:,i)=cubic_values(i,1)*ddglobalphi(:,i)...
   +cubic_values(i,2)*ddglobaleta(:,i);
end
cubic_dual_solution=sum(cubic')';
cubic_first_derivative=sum(cubic_first')';
cubic_second_derivative=sum(cubic_sec')';
```

```
%need comments about inputs and outputs of this function
function linPhi_p=buildPhipCG1(epsilon,wholeU,gaussgrid,...
                  gaussweights,lins,dlins,linphi,outer)
m=length(wholeU);
pmat=epsilon*eye(m); % matrix of perturbations
pmat(:,1)=0;
pmat(:,end)=0; %don't perturb first and last cols-
perturbed=repmat(wholeU,1,m)+pmat;
linphimat=repmat(linphi,1,m);
for i=1:m
    for j=1:m
        dnu(:,j)=perturbed(j,i)*dlins(:,j);
    end
    dnewperturbed(:,i)=sum(dnu')';
    lingaussu=interp1(outer,perturbed(:,i),gaussgrid);
     afn = a(lingaussu,gaussgrid);
     bfn = da(lingaussu,gaussgrid)...
           .*dnewperturbed(:,i)-b(lingaussu,gaussgrid);
     cfn = -df(lingaussu,gaussgrid);
 [linS,linL,pertphis(:,i)]=...
     cG1(gaussgrid,gaussweights,outer,lins,dlins,afn,bfn,cfn);
 clear linS linL afn bfn cfn;
end
linPhi_p=pertphis-linphimat;
linPhi_p=linPhi_p/epsilon;
```

```
%need comments about what function does, inputs and outputs
function Phi_p=buildPhipCG2(epsilon,fullU,dm,...
           gaussgrid,gaussweights,quads,dquads,dualmesh,quadphi)
pmat=epsilon*eye(dm);
pmat(1,1)=0;
pmat(end,end)=0;
perturbed=repmat(fullU,1,dm)+pmat;
quadphimat=repmat(quadphi,1,dm);
for i=1:dm
    for j=1:dm
        nu(:,j)=perturbed(j,i)*quads(:,j);
        dnu(:,j)=perturbed(j,i)*dquads(:,j);
    end
    newperturbed(:,i)=sum(nu')';
    dnewperturbed(:,i)=sum(dnu')';
    afn(:,i) = a(newperturbed(:,i),gaussgrid);
    bfn(:,i) = da(newperturbed(:,i),gaussgrid)...
               .*dnewperturbed(:,i)-b(newperturbed(:,i),gaussgrid);
    cfn(:,i) = -df(newperturbed(:,i),gaussgrid);
    [quadS,quadL,pertphis(:,i)]=...
        cG2(gaussgrid,gaussweights,quads,dquads,...
            dualmesh,afn(:,i),bfn(:,i),cfn(:,i));
    clear quadS quadL;
end
Phi_p=(pertphis-quadphimat)/epsilon;
```

```
% function svdroutine
% need some comments explaining what this function does
% and inputs and outputs
function [eigv,evect,sing_val,svect]=
         svdroutine(iter,mult,gaussGx,gaussGy,...
                    E,Z1,W1,weights,dm,quads,gaussgrid)
for k=1:iter
    z=sum(gaussGx.*E.*Z1.*weights)'; %first integral
    w=sum(gaussGx.*E.*W1.*weights)'; % w's are for power method
    clear i;
    for i=1:dm
        temp(:,i)=z(i)*quads(:,i);
        temp2(:,i)=w(i)*quads(:,i);
    end
    z=sum(temp')'; % put result of first integral
                   % back on gaussgrid
    w=sum(temp2')';
    eigv(k,1)=sum(w.*W1(:,1).*weights(:,1));
    w=w/l2norm(gaussgrid,w(2:end-1));
    evect(:,k)=w; %save normalized eigenvector estimates
    W1=repmat(w,1,dm);
    Z2=repmat(z,1,dm);
    znew=(sum(gaussGy.*Z2.*weights)').*mult; %second integral
    clear i temp;
    for i=1:dm
        temp(:,i)=znew(i)*quads(:,i);
```

```
    end
    znew=sum(temp')'; % put result of second integral
                      % back on gaussgrid
    sing_val(k,1)=sqrt(sum(znew.*Z1(:,1).*weights(:,1)));
    znew=znew/l2norm(gaussgrid,znew(2:end-1)); %normalize
    svect(:,k)=znew; %save singular vector estimates
    Z1=repmat(znew,1,dm);
end
```