Improving the Finite Difference Approximation in the Jacobian-Free Newton-Krylov Method

Robert Loek Van Heyningen

Bachelor's thesis
2016:K9

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Numerical Analysis

Centrum Scientiarum Mathematicarum

LUND UNIVERSITY

Abstract

Numerical Analysis
Centre for Mathematical Sciences

Improving the Finite Difference Approximation in the Jacobian-Free Newton–Krylov Method

by Robert Loek Van Heyningen

The Jacobian-free Newton–Krylov (JFNK) method is designed to solve a linear system of equations that appears in Newton's method. It uses the generalized minimal residual (GMRES) method to solve the linear system and a simple function to approximate the matrix-vector multiplications required in GMRES. An advantage of GMRES is the ability to check the residual of a potential solution without doing any extra computations. A previous bachelor's thesis discovered that the residual seen by the algorithm and the actual residual differ across many test cases. This puts the validity of the solution into question and makes it difficult to implement any sort of error-checking in the algorithm. The purpose of this thesis is to investigate the discrepancy between these residuals. Tests were run on the various problems and although they were unable to determine a concrete explanation for the behavior of the residuals, they provided valuable insight into the potential causes to investigate in the future.

Acknowledgements

I would first like to thank my advisor Philipp Birken for his willingness to take a chance on an exchange student and his constant guidance and advice throughout the course of the semester. I would also like to thank some of my past teachers including Jon Wilkening, John Strain, and Lin Lin, for helping to cultivate my interest in the subject. This would not be possible without the help of the study abroad coordinators and staff at Lund University and the University of California at Berkeley. Lastly, I would like to thank my parents and brother for encouraging me to take on this challenge.

Contents

Chapter 1

Introduction

The solving of large systems of linear equations is vital to the simulation of most physical phenomena. The steps in setting up and solving systems of differential equations, from the smallest test problems to massive industrial simulations, will typically end up relying on a fast and dependable way to solve $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^n$.

In the fluid problems of the type dealt with in this thesis, the matrix A is usually very large and sparse, or consisting of mostly zeros [7]. One method that is used regularly to solve general large sparse linear systems is the generalized minimal residual method (GMRES) introduced by Saad and Schultz in 1986 [8].

This thesis focuses on a specific implementation of GMRES in the context of Newton's method. In this context, the matrix A is the Jacobian of the system, which can be approximated in a convenient way with a finite difference. A previous thesis by Ivo Dravins discovered some unexpected inconsistencies in the algorithm, however. Section 2.1 and 2.2 elaborate on the set up of the linear system. The next two sections discuss the theory and implementation of GMRES. Chapter 2 ends by going over the specific goals and methods of this thesis, while the final two chapters focus on presenting the data and analyzing it.

Chapter 2

The Jacobian-Free Newton–Krylov Method

2.1 Background

Often times, it is useful to know how we arrived at solving this linear system. For fluids, the procedure is usually as follows [3]:

- We discretize the system of partial differential equations (PDEs) in space to get a system of ordinary differential equations (ODEs)
- We use some sort of time integration method to solve the system of ODEs over time. When we discretize the PDEs, we usually end up with a stiff system of ODEs, which is best integrated by an implicit integration scheme [2].
- To solve the implicit scheme, we need some sort of iterative method.
- Within this iterative method, we will typically need to solve a linear system of equations.

Though is there much to be said about the first two steps, it is the last two that are relevant to us.

2.2 Newton's Method

In order to complete a single step in the implicit integration scheme, we need to solve

$$\mathbf{F}(\mathbf{u}) = 0$$

$$\mathbf{F} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}, \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

where $\mathbf{F}(u)$ is a non-linear function of $u \in \mathbb{R}^n$. An efficient and dependable way to find the zero of $\mathbf{F}$ is Newton's Method [2]. Starting with an initial guess $\mathbf{u}_0$, the Newton iteration is as follows:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta\mathbf{u}_k \tag{2.1}$$

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\right|_{\mathbf{u}_k} \Delta\mathbf{u_k} = -\mathbf{F}(\mathbf{u}_k) \tag{2.2}$$

where $\frac{\partial \mathbf{F}}{\partial \mathbf{u}}$ is the Jacobian of $\mathbf{F}$

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & \frac{\partial F_1}{\partial u_n} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} & \cdots & \frac{\partial F_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial u_1} & \frac{\partial F_n}{\partial u_2} & \cdots & \frac{\partial F_n}{\partial u_n} \end{pmatrix}. \tag{2.3}$$

In order to do the Newton iteration, we need to solve the linear system (2.2) to get $\Delta\mathbf{u}_k$.

2.3 The Generalized Minimal Residual Method

The problem is reduced to solving a large system of linear equations

$$Ax = b \tag{2.4}$$

where $A \in \mathbb{R}^{n\times n}$ consists mostly of zeros. An effective way to solve such systems is the generalized minimal residual (GMRES) method [2]. Before describing the method, we first must define the m-th Krylov subspace of $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$ as

$$\mathcal{K}_m(A, x) = \text{span}\{x, Ax, Ax^2, \ldots, Ax^{m-1}\}. \tag{2.5}$$

Given an initial approximation x_0, in each iteration GMRES minimizes $\|Ax_k - b\|_2$ over all $x_k \in x_0 + \mathcal{K}_k(A, r_0)$ where $r_0 = b - Ax_0$. It is best to consider an orthonormal basis $\{v_1, v_2, \ldots, v_k\}$ of $\mathcal{K}_k(A, r_0)$ [7]. Then construct an orthogonal matrix $V_k \in \mathbb{R}^{n\times k}$ where each column j is a basis vector v_j. The kth iterate is a vector $x_k \in x_0 + \mathcal{K}_k(A, r_0)$. With our matrix V_k, this can be written as $x_k = x_0 + V_k y$ for some $y \in \mathbb{R}^k$. So, we have

$$\|b - Ax_k\|_2 = \|b - A(x_0 + V_k y)\|_2 = \|r_0 - AV_k y\|_2. \tag{2.6}$$

Thus, the minimization problem becomes

$$\min_{x \in x_0 + \mathcal{K}_k(A, r_0)} \|b - Ax\|_2 = \min_{y \in \mathbb{R}^k} \|r_0 - AV_k y\|_2. \tag{2.7}$$

This is a least-squares problem, but it can be simplified further by delving in to the structure of the algorithm.

First, it is worth noting the convergence properties of GMRES. For a non-singular $n \times n$ matrix, GMRES finds the exact solution in at most n steps [2]. While this is not particularly helpful in our problems, which are of dimension 10,000 and greater, it is still useful to know that the algorithm terminates after a finite number of steps.

Assume the matrix A is diagonalizable so $A = XDX^{-1}$ where the D is a diagonal matrix such that the entries on the diagonal, $\{\lambda_1, \ldots \lambda_n\} = \sigma$, are the eigenvalues of A. Also, let P_j be the space of polynomials with degree $\leq j$. Then, if we let

$$\varepsilon^{(m)} = \min_{p \in P_m,\, p(0)=1} \max_{\lambda_i \in \sigma} |p(A)|,$$

the following inequality is true for the residual after m steps, $\|r_m\|$:

$$\|r_m\| \leq \kappa(X)\varepsilon^{(m)}\|r_0\| \tag{2.8}$$

where $\kappa(X) = \|X\|\|X^{-1}\|$ is known as the condition number of X [2].

If A is normal ($AA^T = A^TA$), then $\kappa(X) = 1$. This means that for normal matrices, the convergence properties of GMRES depend only on the eigenvalues of A. If A is not normal, the pattern of the residuals also depends on the condition number of X, which can be arbitrarily large. Thus, we can not always expect the residual to decrease with each iteration. The worst case is a residual that stays constant for $n - 1$ iterations and then equals 0 after n iterations. For more on the convergence theory of GMRES, refer to [8].

2.3.1 Gram-Schmidt

The first implementation of GMRES by Saad and Schulz uses Arnoldi's method to generate the matrix V_k [8]. The simplicity of this implementation, based on the Gram-Schmidt process, has a trade-off in the form of reduced accuracy, as explored in [5]. Still, its simplicity more clearly demonstrates the mathematics behind the algorithm.

This leaves us with an orthonormal basis of $\mathcal{K}_k(A, v_1)$ and an upper Hessenberg matrix with structure

Algorithm 1 Arnoldi's Method

1: Suppose$\|v_1\|_2 = 1$
2: **for** $m = 1, 2, \ldots,$ **do**
3: $h_{im} = (Av_m)^T v_i$, $i = 1, 2, \ldots, m$
4: $\hat{v}_{m+1} = Av_m - \sum_{i=1}^{m} h_{im} v_i$
5: $h_{m+1,m} = \|v_{m+1}\|_2$
6: If $h_{m+1,m} = 0$, then stop; otherwise, let
7: $v_{m+1} = \hat{v}_{m+1}/h_{m+1,m}$

$$H_k = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,k} \\ h_{2,1} & h_{2,2} & \ldots & h_{2,k} \\ 0 & h_{3,2} & \ldots & h_{3,k} \\ 0 & 0 & \ldots & h_{4,k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & h_{k+1,k} \end{pmatrix}.$$

Furthermore, we have the identity

$$AV_k = V_{k+1} H_k. \tag{2.9}$$

In GMRES, we let $v_1 = \frac{r_0}{\|r_0\|_2}$ [8]. Since $r_0 = \beta v_1$ with $\beta = \|r_0\|_2$, the residual (2.7) becomes

$$\|r_0 - AV_k y\|_2 = \|r_0 - V_{k+1} H_k\|_2 = \|V_{k+1}(\beta e_1 - H_k y)\|_2. \tag{2.10}$$

Since V_{k+1} is orthogonal and the 2-norm is invariant under multiplication by an orthogonal matrix, we have

$$\min_{y \in \mathbb{R}^k} \|r_0 - AV_k y\|_2 = \min_{y \in \mathbb{R}^k} \|\beta e_1 - H_k y\|_2. \tag{2.11}$$

We can factor H_k as $Q_k R_k$ where $Q_k \in \mathbb{R}^{k+1 \times k+1}$ is orthogonal and $R_k \in \mathbb{R}^{k+1 \times k}$ is upper triangular with the form

$$R_k = \begin{pmatrix} r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 0 & \ddots & \ldots & r_{2,k} \\ 0 & 0 & \ddots & r_{k,k} \\ 0 & 0 & \ldots & 0 \end{pmatrix} = \begin{pmatrix} \tilde{R}_k \\ 0 \ldots 0 \end{pmatrix}.$$

Then, again using the fact that multiplying a vector by an orthogonal matrix does not change the norm, we have

$$\min_{y \in \mathbb{R}^k} \|\beta e_1 - H_k y\|_2 = \min_{y \in \mathbb{R}^k} \|Q_k(\beta Q_k^T e_1 - R_k y)\|_2 = \min_{y \in \mathbb{R}^k} \|\beta Q_k^T e_1 - R_k y\|_2.$$

We write $\beta Q_k^T e_1$ as $w_k = (\gamma_1, \ldots, \gamma_k, \gamma_{k+1}) = (\tilde{w}_k, \gamma_{k+1})$. Note that for any $y \in \mathbb{R}^k$

$$w_k - R_k y = \begin{pmatrix} \tilde{w}_k - \tilde{R}_k y \\ \gamma_{k+1} \end{pmatrix}. \tag{2.12}$$

So, the minimizer of $\|\beta w_k - R_k y\|$ is y such that $y \in \mathbb{R}^k$ solves $\tilde{R}_k y = \tilde{w}_k$. This gives us a residual equal to $|\gamma_{k+1}|$. This drastically simplifies the implementation of GMRES. To solve the least-squares problem, we just have to solve a triangular system. Furthermore, we can check the residual after each iteration by checking the last element of w_k.

To recap, the general implementation of GMRES is as follows [8]:

1. Generate the upper Hessenberg matrix H_k and an orthonormal basis $\{v_1, \ldots, v_k\} = V_k$ of $\mathcal{K}_k(A, r_0)$.
2. Decompose H_k into $Q_k R_k$
3. Check the last element of $w_k = \|r_0\|_2 Q_k^T e_1$ to determine the residual. If it is not acceptable, continue iterating.
4. Once we have an acceptable residual, determine the vector y that minimizes $\|\beta e_1 - H_k y\|$ by solving the upper triangular system $\tilde{R}_k y = \tilde{w}_k$ as defined above.
5. Let $x_k = x_0 + V_k y$.

There are ways to improve Arnoldi's method of generating an orthonormal basis, such as using the modified Gram-Schmidt method. Even with improvements, the Gram-Schmidt implementation tends to struggle when trying to orthogonalize vectors that are not sufficiently independent [9]. Instead, we will look at another implementation that generates an orthonormal basis through Householder transformations.

2.3.2 Householder

A Householder transformation P is defined as $P = I - uu^T$ where $\|u\|_2 = 1$. Note that P is an orthogonal matrix and that $Pv = v - uu^T v$, so we do not need to store P for matrix multiplications [9]. Furthermore, if we have two vectors x and y with the same norm, it is relatively straightforward to find the Householder transform such that $Px = y$ [9]. Algorithm 2 shows how we can generate an orthonormal basis for $\mathcal{K}_k(A, v_1)$ using Householder transformations.

For a proof that this is a basis of the Krylov subspace, refer to [9]. As before, we let $v_1 = \frac{r_0}{\|r_0\|}$. At iteration k of the algorithm, if $(v_1, Av_1, \ldots, Av_k)$ has rank k, we determine P_{k+1} as specified in Algorithm 2 and let

Algorithm 2 Generating an orthogonal basis of the Krylov subspace with Householder transformations

1: Suppose $\|v_1\|_2 = 1$.
2: Choose P_1 such that $P_1 v_1 = e_1$.
3: **for** $m = 1, 2, \ldots$ **do**
4: Set $v_m = P_1 \ldots P_m e_m$
5: **if** $(v_1, Av_1, \ldots, Av_m)$ is upper-triangular **then**
6: stop
7: **else**
8: choose P_{m+1} such that $P_{m+1} \ldots P_1(v_1, Av_1, \ldots, Av_m)$ is upper triangular

$$P_{k+1} \ldots P_1(Av_1, \ldots, Av_k) = H_k \tag{2.13}$$

where $H_k \in \mathbb{R}^{k+1 \times k}$ has the same structure from before. We also define $V_k = (v_1, \ldots, v_k)$ as before and the same analysis that led to needing to solve an upper-triangular system still applies [9]. The algorithm used in our simulations is based off the implementation of this procedure shown in Algorithm 3 below.

Algorithm 3 GMRES with Householder

1: Suppose an initial approximation x_0, a tolerance TOL, and an iteration limit MAXIT are given.
2: Compute $r_0 = b - Ax_0$, and determine P_1 such that $P_1 r_0 = \|r_0\|_2 e_1 \equiv w$.
3: **for** $m = 1, 2, \ldots,$ MAXIT **do**
4: Evaluate $v \equiv P_m \ldots P_1 A P_1 \ldots P_m e_m$
5: If $v^{(m+1)} = \ldots = v^{(n)} = 0$, proceed to step (8); otherwise, continue
6: Determine P_{m+1} with a Householder vector having first m components zero such that $P_{m+1}v$ is zero after the $(m+1)$st component.
7: Overwrite $v \leftarrow P_{m+1}v$
8: If $m > 1$, overwrite $v \leftarrow J_{m-1} \ldots J_1 v$
9: If $v^{m+1} = 0$, proceed to step (12); otherwise, continue
10: Determine J_m acting on components m and $m+1$ such that $(J_m v)^{(m+1)} = 0$
11: Overwrite $v \leftarrow J_m v$ and $w \leftarrow J_m w$.
12: Set

$$R_m = \begin{cases} (v) & \text{if } m = 1; \\ (R_{m-1}, v) & \text{if } m > 1; \end{cases}$$

13: If $|w^{m+1}| \leq$ TOL or $m =$ MAXIT, then solve for y_m and overwrite x_0 with x_m; otherwise, increment m.
14: Determine y_m which minimizes $\|w - R_m y\|_2$ by solving an $m \times m$ upper-triangular system with the first m rows of R_m as the coefficient matrix and the first m components of w as the right-hand side
15: **for** $k = 1, \ldots, m$ **do**
16: Overwrite $x_0 \leftarrow x_0 + y_m^{(k)} P_1 \ldots P_k e_k$.
17: If $|w^{(m+1)}| \leq$ TOL, accept x_0 as the solution. Otherwise, repeat to step (2).

2.4 Jacobian-Free Approximation

Any GMRES method requires repeated matrix multiplications. If possible, we want to avoid the daunting tasks of calculating and storing the Jacobian by instead having a function that approximates matrix multiplication. It helps to recall that multiplying the Jacobian by a vector is the equivalent of a directional derivative, which we can approximate by a finite difference [7]:

$$\frac{\partial F}{\partial x} q \approx \frac{F(u + \varepsilon q) - F(u)}{\varepsilon}. \tag{2.14}$$

A smaller ε will result in a closer approximation, but too small of an epsilon will cause round-off error [7]. If F is continuous, a small epsilon will mean that $F(u + \varepsilon q)$ and $F(u)$ will be extremely close in magnitude. Taking the difference of two very close numbers leads to a loss of precision in floating-point arithmetic, which then get amplified by the division by a small value.

Assume $F(x)$ is an exact function evaluation and $F(x) + \delta(x)$ is the perturbed value that is calculated in the code [1]. Assume we have an accurate enough way to calculate F so that $\delta(x) < \varepsilon_{\text{mach}}$ where $\varepsilon_{\text{mach}}$ is machine accuracy.

Note that, by using Taylor expansions, we see that

$$\begin{aligned} F(u + \varepsilon q) - F(u) &= F(u) + \varepsilon \frac{\partial F}{\partial x} q + \mathrm{O}(\varepsilon^2) - F(u) \\ &= \varepsilon \frac{\partial F}{\partial x} q + \mathrm{O}(\varepsilon^2). \end{aligned}$$

Then the difference between the $\frac{\partial F}{\partial x} q$ and the calculated first-order approximation is

$$\begin{aligned} \frac{\partial F}{\partial x} q - \frac{F(u + \varepsilon q) + \delta(x + \varepsilon q) - F(x) - \delta(x)}{\varepsilon} \\ = \mathrm{O}(\varepsilon) + \frac{\delta(u + \varepsilon q) - \delta(u)}{\varepsilon} \\ = \mathrm{O}(\varepsilon + \frac{\varepsilon_{\text{mach}}}{\varepsilon}). \end{aligned}$$

A good starting point for ε is one that roughly minimizes the residual. To minimize $\varepsilon + \frac{\varepsilon_{\text{mach}}}{\varepsilon}$, we let $\varepsilon = \sqrt{\varepsilon_{\text{mach}}}$. Dividing this by $\|q\|$ gives us the common first choice for ε in Jacobian-free Newton–Krylov (JFNK) methods,

$$\varepsilon = \frac{\sqrt{\varepsilon_{\text{mach}}}}{\|q\|}. \tag{2.15}$$

This choice of ε works well for systems governed by just one well-scaled PDE, known as a scalar PDE [7].

For our fluid problems, the vectors are normalized and thus well-scaled, but we are solving a system of coupled PDEs. For the remainder of this thesis, we will refer to (2.15) as Epsilon 1.

Other choices of ε are based on known methods of calculating the Jacobian. One choice takes inspiration from first approximating the Jacobian component-wise with a finite difference

$$J_{ij} = \frac{F_i(u + \varepsilon_j e_j) - F_i(u)}{\varepsilon_j} \tag{2.16}$$

where $\varepsilon_j = bu_j + b$ and b is a constant related to machine error ($b = 10^{-6}$ in our case of 64-bit double precision) [7]. By setting ε to the average of all ε_j's and scaling, we arrive at

$$\varepsilon = \frac{1}{n\|q\|_2} \sum_{i=1}^{n} b|u_i| + b \tag{2.17}$$

where n is the dimension of the system. This will be known as Epsilon 2.

Another approach to calculating the Jacobian proposed by Dennis and Schnabel is to use the same finite difference as (2.16) but let

$$\varepsilon_j = b \max\{|u_j|,\ \text{typ } u_j\}\text{sign}(u_j) \tag{2.18}$$

where typ x is the typical size of x. The maximum is taken to prevent ε_j from getting too small in cases where u_j is close to zero [6]. This idea was adapted by Brown and Saad for the directional derivative to give us what will be called Epsilon 3:

$$\varepsilon = \frac{b}{\|q\|_2} \max\{|u^T q|,\ \text{typ } u|q|\}\text{sign}(u^T q). \tag{2.19}$$

The finite difference approximation for the derivative we used is just of first-order accuracy. We can construct a second-order approximation to the first derivative by using a centered difference.

$$\frac{\partial F}{\partial x} q \approx \frac{F(u + \varepsilon q) - F(u - \varepsilon q)}{2\varepsilon} \tag{2.20}$$

It is not immediately obvious that this is a second-order approximation of $\frac{\partial F}{\partial x} q$. First, by taking the Taylor expansion of $F(u + \varepsilon q)$ and $F(u - \varepsilon q)$, we see that

$$\begin{aligned} F(u + \varepsilon q) - F(u - \varepsilon q) &= F(u) + \frac{\partial F}{\partial x}\varepsilon q + \frac{\partial^2 F}{\partial x^2}(\varepsilon q)^2 + O(\varepsilon^3) \\ &\quad - (F(u) - \frac{\partial F}{\partial x}\varepsilon q + \frac{\partial^2 F}{\partial x^2}(\varepsilon q)^2 - O(\varepsilon^3)) \\ &= 2\varepsilon \frac{\partial F}{\partial x} q + O(\varepsilon^3). \end{aligned}$$

Thus,

$$\begin{aligned}\frac{F(u+\varepsilon q)-F(u-\varepsilon q)}{2\varepsilon} &- \frac{\partial F}{\partial x}q \\ &= \frac{2\varepsilon\frac{\partial F}{\partial x}q + O(\varepsilon^3)}{2\varepsilon} - \frac{\partial F}{\partial x}q \\ &= O(\varepsilon^2).\end{aligned}$$

To decide which ε to use, we again assume we can evaluate $F(x)$ as $F(x)+\delta(x)$ where $\delta(x) \leq \varepsilon_{\text{mach}}$. Then

$$\begin{aligned}\frac{\partial F}{\partial x}q - &\frac{F(u+\varepsilon q)+\delta(x+\varepsilon q)-F(x-\varepsilon q)-\delta(x-\varepsilon q)}{2\varepsilon} \\ &= \mathrm{O}(\varepsilon^2) + \frac{\delta(u+\varepsilon q)-\delta(u-\varepsilon q)}{2\varepsilon} \\ &= \mathrm{O}(\varepsilon^2 + \frac{\varepsilon_{\text{mach}}}{\varepsilon^2}).\end{aligned}$$

The choice of ε that minimizes $\varepsilon^2 + \frac{\varepsilon_{\text{mach}}}{\varepsilon^2}$ is $\sqrt[3]{\frac{\varepsilon_{\text{mach}}}{2}}$ [1]. So, again after scaling down by $\|q\|$, our choice of ε for the centered difference is

$$\varepsilon = \frac{1}{\|q\|_2}\sqrt[3]{\frac{\varepsilon_{\text{mach}}}{2}}.$$

Despite the increased accuracy, such higher order approximations are not frequently used [7].

The JFNK method is the process of solving (2.2) with GMRES and using a finite difference approximation for the matrix multiplications required in the algorithm. Note that in our problem, the right hand side of our linear equation is $\mathbf{F}(\mathbf{u_k})$ so in the GMRES loop we already have access to this value needed in approximating the Jacobian. Each $n \times n$ matrix multiplication is replaced by one function evaluation (and some minor arithmetic). The centered difference requires two function evaluations for every matrix multiplication, though there are ways to limit the added cost [7].

2.5 Motivation

2.5.1 Preliminary Results

In a previous bachelors thesis, Dravins studied various implementations of the JFNK method on three different physical problems. While the sole focus of the work was not JFNK methods, it stumbled upon a puzzling error that put the validity of the solutions into question. Recall that in practice we check the magnitude of γ_{k+1} to insure we have an acceptable residual.

When testing though, we can also explicitly calculate the residual $\|Ax_k - b\|_2$ where we approximate Ax_k by the finite difference approximation of the Jacobian. These two values should be equal. In certain test cases where we can easily calculate the Jacobian, we can also evaluate Ax_k using the known Jacobian matrix. By comparing this to the residual where we used the approximation of the Jacobian, we can get a good idea of the effectiveness of the approximation.

The work compared these three residuals across six test cases, and the residuals matched only in one case. These tests were all done using the same approximation of the Jacobian, namely the one using Epsilon 1. In an effort to fix this issue, we will redo the tests using different approximations of the Jacobian.

2.5.2 Potential Causes

Before running any tests of our own, we consider some of the possible causes for the discrepancies between the residuals.

Recall that the conclusion that γ_{k+1} is equivalent to the residual came late in to the mathematical argument, after some key assumptions were made. Most importantly, the entire analysis depended on the set of vectors $\{v_1, \ldots, v_k\}$ being an orthonormal basis for the Krylov subspace. With computational errors present in each iteration of the algorithm, we can not assume that we have accurately generated an orthonormal basis for $\mathcal{K}_n(A, r_0)$. If the set of vectors is not orthonormal, we can not assume that γ_{k+1} is an accurate measure of the residual.

The other expected source of error would be the accuracy of the finite difference approximation of the Jacobian. There are two potential sources of error. The choice of epsilon might not have been optimal, either being too large to be an effective approximation or too small to avoid round-off error. We try to solve this by using Epsilon 2 and Epsilon 3. The other possibility is that the first-order approximation is not adequate for our problems. We test the centered difference to test for this issue.

2.5.3 Methods

We run the simulations using the 2DTAU code provided by the supervisor. The Tau code sets up all of the necessary parameters and then uses the TEMPO library written in C++ to carry out implicit time integration and the solving of the necessary linear and nonlinear systems. The code approximates solutions to the governing equations for fluids, the Navier-Stokes and Euler equations. The difference between the two is an extra second-order term in Navier-Stokes that comes from fluid viscosity and heat conduction. Refer to [3] for derivations of these equations and for more specifics on

their differences. The Navier-Stokes equations more accurately simulates the physical phenomenon to be discussed in this thesis, so they will be the primary focus of this work and will be used in all simulations unless stated otherwise.

The program begins by dividing the domain of the flow into a number of discrete cells. More cells are needed around areas with complex flows, like along boundaries. This discretization of the domain is known as a mesh. The code then approximates the flow using piecewise constant or piecewise linear functions across each individual cell. If we use piecewise constant functions for a first-order discretization, we are able to calculate the Jacobian of the matrix. We compare the three measures of the residual: γ_{k+1}, the residual checked by the algorithm, and $\|Ax_k - b\|_2$ calculated using the Jacobian-free (JF) approximation and the explicit matrix. For the piecewise linear or second-order discretization, we do not have the actual Jacobian available, so we can only compare γ_{k+1} and the residual calculated using the finite difference approximation of the Jacobian.

We consider six scenarios, two possible approximating functions for three physical situations. We then run Tau Code for just one Newton iteration. We vary the dimension of the Krylov subspace to compare the behavior as the number of iterations increases. The tests were run on a 64-bit OSX and the results were plotted in MATLAB. The code for the Householder implementation of GMRES was provided by the supervisor and the author wrote the code for the different finite difference approximations.

2.5.4 The Problems

The first problem we consider is the 2D flow around an airfoil known as NACA0012. With a dimension of 18,420, this is by far the smallest problem we consider.

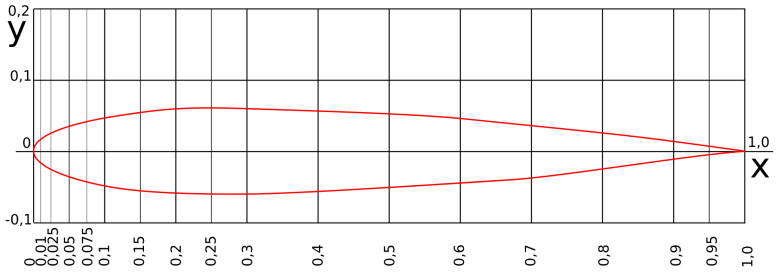


FIGURE 2.1: Shape of the NACA0012 airfoil [4].

The second problem is a wind turbine, which has a dimension of 97,380. In figure (2.2), we see the mesh used for the wind turbine.

The last problem we consider is that of the Flanschwelle, or flanged shaft. Figure (2.3) shows the mesh of system, with each picture zooming further in

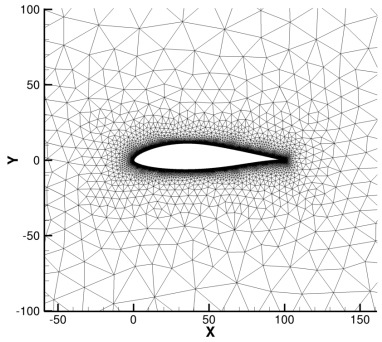


FIGURE 2.2: Wind Turbine Mesh

to the system. Note that even when the problem is zoomed in, the mesh is still incredibly dense, signalling that this is by far the most complex system we deal with. Its dimension is 568,208.

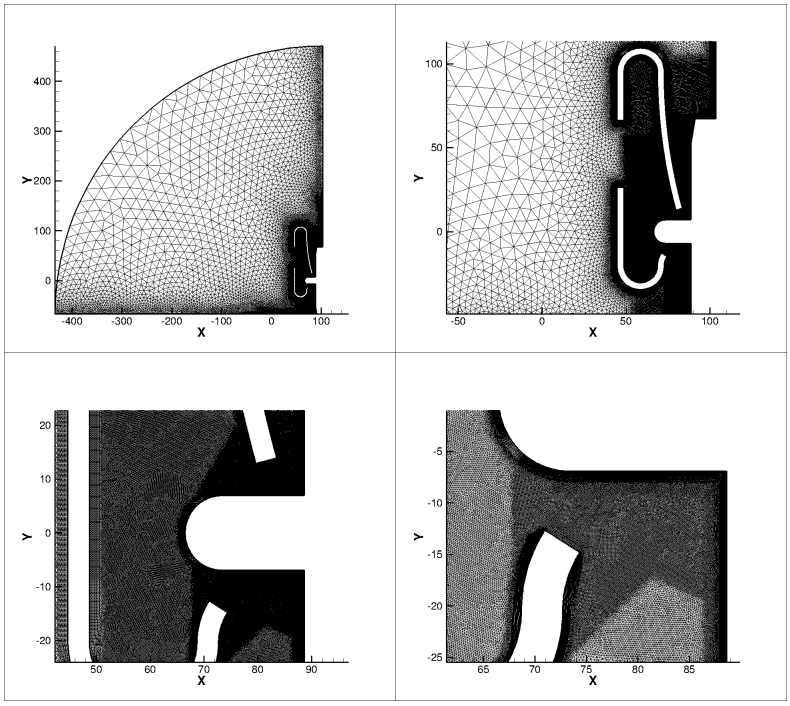


FIGURE 2.3: Flanschwelle Mesh

Chapter 3

Numerical Results

3.1 Piecewise Constant Discretization

We first consider the case where the flow is approximated by a piecewise constant function across each cell. For each scenario, we will have four plots. Each plot contains three measures of the residual $\|Ax - b\|_2$, labeled as follows:

- w-vector: the last element of the w-vector, $|\gamma_{k+1}|$; this is the residual checked by the GMRES algorithm
- JF approx: the norm $\|Ax - b\|_2$ where Ax is calculated using the finite difference approximation
- Explicit Matrix: the norm $\|Ax - b\|_2$ where Ax is calculated explicitly using the Jacobian matrix

Recall that theoretically, all three of these norms should be identical. The plots will compare these norms when using the three versions of epsilon and the centered difference.

3.1.1 NACA0012

We first run the tests on the NACA0012 problem.

See figure (3.1) for the results. In all plots, the w-vector and JF residuals are the same for every iteration. This is promising as it agrees with the theory. The last element of the w-vector equals the residual under the assumption that we have generated a fully orthonormal basis. Without the explicit matrix residual, it would seem that we have succeeded.

The presence of the explicit matrix multiplication complicates things, however. It diverges from the other two residuals at the first iteration and flattens out while the other two residuals decrease. To see if we can solve this issue, we try the other choices for epsilon.

If the finite difference approximation is not accurate enough, the Jacobian-free method could be solving $Ax = b$ for a matrix A that is slightly different than the Jacobian. The goal of the different epsilons was to fix this potential

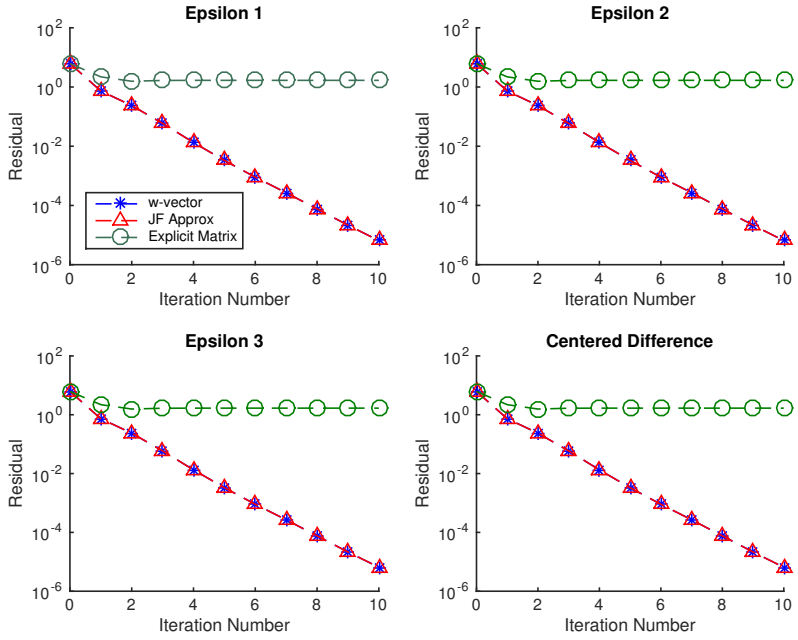


FIGURE 3.1: Residuals for the first-order NACA0012. We see no change regardless of how we approximate the Jacobian

issue, but the problem persists. Another possibility is that there is a bug in the calculation of the Jacobian.

We would have at least expected the centered difference approximation to show an improvement, since it is by construction a better approximation of the Jacobian. The fact that the centered difference agrees with the first-order approximation and shows no improvement in the explicit matrix residual implies that the forward difference was accurate enough already. It is necessary to look at more cases to reinforce this theory, however.

3.1.2 Wind Turbines

Typically with the wind turbines, we use a 2-step SDIRK scheme to accurately solve the differential equations. For testing purposes however, we used the same time integration scheme as the one we used for NACA0012, the implicit Euler method. The error will be noticeably higher for the wind turbines, but we are interested in the general trends of the three measures of residual, not so much the value of the residual itself.

The results for the wind turbines seem more encouraging at first. For the first ten iterations, all three calculations of the residuals line up as they should. In practice though, we will usually use Krylov subspaces of dimension 40 or higher. As the number of iterations increases, the simulation of the wind turbines begins to follow the familiar pattern of NACA0012. The w-vector and JF approximation match but the explicit matrix multiplication flattens out.

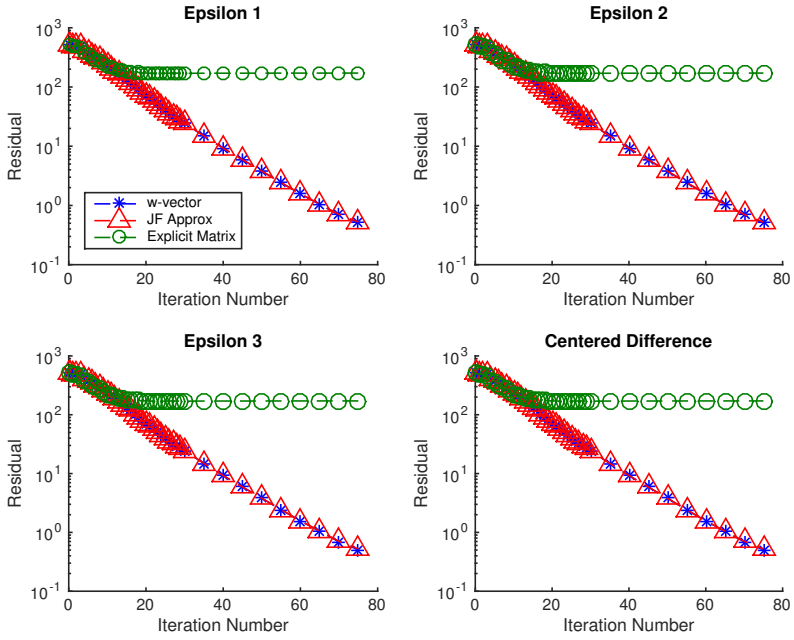


FIGURE 3.2: Residuals for the first-order wind turbine. Again, changing how we approximate the Jacobian seems to have little affect

Neither epsilon seems to have much of an effect, like before. The same also holds true for the centered difference approximation.

This problem is basically the same as the last one. While the w-vector and JF residuals match as they should, we still cannot comment on the effectiveness of the routine as neither agree with the residual when we use the actual matrix. Again, this is assuming that there are no bugs in the implementation of the actual matrix, which we now suspect could be part of the problem.

3.1.3 Flanschwelle

Finally we test the Flanschwelle. The results for Epsilon 1 and Epsilon 2 are in figure (3.3)

Note that the w-vector and JF approximation do not coincide for more than one iteration. The explicit matrix residual behaves as it did in previous tests, diverging after one iteration and leveling off at a certain value. Now the JF error also follows this pattern, except that it levels off at an even worse error than the explicit matrix. None of the three mathematically equivalent measures of residual agree.

We then try Epsilon 3—see figure (3.4). While the w-vector and explicit matrix residuals do not show much change, there is noticeable improvement in the JF approximation. It coincides with the w-vector for two iterations rather than one and flattens out well below the explicit matrix residual. Still

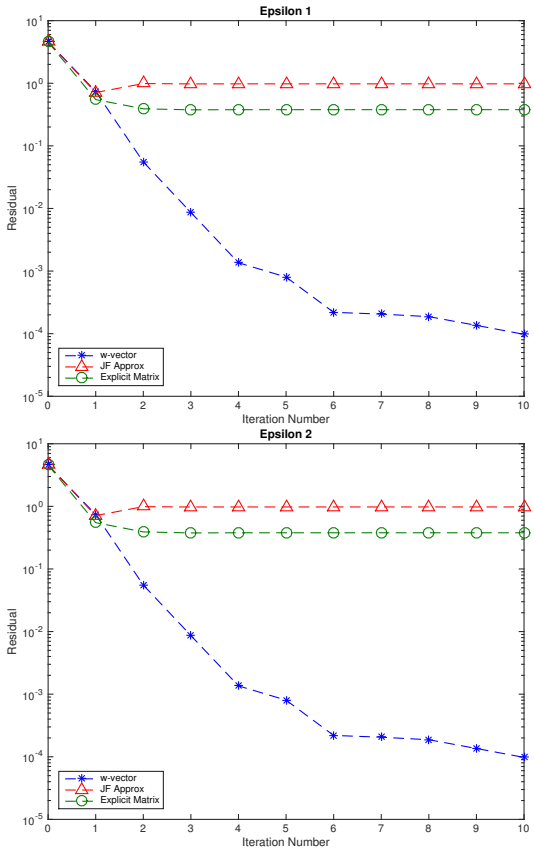


FIGURE 3.3: Residuals for the first-order Flanschwelle are nearly identical for Epsilon 1 and 2

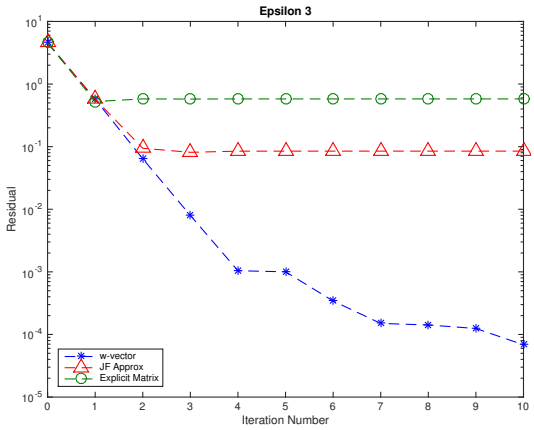


FIGURE 3.4: We see slight improvement in the switch to Epsilon 3 for first-order Flanschwelle

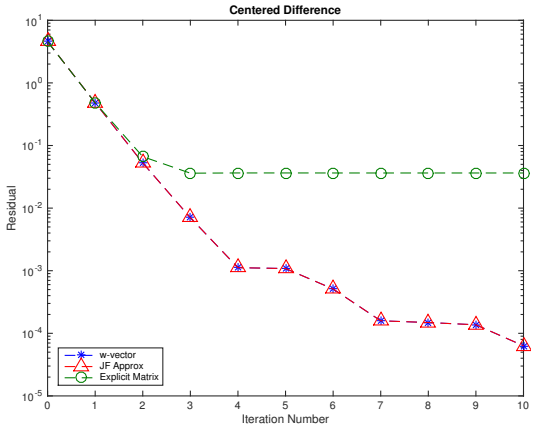


FIGURE 3.5: Promising results for the centered difference in first-order Flanschwelle

though, considering that we will typically use up to 40 iterations, this is not acceptable for our simulations.

The centered difference, shown in figure (3.5), displays massive improvement. Note that now the w-vector and JF approximation are the same for all ten iterations. The explicit matrix residual also agrees with the other two vectors for two iterations instead of one, but still levels off immediately afterwards. This is the first evidence we have of an improved approximation of the Jacobian leading to significantly improved results; however, it fixes a disparity that was not present in the previous simulations and the fact that the explicit matrix still does not agree with the other two residuals is troubling. Recall that the Flanschwelle is a significantly larger problem than the NACA0012 and wind turbine. It appears that the first-order approximation was not accurate enough and that we need a higher-order method to make accurate calculations for problems of this size. It is also interesting to note that the higher-order method improved the explicit matrix residual in this case but not the others.

3.2 Piecewise Linear Discretization

Now each problem is discretized as a piecewise linear function. We do not have access to explicit matrix multiplication for this discretization, but we can still compare the w-vector and JF residuals, which still should be equivalent.

3.2.1 NACA0012

For the default choice of epsilon, the residuals only match for one iteration. For a Krylov subspace of dimension greater than one, the JF error flattens out much like the explicit matrix residual did for previous problems. As

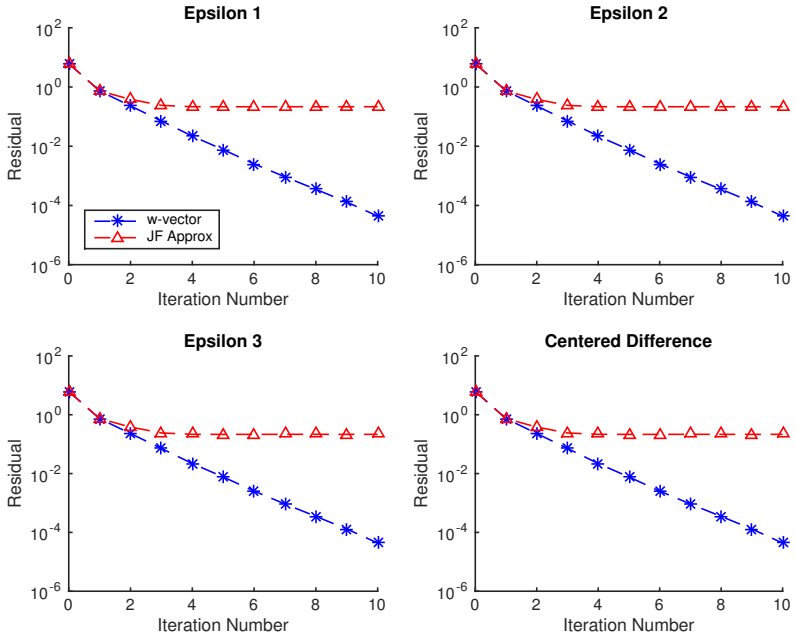


FIGURE 3.6: Second-order NACA0012

before, we try using different finite difference approximations to get the two residuals to match. Unfortunately, the JF error shows no signs of change regardless of how we approximate the Jacobian.

Even the centered difference shows no signs of improvement. Again, the centered difference is designed to have improved accuracy for all problems. Perhaps an even higher-order method is needed to show significant improvement in the JF error. It could also mean that accuracy is not the issue, but rather that the method fails to capture some subtle but important physical or numerical aspect of the problem. We might also suspect that there is an issue with the piecewise linear discretization, but this suspicion goes away when we look at the results for the wind turbines.

3.2.2 Wind Turbines

Even with the simplest choice of epsilon, the results for the wind turbines are mathematically sound. The given residual follows the actual residual for up to 75 iterations. While it is possible that they diverge later, we will not typically use this large of a Krylov subspace. So, for all intents and purposes, the errors match for Epsilon 1, Epsilon 2, and the centered difference.

Epsilon 3, however, exhibits strange behavior. The two residuals roughly coincide for 30 iterations and even by 40 iterations they are still fairly close. As we increase the dimension of the Krylov subspace, we see a slight divergence in the two residuals. In application, it does not seem to be a huge issue since the actual difference between the residuals is not very large and

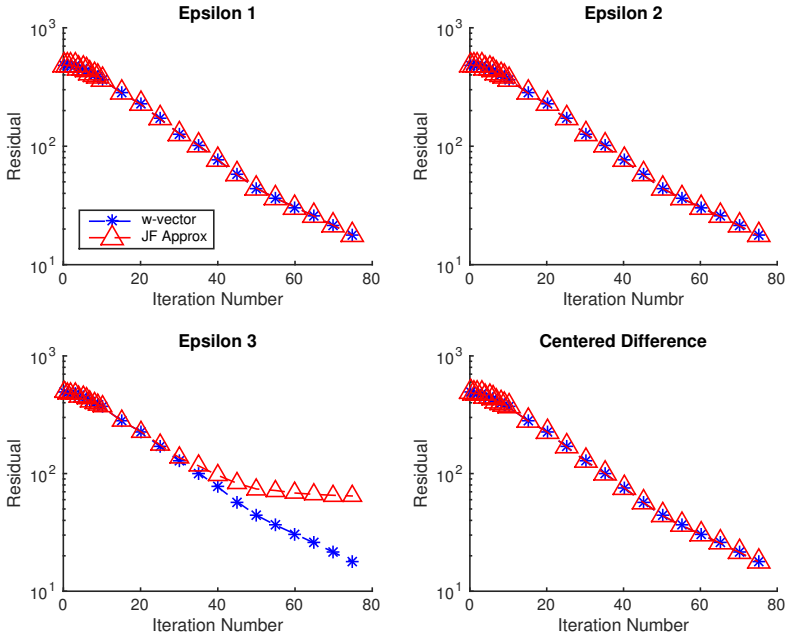


FIGURE 3.7: Second-order wind turbine shows that there isn't an inherent problem with the piecewise linear discretization

since the other approximations of the Jacobian work as expected, we could use them instead. We will return to this strange result in Chapter 4.

Overall, these tests show us that there is not an inherent issue with the second-order discretization. Also recall that this problem is significantly larger than the NACA0012, so we would expect worse results if the finite difference was not a good enough approximation of the Jacobian.

3.2.3 Flanschwelle

The results for the Flanschwelle are more similar to NACA0012. After one iteration, the JF approximation levels off while the w-vector residual decreases significantly. No effect is seen by switching to Epsilon 2.

There is some improvement when using Epsilon 3. The residuals now match for two iterations before diverging. This is reminiscent of the first-order discretization of the Flanschwelle, where Epsilon 3 showed slight improvement and a centered difference fixed the disparity between the two residuals all together. Unfortunately, we do not see the same success with the centered difference now.

While the JF approximation levels off at a slightly lower error with the centered difference compared to Epsilon 3, it still diverges after the second iteration. This might suggest that accuracy is an issue when dealing with the Flanschwelle, regardless of the discretization. Either the centered difference does not provide a good enough increase in accuracy, or there are other

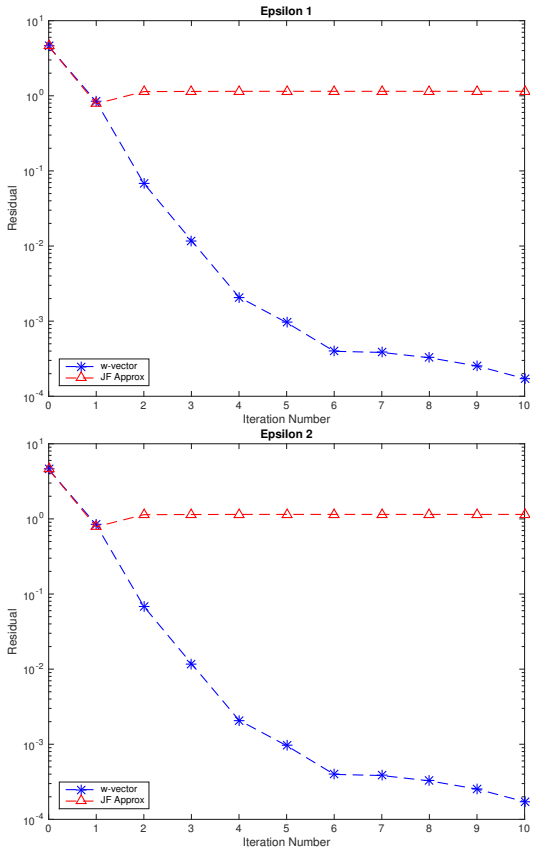


FIGURE 3.8: As in the first-order Flanschwelle, changing from Epsilon 1 to Epsilon 2 does not have a noticeable affect

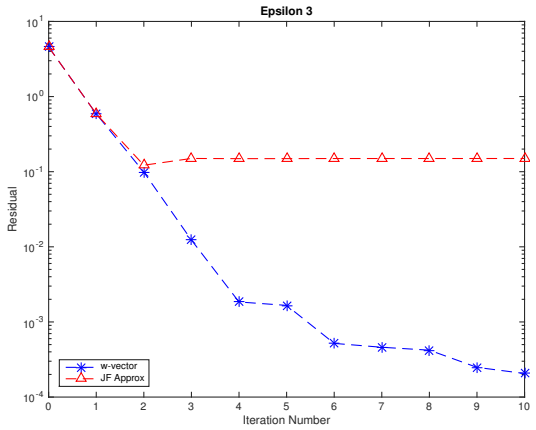


FIGURE 3.9: Epsilon 3 improves the second-order Flanschwelle

underlying issues with the problem that cannot be fixed by simply using higher-order methods. We might suspect that the same issues with the second-order NACA0012 are present in this problem, but we have made little progress in determining exactly what those issues are.

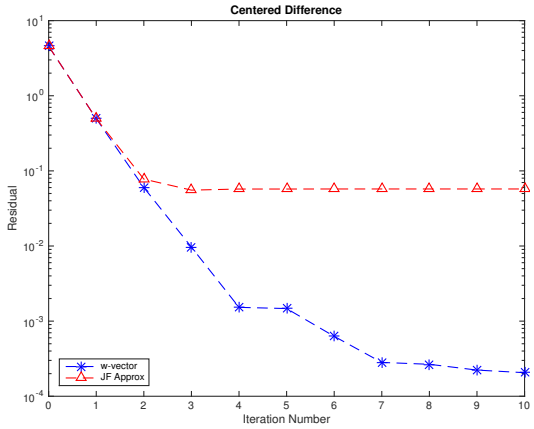


FIGURE 3.10: The centered difference shows the most consistent results in both Flanschwelle discretizations

3.3 Testing for Specific Sources of Error

With time left over after the main tests were run, we tried out various tests to help identify the problems in our simulations.

3.3.1 A Simpler Problem

We first look into the issue of the explicit matrix residual flattening out. In all of the models so far, the Navier-Stokes equations were the governing formulas behind the simulations. By testing the models using the simpler Euler equations, we can see if the additional higher-order terms present in the Navier-Stokes equations cause the strange behavior seen in the explicit matrix residual. If this is the case, we might conclude that the code struggles to calculate the Jacobian due to the more complicated terms in the NS equations. We run the same tests as before on the piecewise constant NACA0012 problem, except using the Euler equations to describe the flow rather than Navier-Stokes. For the differences in the simulating the two equations, refer to [3].

For the NS equations, recall that the explicit matrix residual split from the other two residuals at the first iteration, whereas now they coincide for two iterations. This is a slight improvement, but the explicit matrix residual still flattens out while the others decay together. This is reminiscent of when we changed from Epsilon 1 to the centered difference for the first-order Flanschwelle problem. Although the residual behaves more like it

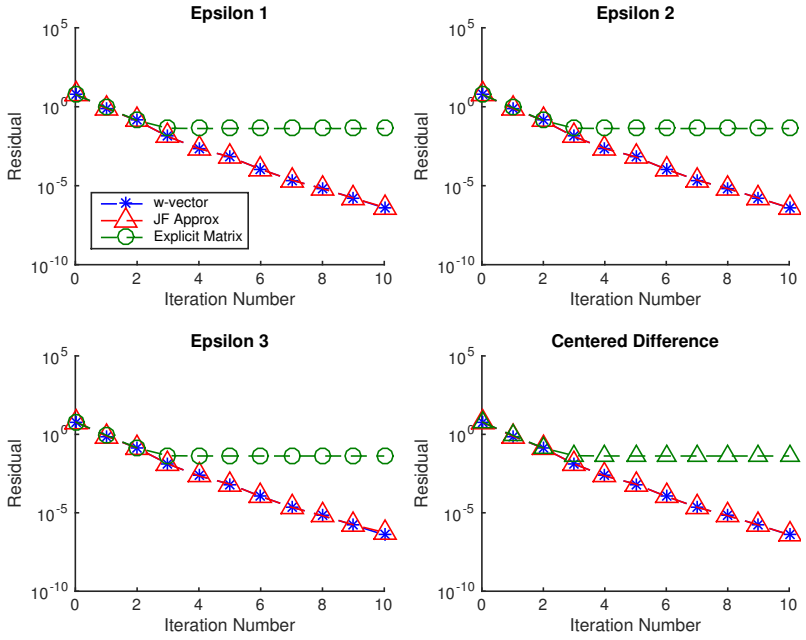
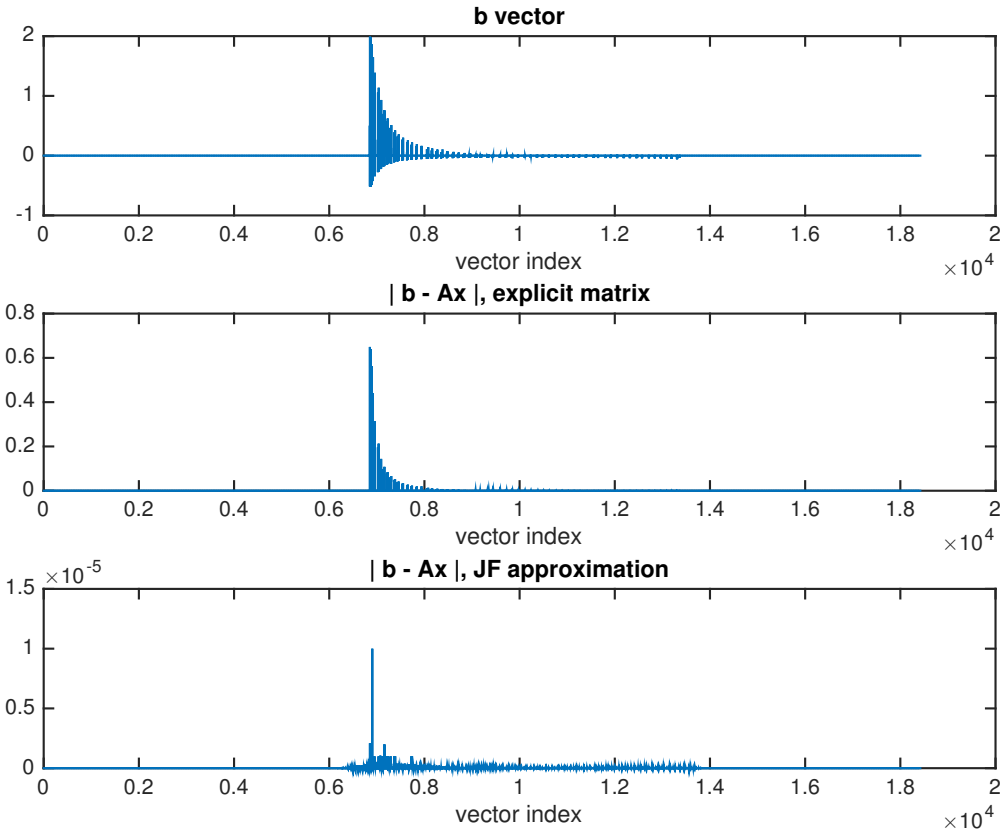


FIGURE 3.11: Piecewise constant NACA0012 governed by the simpler Euler Equations

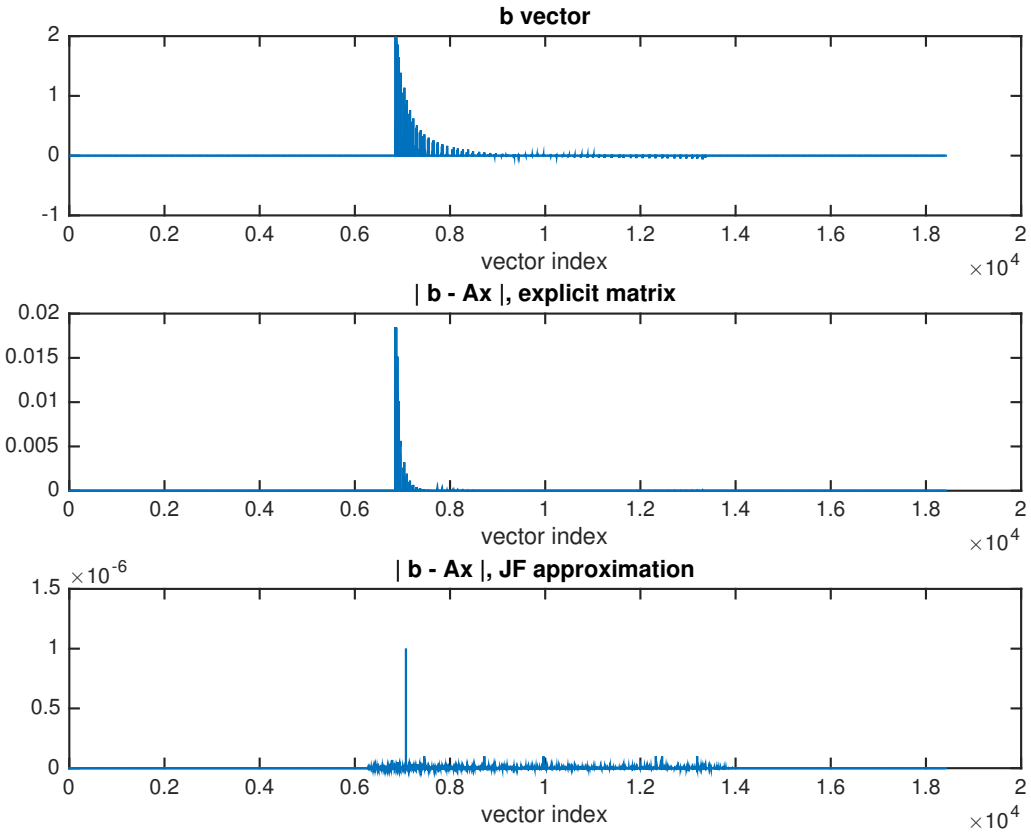
should, the improvement is not significant enough to make any sweeping conclusions. In general it seems that the explicit matrix residual behaves more like we would expect for "easier" problems, but even then, something is not right. If there are problems with the explicit calculation of the Jacobian, we know that the problems persist regardless of the added terms in the NS equations.

3.3.2 A Closer Look

To get a closer look at the residuals in the piecewise constant NACA0012, we examined the three vectors involved in their computation: Ax calculated with the actual Jacobian matrix, Ax calculated with the finite difference approximation, and the right hand side of the equation, b. Below are three plots. The first is a plot of the vector b component-by-component to get an idea of the numerical structure of the solution. The second is the component-wise difference between the explicitly calculated Ax and the right hand side b. The third is the same thing except with Ax approximated by a finite difference:



Note that the explicit error spikes right when b gets more oscillatory, while the finite difference residual does not; with the exception of one spike in the middle, the error is fairly consistent from component to component. It seems that somewhere along the algorithm, whether it is in the GMRES loop or in the final multiplication by A, the explicit matrix fails to capture the behavior of the function in a massive way at a certain location. This notion is reinforced when we look at the same vectors using the Euler equations rather than Navier-Stokes:



We see another jump in the error of the explicit matrix residual. Furthermore, it occurs at the exact same location as the Navier-Stokes simulation (index 6854 of the vector). Whether it is a certain property of the physics like a boundary or just the numerical behavior of the system, it is clear that the explicit matrix struggles significantly at this one location and right after, regardless of the complexity of the equations involved. Since the explicit matrix is supposed to be an exact representation of the Jacobian, we should expect the errors to be minimal everywhere. The fact that there is such a significant spike at one location makes it worthwhile to check the generation of the Jacobian for bugs. The relatively consistent behavior of the JF error gives confidence that the approximation is working as intended and that it may not be the source of these issues.

3.3.3 Non-orthogonal Bases

As mentioned in the introduction, the assumption that $w[k+1] = \|Ax - b\|_2$ hinges on the fact that we successfully generated an orthonormal basis $\{v_1, \ldots, v_k\}$ of the Krylov subspace. We examine the most revealing cases. We would expect the effects of a non-orthonormal basis to be amplified as the number of basis vectors increases, so in all cases we look at tests using the 10-dimensional Krylov subspace.

First, we look at the basis vectors $\{v_1, \ldots, v_{10}\}$ of the 10-dimensional Krylov subspace in the piecewise constant and piecewise linear NACA0012 problem. Recall that the JF approximation and w-vector differed in the piecewise linear discretization, but not in the piecewise constant one. Thus, if we found that the basis was orthonormal in the first-order discretization but not the second-order one, a lack of orthogonality could be contributing to the discrepancies in residual. Unfortunately, tests show that the basis vectors in both cases are orthonormal. See tables A.1 and A.2.

Another test worth looking at is the piecewise 1st order Flanschwelle problem. Recall that while initially the w-vector and JF approximation differed, a change to the centered difference resulted in the two vectors matching for all iterations. While this appears to be the work of the increased accuracy that comes with the higher-order method, it could also have to do with the basis being more orthogonal when using the higher-order method. Tests reveal that this is not the case, as the basis for the piecewise constant Flanschwelle using Epsilon 1 is already orthonormal. See table A.3.

Chapter 4

Conclusion

4.1 Analysis

The immediate conclusion is that there are a number of issues going on here that likely cannot be fixed simply by improving the approximation of the Jacobian.

We are at least able to eliminate two possible sources of the inconsistencies. Since the tests with the Euler equations did not fix the issues, we can assume that the main source of difficulty is not the second-order terms present in the Navier-Stokes equations. Furthermore, the algorithm does not seem to have an issue generating orthonormal bases, which would have taken away the relationship between the JF and w-vector residuals.

No matter what we did, we were not able to get the residual $\|Ax - b\|$ where Ax is calculated explicitly to agree with the two other mathematically equivalent measures of residual. For the NACA0012 and wind turbines, this residual showed no change regardless of which approximation we used. Recall that explicit residual actually improved with a higher-order scheme in the case of the first-order Flanschwelle. This could mean that the explicit matrix residual is actually connected with the effectiveness of the Jacobian approximation, but this would not explain why we saw no improvement in the other cases. As mentioned earlier, it seems likely that accuracy was only an issue in the Flanschwelle simulations. Part of the behavior of the explicit matrix was due to the approximation errors that propagated through each iteration in the Flanschwelle, but once we dealt with this we were still left with the errors present in the other simulations.

Another important result from the piecewise constant Flanschwelle tests was that it shows that we can sometimes fix a disparity between the given and approximated residuals by using a higher-order method. Overall, the centered difference showed the most promise. It fixed the aforementioned disparity in the piecewise constant Flanschwelle and it led to a very minor improvement of the JF residual in the second-order Flanschwelle. In most cases though, the centered difference is not worth the added function evaluations. While it is a valuable tool to solve very specific test cases, it should not become the standard for this algorithm.

What can we make of the Epsilon 3 test for the piecewise linear wind turbine? First, it reminds us that although most results were not affected by a change in ε, the residuals still can be sensitive to this sort of change. Recall that Epsilon 3 is the only ε that has a dependence on the vector q where q is the vector being multiplied by the Jacobian. This dependence is most likely the source of the error. This reinforces the notion that the right choice of ε is a difficult one—a good choice for one problem can cause issues in another. In practice, if one choice for ε gives strange results, it is worth checking other approximations of the Jacobian to solve the problem; however, the number of different issues in this thesis implies that there are numerous subtle issues that will not be fixed by finding the optimal ε.

4.2 Topics for Further Study

Due to the unexpected behavior of the explicit matrix residual and the severity of the errors shown in Section 3.3.2, we would first suggest making sure that the generation of the Jacobian is working as intended.

There are still different ways to approximate multiplication by the Jacobian. One could for example, test a third-order finite difference approximation. Higher order methods are not practical for these problems, but the use of them might help reveal what is going wrong in the algorithm.

More likely to be of use is preconditioning. Though tests were done without preconditioning, we would typically use it for problems of this type. This would be easy to accomplish, as all one would need to do would be to change the preconditioning value from 0 to 1 in the test files. Using an effective preconditioner in JFNK methods will typically lead to us using a simpler form of the Jacobian, which will be easier to approximate. For a reference on the various types of preconditioning we can use and their purpose, see [7]. We can also try restarted GMRES methods, where after n iterations of GMRES we restart the algorithm with v_1 being the normalized residual $Ax_n - b$. Though most of the convergence theory does not apply to restarted GMRES, it has been shown to be effective in practice and may be what is needed here [2].

Another alternative is to investigate the problems more closely. If preconditioning or restarting show no improvement, it would be worthwhile to investigate the specifics of the problems to see if there are any common physical or numerical properties. For example, is there something that sets the mathematically-consistent piecewise linear wind turbines from the other more puzzling second-order problems ? One could also look at the location 6854 in the piecewise constant NACA0012 to see where it lies on the airfoil. In practice, it is not realistic to check individual simulations for unique quirks and fine tune what is supposed to be a general purpose fluid solver,

but this type of investigation could reveal the causes behind these strange inconsistencies.

Appendix A

Appendix

The first three sections consist of the files that were used to run the simulations. Any parameters that were changed during the tests are numbered and explained after the end of the third section.

A.1 NACA0012 Input Parameter

```
restart file =
grid name = NACA_LOW_MACH60.tri

Euler/NS = NAVIER_STOKES (1)
Problem type = STEADY
Explicit/Implicit = IMPLICIT
Time integration = 0

Error controlled time adaptivity = 0
fsafety = 0.9
Minimal time step decrease = 0.5
Maximal time step increase = 2.0
Absolute tolerance in time step control = 1e-3
Relative tolerance in time step control = 1e-3

reconstruction_type = TVD
reconstruction = PIECEWISE_CONSTANT (2)
limiter = BARTH_JESPERSEN

flux_function = AUSMDV
flux_function_jacobi = AUSMDV

ausmpup_ref_mach = 0.3
ausmpup_a_half =
ausmpup_kpfa_fix =
```

```
CFL–number = 4.0
max timesteps = 1
max time  = 999e10
CFL max number = 100.0
CFL update period = 10
CFL update factor = 1.2

Max Newton steps = 1
Newtontolerance = 1e–4
Eisenstat–Walker forcing terms = 0
Extrapolation = 0

krylov dim = 10 (3)
ilu update period = 30
epsilon = 1e–10
max restarts = 0

output period  = 2000
output prefix = NACA_LOW_MACH60

adaption period = 50000
adapt refine limit = 1.0e–1
adapt coarse limit = 1.0e–3
least triangle area = 0

Mach number = 0.85
angle = 1.25

preconditioner = 0
preconupdates = 0
physical renumbering = 1
matrix free solver = 1


Reynolds number = 1000
Prandtl number = 1.2
temperature = 300
isotherm or adiabat = ADIABAT
temperature on wall = 273
```

A.2 Wind Turbines Input Parameters

```
restart file = wind_96_boundarylayer_steadyEuler_t200.pval
grid name = wind_96_boundarylayer.tri

Euler/NS = NAVIER_STOKES
Problem type = STEADY
Explicit/Implicit = IMPLICIT
Time integration = 0

Error controlled time adaptivity = 0
Constant Delta t = 0.01
Timestep controller = 0
theta = 0.9
Minimal time step decrease = 0.5
Maximal time step increase = 2.0
Absolute tolerance in time step control = 1e-1
Relative tolerance in time step control = 1e-1

reconstruction_type = TVD
reconstruction = PIECEWISE_CONSTANT (2)
limiter = BARTH_JESPERSEN

flux_function = AUSMDV
flux_function_jacobi = AUSMDV

ausmpup_ref_mach = 0.3
ausmpup_a_half =
ausmpup_kpfa_fix =

CFL-number = 30
max timesteps = 1
max time  = 202
CFL max number = 30.0
CFL update period = 10
CFL update factor = 1.2

Max Newton steps = 1
Newtontolerance = 1e-13
Eisenstat-Walker forcing terms = 0
Extrapolation = 0

krylov dim = 10 (3)
ilu update period = 30
```

```
epsilon = 1e-3
max restarts = 0

output period  = 200000
output prefix = Wind_96_SDIRK2

adaption period = 50000
adapt refine limit = 1.0e-1
adapt coarse limit = 1.0e-3
least triangle area = 0

Mach number = 0.12
angle = 40.0

preconditioner = 0
preconupdates = 0
physical renumbering = 1
matrix free solver = 1


Reynolds number = 1000
Prandtl number = 1.2
temperature = 300
isotherm or adiabat = ADIABAT
temperature on wall = 273
verbose = 0
```

A.3 Flanschwelle Input Parameters

```
restart file =
grid name = flanschmit2duesen.tri

Euler/NS = NAVIER_STOKES
Problem type = FLANSCHWELLE
Explicit/Implicit = IMPLICIT
Time integration = 0

Error controlled time adaptivity = 0
Constant Delta t = 0.01
Timestep controller = 3
theta = 0.9
```

```
Minimal time step decrease = 0.5
Maximal time step increase = 2.0
Absolute tolerance in time step control = 1e-3
Relative tolerance in time step control = 1e-3

reconstruction_type = TVD
reconstruction = PIECEWISE_CONSTANT (2)
limiter = BARTH_JESPERSEN

flux_function = AUSMDV
flux_function_jacobi = AUSMDV

ausmpup_ref_mach = 0.3
ausmpup_a_half =
ausmpup_kpfa_fix =

CFL-number = 2.0
max timesteps = 1
max time  = 10
CFL max number = 0.9
CFL update period = 10
CFL update factor = 1.2

Max Newton steps = 1
Newtontolerance = 1e-4
Eisenstat-Walker forcing terms = 0
Extrapolation = 0

krylov dim = 10 (3)
ilu update period = 30
epsilon = 1e-20
max restarts = 0

output period  = 200000000
output prefix = flanschwelle_2

adaption period = 5000000000
adapt refine limit = 1.0e-1
adapt coarse limit = 1.0e-3
least triangle area = 0

Mach number = 1.0
```

```
angle = 1.25

preconditioner = 0
preconupdates = 0
physical renumbering = 1
matrix free solver = 1


Reynolds number = 1000
Prandtl number = 1.2
temperature = 300
isotherm or adiabat = ADIABAT
temperature on wall = 273
```

(1) is set to EULER only for NACA0012 for the tests done in section 2.3.1

(2) is set to PIECEWISE_CONSTANT when testing first order discretizations and PIECEWISE_LINEAR when testing second order ones

(3) is set varied from 0 to the maximum number of dimensions tested, which is 10 in most cases

A.4 Orthogonality of Bases

1	2	3	4	5	6	7	8	9	10	
1	1.0000e+00	2.1680e-08	-2.3063e-08	2.3652e-08	2.4144e-07	-2.5049e-07	8.3071e-08	1.2780e-07	-1.0309e-07	-1.1557e-07
2	2.1680e-08	1.0000e+00	3.5270e-08	-8.5358e-08	4.1718e-08	-1.4853e-07	2.3027e-07	1.7191e-07	1.2798e-07	-8.5067e-08
3	-2.3063e-08	3.5270e-08	1.0000e+00	2.2938e-08	2.9011e-07	-3.0442e-08	8.3476e-08	-2.0075e-07	-3.4977e-07	3.0647e-07
4	2.3652e-08	-8.5358e-08	2.2938e-08	1.0000e+00	-5.1457e-07	5.6754e-08	1.6083e-07	-1.4824e-07	4.9057e-07	-2.3046e-07
5	2.4144e-07	4.1718e-08	2.9011e-07	-5.1457e-07	1.0000e+00	-3.3161e-08	-1.2951e-07	1.7953e-07	-3.1144e-07	-1.7367e-07
6	-2.5049e-07	-1.4853e-07	-3.0442e-08	5.6754e-08	-3.3161e-08	1.0000e+00	3.3448e-07	-2.0837e-07	-1.3112e-07	6.5371e-09
7	8.3071e-08	2.3027e-07	8.3476e-08	1.6083e-07	-1.2951e-07	3.3448e-07	1.0000e+00	4.7469e-08	4.6749e-08	6.8905e-08
8	1.2780e-07	1.7191e-07	-2.0075e-07	-1.4824e-07	1.7953e-07	-2.0837e-07	4.7469e-08	1.0000e+00	-1.6538e-07	1.4490e-07
9	-1.0309e-07	1.2798e-07	-3.4977e-07	4.9057e-07	-3.1144e-07	-1.3112e-07	4.6749e-08	-1.6538e-07	1.0000e+00	1.6079e-07
10	-1.1557e-07	-8.5067e-08	3.0647e-07	-2.3046e-07	-1.7367e-07	6.5371e-09	6.8905e-08	1.4490e-07	1.6079e-07	1.0000e+00

TABLE A.1: Orthonormality of V_{10} for first order NACA0012. Item (i, j) in the table is $v_i^T v_j$

1	2	3	4	5	6	7	8	9	10	
1	1.0000e+00	-1.3410e-07	-3.2648e-08	1.1910e-08	-4.2924e-08	-2.6792e-07	-3.4571e-07	-1.0909e-07	2.1074e-07	3.3375e-07
2	-1.3410e-07	1.0000e+00	-5.6011e-07	-2.9011e-07	-2.9821e-07	1.2678e-07	1.0364e-07	1.2877e-07	2.5637e-08	-2.9360e-08
3	-3.2648e-08	-5.6011e-07	1.0000e+00	1.7704e-07	-1.6651e-07	-5.2372e-08	-2.1134e-07	2.1301e-07	-1.6449e-07	8.7571e-09
4	1.1910e-08	-2.9011e-07	1.7704e-07	1.0000e+00	-3.1588e-07	9.4206e-08	-2.3916e-07	-2.3605e-07	2.6102e-07	1.6471e-08
5	-4.2924e-08	-2.9821e-07	-1.6651e-07	-3.1588e-07	1.0000e+00	-1.4346e-07	5.4835e-07	2.8032e-07	-4.9726e-07	-2.9783e-07
6	-2.6792e-07	1.2678e-07	-5.2372e-08	9.4206e-08	-1.4346e-07	1.0000e+00	4.3312e-07	-3.5627e-07	1.2104e-07	-8.9076e-08
7	-3.4571e-07	1.0364e-07	-2.1134e-07	-2.3916e-07	5.4835e-07	4.3312e-07	1.0000e+00	-7.0502e-08	-2.1955e-07	-3.1559e-07
8	-1.0909e-07	1.2877e-07	2.1301e-07	-2.3605e-07	2.8032e-07	-3.5627e-07	-7.0502e-08	1.0000e+00	5.4975e-08	2.1078e-08
9	2.1074e-07	2.5637e-08	-1.6449e-07	2.6102e-07	-4.9726e-07	1.2104e-07	-2.1955e-07	5.4975e-08	1.0000e+00	2.7601e-07
10	3.3375e-07	-2.9360e-08	8.7571e-09	1.6471e-08	-2.9783e-07	-8.9076e-08	-3.1559e-07	2.1078e-08	2.7601e-07	1.0000e+00

TABLE A.2: Orthonormality of V_{10} for second order NACA0012. Item (i, j) in the table is $v_i^T v_j$

1	2	3	4	5	6	7	8	9	10	
1	1.0000e+00	5.6712e-08	-1.8363e-07	-1.2919e-08	-1.6957e-07	-6.2559e-09	2.1097e-08	1.5197e-08	-2.6122e-09	-2.3824e-08
2	5.6712e-08	1.0000e+00	-2.8602e-09	-2.2115e-07	-2.7818e-07	-1.9594e-08	-2.1650e-08	-2.5562e-09	4.0644e-09	-1.4433e-08
3	-1.8363e-07	-2.8602e-09	1.0000e+00	2.4032e-07	-5.2298e-08	-1.3926e-08	-6.2282e-09	-3.6669e-08	2.3871e-08	-1.5094e-08
4	-1.2919e-08	-2.2115e-07	2.4032e-07	1.0000e+00	2.7149e-07	2.5841e-08	1.3933e-07	1.8402e-08	-2.4887e-08	1.0914e-08
5	-1.6957e-07	-2.7818e-07	-5.2298e-08	2.7149e-07	1.0000e+00	-1.0408e-08	-4.3555e-07	-3.7153e-08	1.1662e-08	-4.4143e-08
6	-6.2559e-09	-1.9594e-08	-1.3926e-08	2.5841e-08	-1.0408e-08	1.0000e+00	4.8231e-08	1.5157e-07	1.4132e-07	8.1072e-07
7	2.1097e-08	-2.1650e-08	-6.2282e-09	1.3933e-07	-4.3555e-07	4.8231e-08	1.0000e+00	-3.5025e-07	-1.8624e-07	-5.8900e-08
8	1.5197e-08	-2.5562e-09	-3.6669e-08	1.8402e-08	-3.7153e-08	1.5157e-07	-3.5025e-07	1.0000e+00	-4.4830e-07	5.6917e-08
9	-2.6122e-09	4.0644e-09	2.3871e-08	-2.4887e-08	1.1662e-08	1.4132e-07	-1.8624e-07	-4.4830e-07	1.0000e+00	-4.7437e-08
10	-2.3824e-08	-1.4433e-08	-1.5094e-08	1.0914e-08	-4.4143e-08	8.1072e-07	-5.8900e-08	5.6917e-08	-4.7437e-08	1.0000e+00

TABLE A.3: Orthonormality of V_{10} for first order Flanschwelle. Item (i, j) in the table is $v_i^T v_j$

Bibliography

[1] H. An, J. Wen, and T. Feng. "On finite difference approximation of a matrix-vector product in the Jacobian-Free Newton–Krylov method". In: *Journal of Computational and Applied Mathematics* 236.6 (Oct. 2011), pp. 1399–1409.

[2] P. Birken. "Numerical methods for stiff problems". Lecture Notes. Apr. 2015, pp. 31–57.

[3] P. Birken. "Numerical Methods for the Unsteady Compressible Navier-Stokes Equations". Habilitiation Thesis. University of Kassel, 2012.

[4] Wikimedia Commons. `https://commons.wikimedia.org/wiki/File:NACA_0012_Demo.svg`. Online; accessed 30-March-2016.

[5] Ivo Dravins. "Numerical Implementations of the Generalized Minimal Residual Method (GMRES)". Bachelor Thesis. Lund University, Nov. 2015.

[6] J.E. Dennis Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Philadelphia, PA 19104-2688, USA: SIAM, 1996. ISBN: 0-89871-364-1.

[7] D.A. Knoll and D.E. Keyes. "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193 (2004), pp. 357–397.

[8] Y. Saad and M. H. Schultz. "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM J. Sci. Stat. Comput.* 7.3 (1986), pp. 856–869.

[9] H.F. Walker. "Implementation of the GMRES method using Householder transformations". In: *SIAM J. Sci. Stat. Comput.* 9.1 (1988), pp. 152–163.

Bachelor's Theses in Mathematical Sciences 2016:K9
ISSN 1654-6229

LUTFNA-4003-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/