Inexact Newton Methods Applied to Under–Determined Systems

by

Joseph P. Simonis

A Dissertation

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

in

Mathematics

May 2006

APPROVED:

Professor Homer F. Walker, Dissertation Advisor
Mathematical Sciences Department

Professor Joseph D. Fehribach, Committee Member
Mathematical Sciences Department

Professor Arthur C. Heinricher, Committee Member
Mathematical Sciences Department

Dr. John N. Shadid, Committee Member
Sandia National Laboratories

Professor Suzanne L. Weekes, Committee Member
Mathematical Sciences Department

Contents

List of Figures

Abstract

Consider an under-determined system of nonlinear equations $F(x) = 0$, $F : I\!R^m \to I\!R^n$, where F is continuously differentiable and $m > n$. This system appears in a variety of applications, including parameter–dependent systems, dynamical systems with periodic solutions, and nonlinear eigenvalue problems. Robust, efficient numerical methods are often required for the solution of this system.

Newton's method is an iterative scheme for solving the nonlinear system of equations $F(x) = 0$, $F : I\!R^n \to I\!R^n$. Simple to implement and theoretically sound, it is not, however, often practical in its pure form. *Inexact Newton methods* and *globalized inexact Newton methods* are computationally efficient variations of Newton's method commonly used on large-scale problems. Frequently, these variations are more robust than Newton's method. *Trust region methods*, thought of here as globalized exact Newton methods, are not as computationally efficient in the large–scale case, yet notably more robust than Newton's method in practice.

The *normal flow method* is a generalization of Newton's method for solving the system $F : I\!R^m \to I\!R^n$, $m > n$. Easy to implement, this method has a simple and useful local convergence theory; however, in its pure form, it is not well suited for solving large-scale problems. This dissertation presents new methods that improve the efficiency and robustness of the normal flow method in the large–scale case. These are developed in direct analogy with inexact–Newton, globalized inexact–Newton, and trust–region methods, with particular consideration of the associated convergence theory. Included are selected problems of interest simulated in MATLAB.

Chapter 1

Introduction

This dissertation presents newly developed methods for solving the under-determined nonlinear system of equations $F(x) = 0$, $F : I\!R^m \rightarrow I\!R^n$ with $m > n \gg 1$. These methods are shown to be globally robust, locally fast, and computationally efficient on large-scale systems of equations.

We define an under–determined system of nonlinear equations to be any system of nonlinear equations with more unknowns than equations, regardless of the uniqueness or existence of its solutions. These systems appear in a variety of applications. After discretization, certain nonlinear partial differential equation (PDE) eigenvalue problems (e.g. the Bratu problem) and some parameter–dependent PDE's (e.g. the driven cavity problem) take the form $F(x, \lambda) = 0$ with $x \in I\!R^n$ and $\lambda \in I\!R$. Under-determined systems also sometimes appear when calculating periodic orbits of dynamical systems. Here, one seeks $x(0)$ and T satisfying $\int_0^T f(x(t), t)dt = 0$, where $f(x(t), t) = \frac{dx(t)}{dt}$. The function $f(x(t), t)$ is assumed to be nonlinear.

This dissertation is divided into three main sections following the introduction. The first of these presents background material. Here, general notation is discussed along with useful lemmas and definitions. This section also includes descriptions of Newton's method and relevant Newton-like methods for solving the nonlinear system $F(x) = 0$, $F : I\!R^n \to I\!R^n$; these are important since they will be used later as models for the new methods. Relevant Newton-like methods include inexact Newton methods, globalized inexact Newton methods and trust–region methods. Inexact Newton methods only approximately solve the Newton system at each iteration. Details about all of these methods can be found in [2, 21, 31]. Globalized inexact Newton methods impose additional requirements on the generated iterates to improve robustness of the algorithms. See [5, 6, 22] and included references. Trust–region methods for nonlinear systems of equations stem from methods in unconstrained optimization. For an understanding of both the methods and their subsequent adaptation to nonlinear systems see [4, 18, 32, 20] and the references therein. The *normal flow method*, an adaptation of Newton's method for solving under-determined systems, is also presented here. Further material about the adaptation of Newton's method to under–determined systems can be found in [16, 35] and the included references.

The second section presents new methods for solving the under-determined system of nonlinear equations. Here, the methods from the first section are used to motivate generalizations of the normal flow method. The discussion and development of the new methods closely parallel the development of methods in [2, 5]. These generalizations are shown to have fast local convergence properties and to be globally robust. Additionally, if suitably implemented, they are computationally efficient for solving large-scale systems of equations.

The final section discusses numerical experiments. Several specific methods are coded in MATLAB and applied to model problems of interest. The test problems include nonlinear eigenvalue problems (the Bratu problem [13] and the Chan problem [1]), a parameter–dependent fluid flow problem (the driven cavity problem [7, 30]), and periodic orbit calculations (the Brusselator problem in one and two dimensions [15, 10, 11]).

Chapter 2

Overview

2.1 Preliminaries

We begin with a brief overview of assumptions, notation, and a few definitions and lemmas used throughout the text.

- The norm $\|\cdot\|$ is assumed to be the Euclidean norm on vectors or the induced norm on matrices throughout. Most results can be extended to the case of an arbitrary inner–product vector norm.
- The function F is from $I\!R^m$ to $I\!R^n$ and is continuously differentiable. It has a Jacobian matrix denoted by F', an $n \times m$ matrix with $\left\{F'_{ij} \equiv \left(\frac{\partial F_i(x)}{\partial x_j}\right)\right\}_{1\le i\le n, 1\le j\le m}$.
- The δ-neighborhood of a point $x \in I\!R^n$ is the set $N_\delta(x) \equiv \{y \in I\!R^n | \, \|x-y\| < \delta\}$.
- A stationary point of $\|F\|$ is a point $x \in I\!R^m$ for which there does not exist an $s \in I\!R^m$ such that $\|F(x) + F'(x)s\| < \|F(x)\|$. The stationary points include local minimizers of $\|F\|$.

Definition 1 ([4]).

- *Let* $x_* \in \mathbb{R}^m$ *and* $x_k \in \mathbb{R}^m, k = 1, 2, \ldots$ *Then* $\{x_k\}$ *is said to* ***converge*** *to* x_* *if* $\lim_{k\to\infty} \|x_k - x_*\| = 0$.

- *If there exists a constant* $c \in [0, 1)$ *and an integer* $\hat{k} \geq 0$ *such that for all* $k \geq \hat{k}$, $\|x_{k+1} - x_*\| \leq c\|x_k - x_*\|$, *then* $\{x_k\}$ *is said to be* ***q-linearly convergent*** *to* x_*.

- *If for some sequence* $\{c_k\}$ *that converges to* 0, $\|x_{k+1} - x_*\| \leq c_k\|x_k - x_*\|$ *for each* k, *then* $\{x_k\}$ *is said to* ***converge q-superlinearly*** *to* x_*.

- *If* $\{x_k\}$ *converges to* x_* *and there exist constants* $p > 1$ *and* $c \geq 0$ *such that* $\|x_{k+1} - x_*\| \leq c\|x_k - x_*\|^p$ *for each* k, *then* $\{x_k\}$ *is said to converge to* x_* *with* ***q-order at least p***. *If* $p = 2$ *or* $p = 3$, *then the convergence is said to be* ***q-quadratic*** *or* ***q-cubic***, *respectively.*

Throughout, we use q-convergence as opposed to r-convergence. The q stands for "quotient", and r stands for "root". A sequence $\{x_k\}$ converges to x_k with r-order p if $\{\|x_{k+1} - x_*\|\}$ is bounded above by a sequence in $\mathbb{R}$ that converges to zero with q-order p.

Definition 2 ([4]). *A function* g *is* ***Hölder continuous*** *with exponent* $p \in (0, 1]$ *and constant* γ *in a set* $\Omega \in \mathbb{R}^m$ *if, for every* x, $y \in \Omega$, $\|g(x) - g(y)\| \leq \gamma\|x - y\|^p$.

Definition 3 ([4]). *A function* g *is* ***Lipschitz continuous*** *with constant* γ *in a set* $\Omega \in \mathbb{R}^m$, *written* $g \in Lip_\gamma(\Omega)$, *if for every* x, $y \in \Omega$, $\|g(x) - g(y)\| \leq \gamma\|x - y\|$.

Definition 4 ([3]). *Given* $F : \mathbb{R}^m \to \mathbb{R}^n$ *continuously differentiable and* $x \in \mathbb{R}^m$, $F'(x)^+$ *is the* ***pseudo–inverse*** *of* $F'(x)$, *if, given* $b \in \mathbb{R}^n$, $F'(x)^+b \in \mathbb{R}^m$ *is the* *solution of $F'(x)s = b$ having minimal Euclidean norm. When $F'(x)$ is of full rank, the pseudo–inverse has the form $F'(x)^+ = F'(x)^T(F'(x)F'(x)^T)^{-1}$ and is called the* ***Moore–Penrose pseudo–inverse****.*

Lemma 1 ([21])**.** *Let $F : \mathbb{R}^m \to \mathbb{R}^n$ be a continuously differentiable function. For any $x \in \mathbb{R}^m$ and $\epsilon > 0$, there exists a $\delta > 0$ such that*

$$\|F(z) - F(y) - F'(y)(z - y)\| \leq \epsilon \|z - y\| \tag{2.1}$$

whenever $y,z \in N_\delta(x)$.

Lemma 2 ([21])**.** *Let $F : \mathbb{R}^m \to \mathbb{R}^n$ be continuously differentiable in the open convex set $\Omega \subset \mathbb{R}^m$, let $x \in \Omega$, and let F' be Hölder continuous with exponent p and constant γ at x in the neighborhood Ω. Then, for any $x + s \in \Omega$,*

$$\|F(x + s) - F(x) - F'(x)s\| \leq \frac{\gamma}{1 + p}\|s\|^{1+p}. \tag{2.2}$$

Proof. By Lemma (4.1.9) in [4],

$$\begin{aligned} F(x + s) - F(x) - F'(x)s &= \left[\int_0^1 F'(x + ts)s\ dt\right] - F'(x)s \\ &= \int_0^1 [F'(x + ts) - F'(x)]s\ dt. \end{aligned}$$

We then obtain

$$\begin{aligned} \|F(x + s) - F(x) - F'(x)s\| &\leq \int_0^1 \|F'(x + ts) - F'(x)\|\|s\|dt \\ &\leq \int_0^1 \gamma\|ts\|^p\|s\|dt \\ &= \gamma\|s\|^{1+p}\int_0^1 t^p\ dt \\ &= \frac{\gamma}{1 + p}\|s\|^{1+p}. \end{aligned}$$

□

Lemma 3. *Assume $F'(x) \in I\!R^{n \times m}$ is a continuous function of x and is of full rank. Then $F'(x)^+$ is a continuous function of x.*

Proof. Because $F'(x)$ is of full rank, the Moore–Penrose pseudo–inverse can be written

$$F'(x)^+ \quad = \quad F'(x)^T (F'(x) F'(x)^T)^{-1}.$$

Since $F'(x)^T$ is a continuous function of x, we have that $F'(x)F'(x)^T$ is a continuous function of x, and it follows that $(F'(x)F'(x)^T)^{-1}$ and, hence, $F'(x)^+$ are continuous functions of x. □

2.2 Newton-like Methods

This section presents three classes of Newton-like methods designed to solve $F(x) = 0$ with $F : I\!R^n \to I\!R^n$. We begin with a description of Newton's method and follow with descriptions of *inexact Newton methods*, *globalized inexact Newton methods*, and *trust region methods*.

2.2.1 Newton's Method

Consider the problem:

$$\text{find } x \in I\!R^n \text{ such that } F(x) = 0, \tag{2.3}$$

where $F : I\!R^n \to I\!R^n$ is continuously differentiable. Newton's method begins by assuming an initial guess, x_0, and generates a sequence of iterates via

$$x_{k+1} = x_k - F'(x_k)^{-1} F(x_k).$$

In practice, this involves solving the linear system

$$F'(x_k) s_k = -F(x_k) \tag{2.4}$$

for the *Newton step* s_k, and then defining $x_{k+1} = x_k + s_k$.

Algorithm NM: Newton's Method

Let x_0 be given.

For $k = 0$ step 1 until ∞ do:

Solve $F'(x_k)s_k = -F(x_k)$

Set $x_{k+1} = x_k + s_k$.

Under mild assumptions the sequence will approach a root of F provided x_0 is sufficiently near the root.

Theorem 1 ([34, 4]). *Suppose F is Lipschitz continuously differentiable at x_*, $F(x_*) = 0$ and $F'(x_*)$ is nonsingular. Then for x_0 sufficiently near x_*, $\{x_k\}$ produced by Newton's method is well-defined and converges to x_* with*

$$\|x_{k+1} - x_*\| \leq c\|x_k - x_*\|^2$$

for a constant c independent of k.

The method is simple to implement and theoretically sound, but, in its pure form, not often used to solve large-scale problems. The exact linear solve at each iteration makes the method computationally inefficient.

2.2.2 Inexact Newton Methods

Inexact Newton methods [2] are variations of Newton's method designed to be computationally efficient on large–scale problems, and are commonly used in the large-scale case. Recall, that the general idea of Newton's method is to linearize F around a current guess, x_k, in hope that the root of the linear model, x_{k+1}, is a better approximation of the root of the nonlinear problem than was x_k. There are two drawbacks with this method. First, solving for a root of the linear model, in practice, may be computationally time–consuming. Second, when far from a solution, the root of the linear model may not be a good approximation of the root of the nonlinear problem. We replace the Newton step with an "inexact" Newton step. We no longer require s_k to exactly solve $F'(x_k)s_k = -F(x_k)$, rather only that s_k be a point where the norm of the local linear model has been reduced. Precisely, we find some $\eta_k \in [0, 1)$ and require s_k to satisfy

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|. \tag{2.5}$$

Notice that as η_k approaches zero, s_k approaches the Newton step. This replacement allows s_k to be calculated "cheaply." Often, an efficient iterative linear solver such as the generalized minimal residual method (GMRES)[1] is used for this calculation. The following algorithm is the inexact Newton method (INM).

Algorithm INM:

LET x_0 BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
FIND **some** $\eta_k \in [0, 1)$ AND s_k THAT SATISFY

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|$$

SET $x_{k+1} = x_k + s_k$.

The scalar η_k is called the *forcing term* and its choice affects both local convergence properties and the robustness of the method [2, 6]. Assume x_* is a solution of (2.3) at which the Jacobian is of full rank. If x_0 is sufficiently close to x_* and $0 \leq \eta_k \leq \eta_{max} < 1$ for each k, then $\{x_k\}$ converges to x_* q-linearly in some norm. Furthermore, q-superlinear convergence is obtained if $\lim_{k\to\infty} \eta_k = 0$. Finally, if $\eta_k = O(\|F(x_k)\|)$, then the convergence is q-quadratic. See [2] for further details.

[1] See Appendix A

2.2.3 Globalized Inexact Newton Methods

The robustness of an inexact Newton method often is enhanced by "globalizations," i.e., augmentations of the basic method that test and modify steps to ensure adequate progress toward a solution [5, 22]. A step satisfying the inexact Newton condition (2.5) yields a decrease in the local linear model norm, yet this decrease is not always reflected in the nonlinear residual norm. In other words, the chosen step may not actually reduce $\|F\|$. To ensure a reduction of $\|F\|$, an additional step selection criterion is added. The step should reduce $\|F\|$ at least some fraction of the reduction predicted by the local linear model of F. More precisely, given a $t \in (0, 1)$, s_k should be chosen to satisfy (2.5) and a sufficient decrease condition:

$$\|F(x_k + s_k)\| \leq [1 - t(1 - \eta_k)]\|F(x_k)\|. \tag{2.6}$$

The resulting algorithm is a globalized inexact Newton method (GINM).

Algorithm GINM:

Let x_0 and $t \in (0, 1)$ be given.
For $k = 0$ step 1 until ∞ do:
Find **some** $\eta_k \in [0, 1)$ and s_k that satisfy

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k\|F(x_k)\|$$

and

$$\|F(x_k + s_k)\| \leq [1 - t(1 - \eta_k)\|F(x_k)\|$$

Set $x_{k+1} = x_k + s_k$.

The following is a global convergence theorem for algorithm GINM.

Theorem 2 ([5])**.** *Assume that algorithm GINM does not break down. If* $\Sigma_{k\geq 0}(1-\eta_k)$ *is divergent, then* $F(x_k) \to 0$. *If, in addition,* x_* *is a limit point of* $\{x_k\}$ *such that* $F'(x_*)$ *is invertible, then* $F(x_*) = 0$ *and* $x_k \to x_*$.

2.2.4 Trust Region Methods

A general trust region method produces a sequence of iterates using the following procedure: at each iteration we assume the local linear model is an accurate representation of the nonlinear function within some closed δ-ball around the current iterate. We choose a step, s, to minimize $\|F(x) + F'(x)s\|$ over all s satisfying $\|s\| \leq \delta$. Then, we check to see if s is acceptable. If it is not acceptable, this indicates the local linear model is not a good representation of F in the δ-ball; δ is decreased and a new s is chosen. This is repeated until an acceptable step is found. The value δ is a measure of our "trust" of the local linear model. One commonly used test for step acceptability is the *ared/pred* condition[5]. Let *ared* be the actual reduction in function norm obtained by taking a step s:

$$ared(s) \equiv \|F(x)\| - \|F(x+s)\|. \tag{2.7}$$

The predicted reduction, *pred*, is the reduction predicted by the local linear model;

$$pred(s) \equiv \|F(x)\| - \|F(x) + F'(x)s\|. \tag{2.8}$$

The *ared/pred* condition for step acceptability requires the actual reduction in function norm to be at least some fraction of the predicted reduction;

$$ared(s) \geq t \cdot pred(s), \; t \in [0, 1). \tag{2.9}$$

The following general trust region method (TR) from [5] is similar in spirit to the method in [18].

Algorithm TR: Trust Region Method

LET x_0, $\bar{\delta}_0 > 0$, $0 < t \leq u < 1$, AND
$0 < \theta_{min} < \theta_{max} < 1$ BE GIVEN
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
SET $\delta_k = \bar{\delta}_k$ AND

CHOOSE $s_k \in \arg\min_{\|s\| \leq \delta_k} \|F(x_k) + F'(x_k)s\|$

WHILE $ared_k(s_k) < t \cdot pred_k(s_k)$ DO:

CHOOSE $\theta \in [\theta_{min}, \theta_{max}]$

UPDATE $\delta_k \leftarrow \theta\delta_k$ AND CHOOSE

$s_k \in \arg\min_{\|s\| \leq \delta_k} \|F(x_k) + F'(x_k)s\|$

SET $x_{k+1} = x_k + s_k$

IF $ared_k(s_k) \geq u \cdot pred_k(s_k)$ CHOOSE $\bar{\delta}_{k+1} \geq \delta_k$;

ELSE CHOOSE $\bar{\delta}_{k+1} \geq \theta_{min}\delta_k$

Theorem 3 ([5]). *Assume that Algorithm TR does not break down. Then every limit point of* $\{x_k\}$ *is a stationary point of* $\|F\|$*. If* x_* *is a limit point of* $\{x_k\}$ *such that* $F'(x_*)$ *is invertible, then* $F(x_*) = 0$ *and* $x_k \to x_*$*; furthermore,* $s_k = -F'(x_k)^{-1}F(x_k)$*, the full Newton step, whenever* k *is sufficiently large.*

2.3 Normal Flow Method

Now, consider an under-determined root–finding problem:

$$\text{find } x \in I\!R^m \text{ such that } F(x) = 0, \tag{2.10}$$

where $F : I\!R^m \to I\!R^n$ is a continuously differentiable function with $m > n$. Here, the linear system (2.4) is under-determined, i.e., it may have an infinite number of solutions. In order to develop a well–defined algorithm, an additional constraint must be imposed so that a unique step, s_k, can be defined. Choosing s_k to be the solution of the linear system (2.4) with minimum Euclidean norm gives the *normal flow method* [35]. The pseudo–inverse solution of the linear system is a natural choice for a "Newton" step because it is the shortest step from the current iterate to a root of the linear problem and, therefore, the linear model is likely to be a better representation of the nonlinear function at that step than at other solutions of (2.4). Hereafter, the normal flow algorithm will be called Newton's method for under-determined systems (NMU).

Algorithm NMU:

LET x_0 BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
LET $s_k = -F'(x_k)^+ F(x_k)$
SET $x_{k+1} = x_k + s_k$.

Mathematically, we have $\|F'(x_k)s_k + F(x_k)\| = 0$ and $s_k \perp \text{Null}(F'(x_k))$. When $F'(x_k)$ is of full rank, s_k will hereafter be referred to as the *Moore-Penrose* step.

A local convergence theory for Algorithm NMU is given in [35] and generalized in [16]. The central result from [35] with respect to this method follows.

Hypothesis 1. *F is differentiable and F' is of full rank n in an open convex set Ω, and the following hold:*

(i) There exist $\gamma \geq 0$ and $p \in (0, 1]$ such that $\|F'(y) - F'(x)\| \leq \gamma\|y - x\|^p$ for all $x, y \in \Omega$.

(ii) There is a constant μ for which $\|F'(x)^+\| \leq \mu$ for all $x \in \Omega$.

Definition 5. *For $\rho > 0$, let $\Omega_\rho = \{x \in \Omega : \|y - x\| < \rho \Rightarrow y \in \Omega\}$.*

Theorem 4 ([35])**.** *Let F satisfy Hypothesis 1 and suppose Ω_η is given by Definition (5) for some $\eta > 0$. Then there is an $\epsilon > 0$ which depends only on γ, p, μ, and η such that if $x_0 \in \Omega_\eta$ and $\|F(x_0)\| < \epsilon$, then the iterates $\{x_k\}_{k=0,1,\ldots}$ determined by Algorithm NMU are well defined and converge to a point $x_* \in \Omega$ such that $F(x_*) = 0$. Furthermore, there is a constant β for which*

$$\|x_{k+1} - x_*\| \leq \beta\|x_k - x_*\|^{p+1}, \; k = 0, 1, \ldots \tag{2.11}$$

If $F'(x)$ is Lipschitz continuous in Ω then $p = 1$ and the iterates produced by Algorithm NMU converge q-quadratically.

Chapter 3

Methods and Theories

3.1 Inexact Newton Methods for Under–Determined Systems

The previous subsection introduced a variation of Newton's method for solving the under-determined system $F(x) = 0,\ F : I\!R^m \rightarrow I\!R^n$, and briefly discussed its local convergence theory. This section presents a class of inexact Newton methods for application to the under-determined system. A convergence theory is developed for these new methods.

Each iteration of NMU requires the step, s_k, satisfying

$$\|F(x_k) + F'(x_k)s_k\| = 0$$

with

$$s_k \perp \mathrm{Null}F'(x_k).$$

Calculation of s_k requires solving a linear system of equations. When n is large, this may be computationally expensive. To improve the computational efficiency, we allow for an approximate solution of the linear system. We seek an s_k satisfying

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|, \text{ where } \eta_k \in [0, 1). \tag{3.1}$$

and

$$s_k \perp \mathrm{Null}(F'(x_k)). \tag{3.2}$$

Constraint (3.1) will henceforth be called the inexact Newton condition. The inexact Newton method for under-determined systems (INMU) follows:

Algorithm INMU:

LET x_0 BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
FIND **some** $\eta_k \in [0, 1)$ AND s_k THAT SATISFY

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|$$
$$s_k \perp \mathrm{Null}(F'(x_k))$$

SET $x_{k+1} = x_k + s_k$.

The remainder of this section presents a theoretical foundation for this algorithm.

Lemma 4. *Assume* $s \perp \mathrm{Null}(F'(x))$, *then* $\|s\| = \|F'(x)^+ F'(x)s\|$.

Proof. Define $\bar{s} = F'(x)^+ F'(x)s$. Then $\bar{s}$ is the pseudo–inverse solution of

$$F'(x)\bar{s} = F'(x)s. \tag{3.3}$$

Additionally, $\bar{s} \perp \mathrm{Null}(F'(x))$ because $\bar{s}$ is the minimum norm solution of the linear problem. Rearranging equation (3.3) gives

$$F'(x)(\bar{s} - s) = 0;$$

therefore, the vector $(\bar{s} - s) \in \mathrm{Null}(F'(x))$. However, because both $\bar{s}$ and s are orthogonal to the null space of the Jacobian, it is also true that $(\bar{s} - s) \in \mathrm{Null}(F'(x))^{\perp}$. Therefore, $(\bar{s} - s) \in \mathrm{Null}(F'(x)) \cap \mathrm{Null}(F'(x))^{\perp} = \{0\}$. We conclude that $\bar{s} = s$. Then, $\|s\| = \|\bar{s}\| = \|F'(x)^+ F'(x)s\|$. □

Theorem 5. *Let F satisfy Hypothesis 1 and suppose $\rho > 0$. Assume that $\eta_k \leq \eta_{\max} < 1$ for $k = 0, 1, \ldots$. Then there is an $\epsilon > 0$ depending only on γ, p, μ, ρ and $\eta_{\max}$ such that if $x_0 \in \Omega_\rho$ and $\|F(x_0)\| \leq \epsilon$, then the iterates $\{x_k\}$ determined by Algorithm INMU are well–defined and converge to a point $x_* \in \Omega$ such that $F(x_*) = 0$.*

Proof. Suppose $x \in \Omega$ and s is such that $s \perp \mathrm{Null}(F'(x))$ and $\|F(x) + F'(x)s\| \leq \eta_{\max}\|F(x)\|$. Define $x_+ \equiv x + s$ and suppose $x_+ \in \Omega$. We can write $\|s\| = \|F'(x)^+ F'(x)s\|$ because $s \perp \mathrm{Null}(F'(x))$. Then

$$\begin{aligned}
\|s\| &\leq \|F'(x)^+\|\|F'(x)s\| \\
&= \|F'(x)^+\|\| - F(x) + F(x) + F'(x)s\| \\
&\leq \|F'(x)^+\|(\|F(x)\| + \|F(x) + F'(x)s\|) \\
&\leq \mu(\|F(x)\| + \eta_{\max}\|F(x)\|) \\
&= \mu(1 + \eta_{\max})\|F(x)\|
\end{aligned}$$

and

$$\begin{aligned}
\|F(x_+)\| &\leq \|F(x_+) - F(x) - F'(x)s\| + \|F(x) + F'(x)s\| \\
&\leq \tfrac{\gamma}{1+p}\|s\|^{1+p} + \eta_{\max}\|F(x)\| \\
&\leq \tfrac{\gamma}{1+p}[\mu(1 + \eta_{\max})]^{1+p}\|F(x)\|^{1+p} + \eta_{\max}\|F(x)\|.
\end{aligned}$$

Choose $\epsilon > 0$ sufficiently small that

$$\tau \equiv \tfrac{\gamma}{1+p}[\mu(1 + \eta_{\max})]^{1+p}\epsilon^p + \eta_{\max} < 1 \text{ and } \tfrac{\epsilon\mu(1+\eta_{\max})}{1-\tau} < \rho.$$

If $\|F(x)\| \leq \epsilon$, then $\|F(x_+)\| \leq \tau\|F(x)\|$.

We argue by induction that if $x_0 \in \Omega_\rho$ and $\|F(x_0)\| \leq \epsilon$ then $\|F(x_{k+1})\| \leq \tau\|F(x_k)\| < \epsilon$ and $x_k \in \Omega$ for all k. First we show that $\|F(x_1)\| \leq \tau\|F(x_0)\| < \epsilon$ and $x_1 \in \Omega$. We have that $s_0 \perp \mathrm{Null}(F'(x_0))$ and $\|F(x_0) + F'(x_0)s_0\| \leq \eta_{\max}\|F(x_0)\|$, so

$$
\begin{aligned}
\|s_0\| &\leq \mu(1+\eta_{\max})\|F(x_0)\| \\
&\leq \mu(1+\eta_{\max})\epsilon \\
&< \tfrac{\mu(1+\eta_{\max})\epsilon}{1-\tau} \\
&< \rho.
\end{aligned}
$$

Therefore we have $x_1 \in \Omega$ because $x_0 \in \Omega_\rho$. By the above argument, $\|F(x_1)\| \leq \tau\|F(x_0)\| < \epsilon$.

Now assume $\|F(x_{j+1})\| \leq \tau\|F(x_j)\| < \epsilon$ and $x_j \in \Omega$ for all $j \leq k$. We show that this is true for $j = k+1$. As before $x_j \in \Omega$, s_j satisfies $s_j \perp \mathrm{Null}(F'(x_j))$, and $\|F(x_j) + F'(x_j)s_j\| \leq \eta_{\max}\|F(x_j)\|$. Then

$$
\begin{aligned}
\|x_{j+1} - x_0\| &\leq \textstyle\sum_{l=0}^{j} \|s_l\| \\
&\leq \textstyle\sum_{l=0}^{j} \mu(1+\eta_{\max})\|F(x_l)\| \\
&\leq \mu(1+\eta_{\max}) \textstyle\sum_{l=0}^{j} \tau^l \epsilon \\
&< \mu(1+\eta_{\max}) \textstyle\sum_{l=0}^{\infty} \tau^l \epsilon \\
&= \tfrac{\mu(1+\eta_{\max})\epsilon}{1-\tau} \\
&< \rho,
\end{aligned}
$$

so $x_0 \in \Omega_\rho$ implies $x_{j+1} \in \Omega$. Again, using the earlier argument,

$$\|F(x_{j+1})\| \leq \tau\|F(x_j)\| \leq \tau^j\|F(x_0)\| < \epsilon.$$

Now, $\|F(x_{k+1})\| \leq \tau\|F(x_k)\|$ implies the sequence $\{\|F(x_k)\|\}$ converges to zero, and since $\|s_k\| \leq \mu(1+\eta_{\max})\|F(x_k)\|$, it must be that $\|s_k\| \to 0$ as $k \to \infty$. Note

$$
\begin{aligned}
\|x_{k+l} - x_k\| &\leq \textstyle\sum_{j=0}^{l-1} \|s_{k+j}\| \\
&\leq \textstyle\sum_{j=0}^{l-1} \mu(1+\eta_{\max})\|F(x_{k+j})\| \\
&\leq \mu(1+\eta_{\max}) \textstyle\sum_{j=0}^{l-1} \tau^j \|F(x_k)\| \\
&< \mu(1+\eta_{\max}) \textstyle\sum_{j=0}^{\infty} \tau^j \|F(x_k)\| \\
&= \tfrac{\mu(1+\eta_{\max})}{1-\tau}\|F(x_k)\| \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\tau^k\|F(x_0)\| \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\tau^k\epsilon.
\end{aligned}
$$

Therefore, $\{x_k\}$ is a Cauchy sequence. It has a limit $x_* \in \Omega$. The continuity of F yields $F(x_*) = 0$. $\square$

Theorem 6. *Let F satisfy Hypothesis 1 and suppose $\rho > 0$. Assume that $0 \leq \eta_k \leq \eta_{\max} < 1$ for $k = 0, 1, \ldots$ and that $\{x_k\}$ produced by Algorithm INMU converges to $x_* \in \Omega$ such that $F(x_*) = 0$. Let $M \equiv \|F'(x_*)\|$ and assume $\|F'(x_k)\| \leq 2M$ for all $x_k \in \Omega$. If $\epsilon > 0$ and $\eta_{\max}$ in the proof above are chosen sufficiently small such that*

$$\tau \equiv \tfrac{\gamma}{1+p}(\mu(1+\eta_{\max}))^{1+p}\epsilon^p + \eta_{\max} < \tfrac{1}{(1+\eta_{\max})2M\mu+1} \text{ and } \tfrac{\epsilon\mu(1+\eta_{\max})}{1-\tau} < \rho,$$

then $\{x_k\}$ converges to x_* q-linearly.

Proof. Start with

$$\begin{aligned}
\|x_{k+1} - x_*\| &\leq \textstyle\sum_{j=k+1}^{\infty} \|s_j\| \\
&\leq \textstyle\sum_{j=k+1}^{\infty} \mu(1+\eta_{\max})\|F(x_j)\| \\
&= \mu(1+\eta_{\max}) \textstyle\sum_{j=k+1}^{\infty} \|F(x_j)\| \\
&\leq \mu(1+\eta_{\max}) \textstyle\sum_{j=0}^{\infty} \tau^j \|F(x_{k+1})\| \\
&= \tfrac{\mu(1+\eta_{\max})}{1-\tau}\|F(x_{k+1})\| \\
&\leq \tfrac{\mu(1+\eta_{\max})\tau}{1-\tau}\|F(x_k)\| \\
&= \tfrac{\mu(1+\eta_{\max})\tau}{1-\tau}\|F(x_k) - F(x_*)\| \\
&\leq \tfrac{2M\mu(1+\eta_{\max})\tau}{1-\tau}\|x_k - x_*\|.
\end{aligned}$$

By the choice of ϵ, the term $\frac{2M\mu(1+\eta_{\max})\tau}{1-\tau}$ is less than 1. Therefore $\{x_k\}$ is q-linearly convergent. $\square$

Theorem 7. *Let F satisfy Hypothesis 1 and suppose $\rho > 0$. Assume that $0 \leq \eta_k \leq \eta_{\max} < 1$ for $k = 0, 1, \ldots$ and that $\{x_k\}$ produced by Algorithm INMU converges to $x_* \in \Omega$ such that $F(x_*) = 0$. If $\eta_k \to 0$, then $\{x_k\}$ converges to x_* q-superlinearly. If $\eta_k = O(\|F(x_k)\|^p)$, then $\{x_k\} \to x_*$ with q-order $1 + p$.*

Proof. Let $M \equiv \|F'(x_*)\|$. There exists a $\delta > 0$ such that $\|F'(x)\| \leq 2M$ and $\|F(x+s) - F(x) - F'(x)s\| < \frac{\gamma}{1+p}\|s\|^{1+p}$ whenever $\|x - x_*\| \leq \delta$. Assume that k is sufficiently large that $\|x_k - x_*\| \leq \delta$.

We first show superlinear convergence. We have

$$\begin{aligned}
\|F(x_{k+1})\| &\leq \|F(x_{k+1}) - F(x_k) - F'(x_k)s_k\| + \|F(x_k) + F'(x_k)s_k\| \\
&\leq \tfrac{\gamma}{1+p}\|s_k\|^{1+p} + \eta_k\|F(x_k)\| \\
&\leq \tfrac{\gamma}{1+p}\left[\mu(1+\eta_k)\|F(x_k)\|\right]^{1+p} + \eta_k\|F(x_k)\| \\
&\leq \tfrac{\gamma}{1+p}\left[2M\mu(1+\eta_k)\|x_k - x_*\|\right]^{1+p} + \eta_k 2M\|x_k - x_*\|.
\end{aligned}$$

Previous calculations give

$$\begin{aligned}
\|x_{k+1} - x_*\| &\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\|F(x_{k+1})\| \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\left[\tfrac{\gamma}{1+p}[2M\mu(1+\eta_k)\|x_k - x_*\|]^{1+p} + \eta_k 2M\|x_k - x_*\|\right] \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\left[\tfrac{\gamma}{1+p}[2M\mu(1+\eta_k)]^{1+p}\|x_k - x_*\|^p + \eta_k 2M\right]\|x_k - x_*\|.
\end{aligned}$$

Now, let $c_k \equiv \frac{\gamma}{1+p}\left[2M\mu(1+\eta_k)\right]^{1+p}\|x_k - x_*\|^p + \eta_k 2M$. Combined, $\lim_{k\to\infty}\|x_k - x_*\| = 0$ and $\lim_{k\to\infty}\eta_k = 0$ imply $\lim_{k\to\infty} c_k = 0$. Thus, $\{x_k\}$ is q-superlinearly convergent. Now assume $\eta_k = O(\|F(x_k)\|^p)$. Because η_k is on the order of $\|F(x_k)\|^p$, there exists a constant C independent of k such that $\|\eta_k\| \leq C\|F(x_k)\|^p \leq C(2M)^p\|x_k - x_*\|^p$ for all sufficiently large k. Then

$$\begin{aligned}
\|x_{k+1} - x_*\| &\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\left[\tfrac{\gamma}{1+p}2M\mu(1+\eta_k)^{1+p}\|x_k - x_*\|^p + \eta_k\mu\right]\|x_k - x_*\| \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\Big[\tfrac{\gamma}{1+p}2M\mu(1+\eta_k)^{1+p}\|x_k - x_*\|^p \\
&\quad + C(2M)^p\mu\|x_k - x_*\|^p\Big]\|x_k - x_*\| \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\left[\tfrac{\gamma}{1+p}2M\mu(1+\eta_k)^{1+p} + C(2M)^p\mu\right]\|x_k - x_*\|^{1+p} \\
&\leq \tfrac{\mu(1+\eta_{\max})}{1-\tau}\left[\tfrac{\gamma}{1+p}2M\mu(1+\eta_{\max})^{1+p} + C(2M)^p\mu\right]\|x_k - x_*\|^{1+p},
\end{aligned}$$

which gives q-order $1+p$ convergence. □

It follows that, if F is Lipschitz continuous, then $p = 1$, and we have q-quadratic convergence.

3.2 A Globalized Inexact Newton Method for Under–Determined Systems

Inexact Newton methods for under-determined systems can achieve fast local convergence rates. Under mild assumptions, including an x_0 such that $\|F(x_0)\|$ is sufficiently small, the sequence generated by Algorithm INMU converges to a solution of problem (2.10). However, if an acceptable x_0 cannot be found, the sequence may fail to converge. Here, the goal is to augment the step selection criteria of Algorithm INMU with a sufficient decrease condition. In analogy with GINM, the additional requirement on the chosen steps is meant to increase the likelihood that the iterates converge to a solution, given an arbitrary x_0. Additionally, we seek a modification that retains the fast rates of convergence.

Each step, s_k, must still satisfy the inexact Newton conditions (3.1) and (3.2). We now also require that the step reduce the norm of F at least some fraction of the reduction predicted by the local linear model. Given some $t \in (0,1)$, s_k should be chosen such that

$$\|F(x_k + s_k)\| \leq [1 - t(1 - \eta_k)]\|F(x_k)\|, \tag{3.4}$$

the same criterion chosen by Eisenstat and Walker in [5]. They note that step criteria (3.1) and (3.4) are similar to acceptability tests used in certain minimization algorithms [18, 32] and methods for solving nonlinear equations [12, 26]. Imposing this additional constraint yields our globalized inexact Newton method for under-determined systems (GINU)

Algorithm GINMU: Global Inexact Newton Method For Under–Determined Systems

Let x_0 and $t \in (0,1)$ be given.

For $k = 0$ step 1 until ∞ do:

FIND **some** $\eta_k \in [0, 1)$ AND s_k THAT SATISFY

$$\|F(x_k) + F'(x_k)s_k\| \le \eta_k \|F(x_k)\|$$
$$s_k \perp \mathrm{Null}(F'(x_k))$$
and
$$\|F(x_k + s_k)\| \le [1 - t(1 - \eta_k)]\|F(x_k)\|$$

SET $x_{k+1} = x_k + s_k$.

The first lemma shows that an s_k satisfying (3.1), (3.2) and (3.4) can be found for each k so long as $F(x_k) \neq 0$ and x_k is not a stationary point of $\|F\|$.

Lemma 5. *Let x and $t \in (0, 1)$ be given and assume that there exists an $\bar{s}$ satisfying $\|F(x) + F'(x)\bar{s}\| < \|F(x)\|$ and $\bar{s} \perp \mathrm{Null}(F'(x))$. Then there exists $\eta_{\min} \in [0, 1)$ such that, for any $\eta \in [\eta_{\min}, 1)$, there is an s satisfying*

$$\|F(x) + F'(x)s\| \le \eta\|F(x)\|$$
$$\|F(x + s)\| \le [1 - t(1 - \eta)]\|F(x)\|$$
$$s \perp \mathrm{Null}(F'(x)).$$

Proof. Clearly $F(x) \neq 0$ and $\bar{s} \neq 0$. Set

$$\begin{aligned} \bar{\eta} &\equiv \tfrac{\|F(x)+F'(x)\bar{s}\|}{\|F(x)\|}, \\ \epsilon &\equiv \tfrac{(1-t)(1-\bar{\eta})\|F(x)\|}{\|\bar{s}\|}, \\ \eta_{\min} &\equiv \max\left\{\bar{\eta}, 1 - \tfrac{(1-\bar{\eta})\delta}{\|\bar{s}\|}\right\}, \end{aligned}$$

where $\delta > 0$ is sufficiently small that

$$\|F(x + s) - F(x) - F'(x)s\| \le \epsilon\|s\|$$

whenever $\|s\| \le \delta$.

For any $\eta \in [\eta_{\min}, 1)$, let $s \equiv \frac{1-\eta}{1-\bar{\eta}}\bar{s}$. Then

$$
\begin{aligned}
\|F(x)+F'(x)s\| &= \|\tfrac{\eta-\bar\eta}{1-\bar\eta}(F(x))+\tfrac{1-\eta}{1-\bar\eta}(F(x)+F'(x)\bar s)\| \\
&\le \tfrac{\eta-\bar\eta}{1-\bar\eta}\|F(x)\|+\tfrac{1-\eta}{1-\bar\eta}\|F(x)+F'(x)\bar s\| \\
&= \tfrac{\eta-\bar\eta}{1-\bar\eta}\|F(x)\|+\tfrac{1-\eta}{1-\bar\eta}\bar\eta\|F(x)\| \\
&= \eta\|F(x)\|,
\end{aligned}
$$

and, since

$$
\|s\| = \tfrac{1-\eta}{1-\bar\eta}\|\bar s\| \le \tfrac{1-\eta_{\min}}{1-\bar\eta}\|\bar s\| \le \delta,
$$

it follows that

$$
\begin{aligned}
\|F(x+s)\| &\le \|F(x+s)-F(x)-F'(x)s\|+\|F(x)+F'(x)s\| \\
&\le \epsilon\|s\|+\eta\|F(x)\| \\
&= \epsilon\cdot\tfrac{1-\eta}{1-\bar\eta}\|\bar s\|+\eta\|F(x)\| \\
&= (1-t)(1-\eta)\|F(x)\|+\eta\|F(x)\| \\
&= [1-t(1-\eta)]\|F(x)\|.
\end{aligned}
$$

Assume y is an element of the null-space of $F'(x)$. Then

$$
\begin{aligned}
s^T y &= (\tfrac{1-\eta}{1-\bar\eta}\bar s)^T y \\
&= \tfrac{1-\eta}{1-\bar\eta}(\bar s)^T y \\
&= 0.
\end{aligned}
$$

Thus $s \perp \mathrm{Null}(F'(x))$. □

Theorem 8. *Assume that $\{x_k\}$ is generated by Algorithm GINMU. If $\sum_{k\ge 0}(1-\eta_k)$ is divergent, then $F(x_k)\to 0$. If, in addition, x_* is a limit point of $\{x_k\}$ such that $F'(x_*)$ is of full rank, and there exists a Γ independent of k for which*

$$
\|s_k\| \le \Gamma(1-\eta_k)\|F(x_k)\| \tag{3.5}
$$

whenever x_k is sufficiently near x_* and k is sufficiently large, then $F(x_*)=0$ and $x_k\to x_*$.

Proof. From equation (3.4),

$$\begin{aligned}
\|F(x_k)\| &\le [1 - t(1-\eta_{k-1})]\|F(x_{k-1})\| \\
&\le \|F(x_0)\| \prod_{0\le j<k} [1 - t(1-\eta_j)] \\
&\le \|F(x_0)\|\exp\left[-\mathrm{t}\sum_{0\le \mathrm{j}<\mathrm{k}} (1-\eta_{\mathrm{j}})\right].
\end{aligned}$$

Since $t > 0$ and $1 - \eta_j > 0$, the divergence of $\sum_{k\ge 0}(1-\eta_k)$ implies $F(x_k) \to 0$.

Suppose that x_* is a limit point of $\{x_k\}$ such that $F'(x_*)$ is of full–rank and that $\{x_k\}$ does not converge to x_*. Let $\delta > 0$ be such that there exist infinitely many k for which $x_k \notin N_\delta(x_*)$ and sufficiently small that (3.5) holds whenever $x_k \in N_\delta(x_*)$ and k is sufficiently large. Since x_* is a limit point of $\{x_k\}$, there exist $\{k_j\}$ and $\{l_j\}$ such that, for each j,

$$\begin{aligned}
x_{k_j} &\in N_{\delta/j}(x_*), \\
x_{k_j+i} &\in N_\delta(x_*),\ i = 0, \ldots, l_j - 1 \\
x_{k_j+l_j} &\notin N_\delta(x_*), \\
k_j + l_j &< k_{j+1}.
\end{aligned}$$

Then for j sufficiently large,

$$\begin{aligned}
\delta/2 &\le \|x_{k_j+l_j} - x_{k_j}\| \\
&\le \textstyle\sum_{k=k_j}^{k_j+l_j-1} \|s_k\| \\
&\le \textstyle\sum_{k=k_j}^{k_j+l_j-1} \Gamma(1-\eta_k)\|F(x_k)\| \\
&\le \textstyle\sum_{k=k_j}^{k_j+l_j-1} \frac{\Gamma}{t}\{\|F(x_k)\| - \|F(x_{k+1})\|\} \\
&= \tfrac{\Gamma}{t}\left\{\|F(x_{k_j})\| - \|F(x_{k_j+l_j})\|\right\} \\
&\le \tfrac{\Gamma}{t}\left\{\|F(x_{k_j})\| - \|F(x_{k_{j+1}})\|\right\}.
\end{aligned}$$

But the last right-hand side converges to zero since $x_{k_j} \to x_*$; hence, this inequality cannot hold for large j. □

An alternate proof of the first half of the theorem follows:

Proof. (Walker, private communication) From equation (3.4),

$$t(1-\eta_{k-1})\|F(x_{k-1})\| \quad \leq \quad \|F(x_{k-1})\| - \|F(x_k)\|$$

and

$$\begin{aligned}
\|F(x_0)\| - \|F(x_k)\| &= \sum_{i=1}^{k}(\|F(x_{i-1})\| - \|F(x_i)\|) \\
&= \sum_{i=1}^{k-1}(\|F(x_{i-1})\| - \|F(x_i)\|) + \|F(x_{k-1})\| - \|F(x_k)\|.
\end{aligned}$$

This implies

$$\begin{aligned}
\|F(x_0)\| &= \sum_{i=1}^{k-1}(\|F(x_{i-1})\| - \|F(x_i)\|) + \|F(x_{k-1})\| \\
&\geq \sum_{i=1}^{k-1}(\|F(x_{i-1})\| - \|F(x_i)\|) \\
&\geq \sum_{i=1}^{k-1}(t(1-\eta_{i-1})\|F(x_{i-1})\|) \\
&= t\sum_{i=1}^{k-1}(1-\eta_{i-1})\|F(x_{i-1})\|.
\end{aligned}$$

Since $t > 0$ and $1 - \eta_j > 0$, the divergence of $\sum_{k\geq 0}(1-\eta_k)$ implies $F(x_k) \to 0$. □

3.3 Backtracking Methods

The global inexact Newton method for an under-determined system presented above generates a sequence of steps satisfying the inexact Newton and sufficient decrease conditions. This section discusses methods for determining satisfactory steps. Assume that an initial step satisfying (3.1) and (3.2) can be found, i.e., an $\bar{s}_k$ approximating the Moore–Penrose step and a forcing term, $\bar{\eta}_k$, are computed. Furthermore, assume this step does not satisfy the sufficient decrease condition. Backtracking methods systematically scale $\bar{s}_k$ and $\bar{\eta}_k$ to find an s_k and η_k satisfying all three conditions (3.1), (3.2), and (3.4). This leads to the under-determined backtracking method (BINMU):

Algorithm BINMU:

LET x_0 AND $t \in (0,1)$, $\eta_{\max} \in [0,1)$, AND
$0 < \theta_{\min} < \theta_{\max} < 1$ BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
FIND **some** $\bar{\eta}_k \in [0, \eta_{\max}]$ AND $\bar{s}_k$ THAT SATISFY

$$\|F(x_k) + F'(x_k)\bar{s}_k\| \leq \bar{\eta}_k \|F(x_k)\|,$$
$$\bar{s}_k \perp \mathrm{Null}(F'(x_k)).$$

EVALUATE $F(x_k + \bar{s}_k)$. SET $\eta_k = \bar{\eta}_k$ AND $s_k = \bar{s}_k$.
WHILE $\|F(x_k + s_k)\| > [1 - t(1 - \eta_k)]\|F(x_k)\|$, DO
CHOOSE $\theta \in [\theta_{\min}, \theta_{\max}]$.
UPDATE $s_k \leftarrow \theta s_k$ AND $\eta_k \leftarrow 1 - \theta(1 - \eta_k)$.
EVALUATE $F(x_k + s_k)$.
SET $x_{k+1} = x_k + s_k$.

If a step satisfying the original inexact Newton condition is found, then properly scaling the step will yield a step satisfying both a modified inexact Newton condition and the associated sufficient decrease condition.

Theorem 9. *Assume that at the k^{th} step of Algorithm BINMU there exists an $\bar{\eta}_k \in [0, \bar{\eta}_{\max}]$ and $\bar{s}_k$ satisfying*

$$\|F(x_k) + F'(x_k)\bar{s}_k\| \leq \bar{\eta}_k \|F(x_k)\|$$
$$\bar{s}_k \perp \mathrm{Null}(F'(x_k)).$$

Also, assume $F'(x_k)$ is of full rank. Then the while–loop will terminate in a finite *number of steps with an s_k and η_k satisfying*

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|$$

$$s_k \perp \text{Null}(F'(x_k))$$

$$\|F(x_{k+1})\| \leq [1 - t(1 - \eta_k)]\|F(x_k)\|.$$

Proof. The ith iteration of the while–loop scales s_k by some $\theta_i \in [\theta_{\min}, \theta_{\max}]$. At the m^{th} step of the while–loop $s_k = \prod_{i=1}^m \theta_i \bar{s}_k$ and $\eta_k = 1 - \prod_{i=1}^m \theta_i(1 - \bar{\eta}_k)$. Notice $\prod_{i=1}^m \theta_i \leq \prod_{i=1}^m \theta_{\max} = \theta_{\max}^m$. Given any $\epsilon > 0$, an m can be found such that $\Theta_m \equiv \prod_{i=1}^m \theta_i < \epsilon$.

Choose m large enough such that

$$\|F(x_k + \Theta_m \bar{s}_k) - F(x_k) - F'(x_k)\Theta_m \bar{s}_k\| \leq C\|\Theta_m \bar{s}_k\|,$$

where $C = \frac{1}{\|F'(x_k)^+\|}(\frac{(1-t)(1-\bar{\eta}_{\max})}{(1+\bar{\eta}_{\max})})$.
We claim that $s_k = \Theta_m \bar{s}_k$ and $\eta_k = 1 - \Theta_m(1 - \bar{\eta}_k)$ are satisfactory. Indeed,

$$\begin{aligned}
\|F(x_k) + F'(x_k)s_k\| &= \|(1 - \Theta_m)F(x_k) + \Theta_m F(x_k) + \Theta_m F'(x_k)\bar{s}_k\| \\
&\leq (1 - \Theta_m)\|F(x_k)\| + \Theta_m\|F(x_k) + F'(x_k)\bar{s}_k\| \\
&\leq (1 - \Theta_m)\|F(x_k)\| + \Theta_m \bar{\eta}_k \|F(x_k)\| \\
&= [1 - \Theta_m + \Theta_m \bar{\eta}_k]\|F(x_k)\| \\
&= [1 - \Theta_m(1 - \bar{\eta}_k)]\|F(x_k)\| \\
&= \eta_k \|F(x_k)\|.
\end{aligned}$$

Further, assume y is an element of the null-space of $F'(x_k)$;

$$\begin{aligned} s_k^T y &= (\Theta_m \bar{s}_k)^T y \\ &= \Theta_m (\bar{s}_k)^T y \\ &= 0. \end{aligned}$$

Finally,

$$\begin{aligned} \|F(x_k + s_k)\| &\leq \|F(x_k) + F'(x_k)s_k\| + \|F(x_k + s_k) - F(x_k) - F'(x_k)s_k\| \\ &\leq \eta_k \|F(x_k)\| + \|F(x_k + s_k) - F(x_k) - F'(x_k)s_k\| \\ &\leq \eta_k \|F(x_k)\| + C\|s_k\| \\ &\leq \eta_k \|F(x_k)\| + C\Theta_m \|F'(x_k)^+\| \|F'(x_k)\bar{s}_k\| \\ &\leq \eta_k \|F(x_k)\| + C\Theta_m \|F'(x_k)^+\| (\|F(x_k)\| + \|F(x_k) + F'(x_k)\bar{s}_k\|) \\ &\leq \eta_k \|F(x_k)\| + C\Theta_m \|F'(x_k)^+\| (1 + \bar{\eta}_k) \|F(x_k)\| \\ &= \left[\eta_k + C\Theta_m \|F'(x_k)^+\| (1 + \bar{\eta}_k)\right] \|F(x_k)\| \\ &= \left[\eta_k + \frac{\Theta_m}{1 + \bar{\eta}_{\max}} (1 - t)(1 - \bar{\eta}_{\max})(1 + \bar{\eta}_k)\right] \|F(x_k)\| \\ &\leq \left[\eta_k + \frac{\Theta_m}{1 + \bar{\eta}_{\max}} (1 - t)(1 - \bar{\eta}_{\max})(1 + \bar{\eta}_{\max})\right] \|F(x_k)\| \\ &\leq \left[\eta_k + \Theta_m (1 - t)(1 - \bar{\eta}_k)\right] \|F(x_k)\| \\ &= \left[\eta_k + 1 - 1 + \Theta_m(1 - \bar{\eta}_k) - t + t - t\Theta_m(1 - \bar{\eta}_k)\right] \|F(x_k)\| \\ &= \left[\eta_k + 1 - (1 - \Theta_m(1 - \bar{\eta}_k)) - t + t(1 - \Theta_m(1 - \bar{\eta}_k))\right] \|F(x_k)\| \\ &= \left[\eta_k + 1 - \eta_k - t + t\eta_k\right] \|F(x_k)\| \\ &= \left[1 - t + t\eta_k\right] \|F(x_k)\| \\ &= \left[1 - t(1 - \eta_k)\right] \|F(x_k)\|. \end{aligned}$$

Therefore, satisfactory s_k and η_k are found in at most m steps, so the while–loop always terminates in a finite number of steps. □

Theorem 10. *Assume that* $\{x_k\}$ *is generated by Algorithm BINMU. Assume* x_* *is a limit point of* $\{x_k\}$ *such that* $F'(x_*)$ *is of full rank. Then* $F(x_*) = 0$ *and* $x_k \to x_*$*. Furthermore,* $\eta_k = \bar{\eta}_k$ *and* $s_k = \bar{s}_k$ *for all sufficiently large* k*.*

Proof. Set $K = \|F'(x_*)^+\|$ and let $\bar{\delta} > 0$ be sufficiently small that $\|F'(x)^+\| \leq 2K$ whenever $x \in N_{\bar{\delta}}(x_*)$. Let $\epsilon = \frac{1}{2K}(\frac{(1-t)(1-\bar{\eta}_{\max})}{(1+\bar{\eta}_{\max})})$. There exists a $\hat{\delta} > 0$ such that

$$\|F(z) - F(y) - F'(y)(z-y)\| \leq \epsilon\|z-y\|$$

for all $z, y \in N_{\hat{\delta}}(x_*)$. Let $\delta = \min\{\bar{\delta}, \hat{\delta}/2\}$. Suppose that $x_k \in N_\delta(x_*)$. Let m be the smallest integer such that $\theta_{\max}^m 2K(1+\eta_{\max})\|F(x_0)\| < \delta$. Then, with Θ_m as in the proof of Theorem 9,

$$\begin{aligned} \|\Theta_m \bar{s}_k\| &\leq \|\theta_{\max}^m \bar{s}_k\| \\ &= \theta_{\max}^m \|\bar{s}_k\| \\ &\leq \theta_{\max}^m 2K(1+\bar{\eta}_k)\|F(x_k)\| \\ &\leq \theta_{\max}^m 2K(1+\eta_{\max})\|F(x_k)\| \\ &\leq \theta_{\max}^m 2K(1+\eta_{\max})\|F(x_0)\| \\ &< \delta. \end{aligned}$$

This implies

$$\|F(x_k + \Theta_m \bar{s}_k) - F(x_k) - F'(x_k)\Theta_m \bar{s}_k\| \leq \epsilon\|\Theta_m \bar{s}_k\|,$$

which, as in the proof of Theorem 9, guarantees that the while–loop terminates in at most m iterations. Therefore, when $x_k \in N_\delta(x_*)$ we have

$$\begin{aligned} 1 - \eta_k &= \Theta_m(1 - \bar{\eta}_k) \\ &\geq \Theta_m(1 - \eta_{\max}) \\ &\geq \theta_{\min}^m(1 - \eta_{\max}). \end{aligned}$$

It is clear that $\theta_{\min}^m(1-\eta_{\max}) > 0$. It is given that x_* is a limit point of $\{x_k\}$, so there are an infinite number of $x_k \in N_\delta(x_*)$. Therefore, the sum $\sum_{k\geq 0}(1-\eta_k)$ diverges. We claim that $\|s_k\| \leq \Gamma(1-\eta_k)\|F(x_k)\|$ for some Γ independent of k when $x_k \in N_\delta(x_*)$. Indeed,

$$\begin{aligned}
\|s_k\| &\leq \|F'(x_k)^+\|\|F'(x_k)s_k\| \\
&\leq 2K\left(\|F(x_k)\| + \|F(x_k)+F'(x_k)s_k\|\right) \\
&\leq 2K(1+\eta_k)\|F(x_k)\| \\
&\leq 2K(1+\eta_{\max})\|F(x_k)\| \\
&\leq \tfrac{2K(1+\eta_{\max})(1-\eta_k)}{\theta_{\min}^m(1-\eta_{\max})}\|F(x_k)\| \\
&= \Gamma(1-\eta_k)\|F(x_k)\|
\end{aligned}$$

with

$$\Gamma = \frac{2K(1+\eta_{\max})}{\theta_{\min}^m(1-\eta_{\max})}.$$

Therefore, by Theorem 8 we have that $F(x_*) = 0$ and $x_k \to x_*$. To show $\eta_k = \bar{\eta}_k$ for all sufficiently large k, it is sufficient to show that

$$\|F(x_k+\bar{s}_k) - F(x_k) - F'(x_k)\bar{s}_k\| \leq \epsilon\|\bar{s}_k\| \tag{3.6}$$

for all sufficiently large k. Equation (3.6) is true if $\|\bar{s}_k\| < \delta$. Note that $x_k \in N_\delta(x_*)$ for all sufficiently large k, and, therefore $\|F'(x_k)^+\| \leq 2K$. Now

$$\begin{aligned}
\|\bar{s}_k\| &\leq \|F'(x_k)^+\|\|F'(x_k)\bar{s}_k\| \\
&\leq 2K(\|F(x_k)\| + \|F(x_k)+F'(x_k)\bar{s}_k\|) \\
&\leq 2K(1+\bar{\eta}_k)\|F(x_k)\| \\
&\leq 2K(1+\eta_{\max})\|F(x_k)\|.
\end{aligned}$$

It is clear that $\|F(x_k)\| \to 0$, so there exists some $\bar{k}$ such that for all $k > \bar{k}$ we have $2K(1+\eta_{\max})\|F(x_k)\| < \delta$. Therefore, for $k > \bar{k}$, $\|s_k\| < \delta$, which implies (3.6). $\square$

3.3.1 Choosing the Scaling Factor

This section does not contribute to the mathematical literature, but it is included here for completeness. In-depth descriptions of the subsequent methods can be found in [4] and [34].

A scaling factor $\theta \in [\theta_{\min}, \theta_{\max}]$ must be chosen at each iteration of the while–loop in Algorithm BINMU. The goal is to choose θ such that $x_{k+1} = x_k + \theta s_k$ is an acceptable next iterate. Ideally, θ minimizes $\|F(x_k + \theta s_k)\|$, or equivalently $\|F(x_k + \theta s_k)\|^2$. However, this 1-dimensional minimization problem may be computationally expensive. An alternative is to find an easy-to-minimize approximation to $\|F(x_k + \theta s_k)\|^2$. Two of the most popular schemes for choosing θ are described below.

The quadratic backtracking method chooses θ to be the minimizer of a quadratic polynomial approximating the function $g(\theta) = \|F(x_k + \theta s_k)\|^2$. Let $p(\theta)$ denote this quadratic polynomial. The polynomial can be defined using three pieces of information; $g(0) = \|F(x_k)\|^2$, $g(1) = \|F(x_k + s_k)\|^2$ and $g'(0) = 2F(x_k)^T F'(x_k)s_k$. Notice the first two are already known, and the third is relatively inexpensive to calculate. Using these three values, $p(\theta)$ is determined, and its minimizer can be calculated. The quadratic polynomial is given by

$$p(\theta) = [g(1) - g(0) - g'(0)]\theta^2 + g'(0)\theta + g(0). \tag{3.7}$$

The derivatives are:

$$p'(\theta) = 2[g(1) - g(0) - g'(0)]\theta + g'(0)$$

and

$$p''(\theta) = 2[g(1) - g(0) - g'(0)].$$

If $p''(\theta) \leq 0$, then the quadratic function is concave down, so choose $\theta = \theta_{\max}$. If $p''(\theta) > 0$, then find θ such that $p'(\theta) = 0$:

$$\begin{aligned} 0 &= 2[g(1) - g(0) - g'(0)]\theta + g'(0) \\ \Rightarrow \theta &= \tfrac{-g'(0)}{2[g(1)-g(0)-g'(0)]}. \end{aligned}$$

Correcting to ensure $\theta \in [\theta_{\min}, \theta_{\max}]$, one obtains θ. The method updates $\theta s_k \rightarrow s_k$ and $1 - \theta(1 - \eta_k) \rightarrow \eta_k$ and then checks to see if the updated s_k and η_k satisfy the step–selection criterion.

The cubic backtracking method uses a cubic polynomial to approximate $g(\theta) = \|F(x_k + \theta s_k)\|^2$ after the first step-length reduction. The cubic polynomial, $p(\theta)$, is constructed using four interpolation values. Again, θ is chosen to be the minimizer of $p(\theta)$ over $[\theta_{\min}, \theta_{\max}]$. On the first step reduction there is no clear way to choose a fourth value, so just three are chosen, and a quadratic polynomial is minimized, yielding θ_1. If subsequent reductions are necessary four values are available. The two values $g(0)$ and $g'(0)$ are used, along with values of g at the two previous θ values. For example, θ_2 is found using $g(0)$, $g'(0)$, $g(\theta_1)$, and $g(1)$. Generalizing, θ_i uses $g(0)$, $g'(0)$, $g(\theta_{i-1})$, and $g(\theta_{i-2})$.

As in [4], denote the two previous θ values as θ_{prev} and θ_{2prev}. The cubic polynomial approximation of the function $\|F(x + \theta s_k)\|^2$ is

$$p(\theta) = a\theta^3 + b\theta^2 + g'(0)\theta + g(0)$$

with

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\theta_{prev} - \theta_{2prev}} \begin{bmatrix} \frac{1}{\theta_{prev}^2} & \frac{-1}{\theta_{2prev}^2} \\ \frac{-\theta_{2prev}}{\theta_{prev}^2} & \frac{\theta_{prev}}{\theta_{2prev}^2} \end{bmatrix} \begin{bmatrix} g(\theta_{prev}) - g(0) - g'(0)\theta_{prev} \\ g(\theta_{2prev}) - g(0) - g'(0)\theta_{2prev} \end{bmatrix}.$$

The local minimizer of the model is given by $\theta_+ = \frac{-b+\sqrt{b^2-3ag'(0)}}{3a}$. As before, update the step and forcing term by $\theta s_k \rightarrow s_k$ and $1 - \theta(1 - \eta_k) \rightarrow \eta_k$.

3.4 Trust–Region Methods for Under–Determined Systems

Trust–region methods for an under-determined system of equations are very similar to the methods for the fully determined system. That is, we define a region in which the local linear model is expected to be an accurate representation of the nonlinear function. A step is chosen to minimize the local linear model norm within this region and tested to see whether it satisfies the *ared/pred* condition. If it does not, the trust region is shrunk and a new minimum is calculated. To ensure locally fast convergence, the steps must approach the Moore-Penrose steps as $\{x_k\}$ approaches a root of $F(x)$. This condition suggests that each step be chosen such that it is orthogonal to the null space of the Jacobian. The trust–region method for under-determined systems (UTR) becomes:

Algorithm UTR:

LET x_0, $\bar{\delta}_0 > 0$, $0 < t \le u < 1$, AND
  $0 < \theta_{\min} < \theta_{\max} < 1$ BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL ∞ DO:
  SET $\delta_k = \bar{\delta}_k$ AND
    CHOOSE $s_k \in \arg\min_{\|s\| \le \delta_k} \|F(x_k) + F'(x_k)s\|$
    WITH $s_k \perp \mathrm{Null}(F'(x_k))$.
  WHILE $ared_k(s_k) < t \cdot pred_k(s_k)$ DO:
    CHOOSE $\theta \in [\theta_{\min}, \theta_{\max}]$.
    UPDATE $\delta_k \leftarrow \theta\delta_k$, AND CHOOSE
      $s_k \in \arg\min_{\|s\| \le \delta_k} \|F(x_k) + F'(x_k)s\|$
      WITH $s_k \perp \mathrm{Null}(F'(x_k))$.
  SET $x_{k+1} = x_k + s_k$.
  IF $ared_k(s_k) \ge u \cdot pred_k(s_k)$ CHOOSE $\bar{\delta}_{k+1} \ge \delta_k$;
    ELSE CHOOSE $\bar{\delta}_{k+1} \ge \theta_{\min}\delta_k$.

The analogy between UTR and TR is manifest in this section. Parallels between the theorems presented here and in [5] reflect this close relationship. The following Lemma was shown in [5] in the fully determined case and is extended here to the under–determined case.

Lemma 6. *Assume that* $\{x_k\}$ *is such that* $pred_k(s_k) \geq (1-\eta_k)\|F(x_k)\|$ *and* $ared_k(s_k) \geq t \cdot (1-\eta_k)\|F(x_k)\|$ *for each* k*, where* $t \in (0,1)$ *is independent of* k*. If* x_* *is a limit point of* $\{x_k\}$ *such that there exists a* Γ *independent of* k *for which* $\|s_k\| \leq \Gamma \cdot pred_k(s_k)$ *whenever* x_k *is sufficiently near* x_* *and* k *is sufficiently large, then* $x_k \longrightarrow x_*$*.*

Proof. Assume that $\{x_k\}$ does not converge to x_*. Let $\delta > 0$ be such that there exist infinitely many k for which $x_k \notin N_\delta(x_*)$ and sufficiently small that $\|s_k\| \leq \Gamma \cdot pred_k(s_k)$ holds whenever $x_k \in N_\delta(x_*)$ and k is sufficiently large.
Since x_* is a limit point of $\{x_k\}$, there exist $\{k_j\}$ and $\{l_j\}$ such that, for each j,

$$\begin{aligned} x_{k_j} &\in N_{\delta/j}(x_*), \\ x_{k_j+i} &\in N_\delta(x_*),\ i = 0, \ldots, l_j - 1 \\ x_{k_j+l_j} &\notin N_\delta(x_*), \\ k_j + l_j &< k_{j+1}. \end{aligned}$$

Then for j sufficiently large,

$$\begin{aligned} \delta/2 &\leq \|x_{k_j+l_j} - x_{k_j}\| \\ &\leq \sum_{k=k_j}^{k_j+l_j-1} \|s_k\| \\ &\leq \sum_{k=k_j}^{k_j+l_j-1} \Gamma \cdot pred_k(s_k) \\ &\leq \sum_{k=k_j}^{k_j+l_j-1} \tfrac{\Gamma}{t} ared_k(s_k) \\ &= \tfrac{\Gamma}{t}\{\|F(x_{k_j})\| - \|F(x_{k_j+l_j})\|\} \\ &\leq \tfrac{\Gamma}{t}\{\|F(x_{k_j})\| - \|F(x_{k_{j+1}})\|\}. \end{aligned}$$

But the last right-hand side converges to zero because F is continuous and $x_{k_j} \to x_*$; hence, this inequality cannot hold for large j. $\square$

Lemma 7. *Assume that $\{x_k\}$ is a sequence generated by Algorithm UTR. Suppose that x_* is a limit point of $\{x_k\}$ such that there exists a Γ independent of k for which*

$$\|s_k\| \leq \Gamma \cdot pred_k(s_k) \tag{3.8}$$

whenever x_k is sufficiently near x_* and k is sufficiently large. Then $x_k \to x_*$ and $\liminf_{k\to\infty} \delta_k > 0$.

Proof. It is clear that $\{x_k\}$ satisfies the hypotheses of Lemma 6, and it follows immediately that $x_k \to x_*$. Choose $\delta > 0$ such that (3.8) holds whenever $x_k \in N_\delta(x_*)$ and k is sufficiently large and also such that

$$\|F(y) - F(x) - F'(x)(y-x)\| \leq \frac{1-u}{\Gamma}\|y-x\| \tag{3.9}$$

whenever $x, y \in N_\delta(x_*)$. Let k_0 be such that if $k \geq k_0$, then $x_k \in N_{\delta/2}(x_*)$ and (3.8) holds.

We claim if $k \geq k_0$, then the while-loop in Algorithm UTR terminates with

$$\delta_k \geq \min\{\bar{\delta}_{k_0}, \theta_{\min}\delta/2\}.$$

Note that if $k \geq k_0$ and if s_k is a trial step for which $\|s_k\| \leq \delta/2$, then (3.8) and (3.9) give

$$\begin{aligned}
ared_k(s_k) &\equiv \|F(x_k)\| - \|F(x_k + s_k)\| \\
&\geq \|F(x_k)\| - \|F(x_k) + F'(x_k)s_k\| - \|F(x_k+s_k) - F(x_k) - F'(x_k)s_k\| \\
&\geq pred_k(s_k) - \tfrac{1-u}{\Gamma}\|s_k\| \\
&\geq pred_k(s_k) - (1-u)pred_k(s_k) \\
&\geq u \cdot pred_k(s_k)
\end{aligned}$$

From this, we see that the while–loop terminates when $\delta_k \leq \delta/2$. So, if the while–loop never reduces the radius, we have $\delta_k = \bar{\delta}_k$. If reductions occur, the loop terminates on or before the iteration that first brings $\delta_k \leq \delta/2$, which implies $\delta_k \geq \theta_{\min}\delta/2$. So

$$\delta_k \geq \min\{\bar{\delta}_k, \theta_{\min}\delta/2\}. \tag{3.10}$$

Furthermore, if $\delta_k \leq \delta/2$ on termination, then $\bar{\delta}_{k+1} \geq \delta_k$; whereas, if $\delta_k > \delta/2$ on termination, then $\bar{\delta}_{k+1} \geq \theta_{\min}\delta_k > \theta_{\min}\delta/2$. Thus

$$\delta_{k+1} \geq \min\{\bar{\delta}_k, \theta_{\min}\delta/2\}. \tag{3.11}$$

Induction on (3.10) and (3.11) gives that the while–loop terminates with

$$\delta_k \geq \min\{\bar{\delta}_{k_0}, \theta_{\min}\delta/2\}.$$

Finally, $\delta_k \geq \min\{\bar{\delta}_{k_0}, \theta_{\min}\delta/2\}$ implies $\liminf_{k\to\infty} \delta_k \geq \min\{\bar{\delta}_{k_0}, \theta_{\min}\delta/2\} > 0$. □

Lemma 8. *If x_* is such that $F'(x_*)$ is of full rank, then there exist Γ and $\epsilon_* > 0$ such that for any $\delta > 0$,*

$$s \in \arg\min_{\|\bar{s}\|\leq\delta} \|F(x) + F'(x)\bar{s}\| \tag{3.12}$$

and

$$s \perp \mathrm{Null}(F'(x_k)) \tag{3.13}$$

satisfies

$$\|s\| \leq \Gamma\{\|F(x)\| - \|F(x) + F'(x)s\|\} \tag{3.14}$$

whenever $x \in N_{\epsilon_*}(x_*)$.

Proof. Set $K \equiv \|F(x_*)^+\|$ and let $\epsilon_* > 0$ be sufficiently small that $F'(x)$ is of full rank and $\|F(x)^+\| \leq 2K$ whenever $x \in N_{\epsilon_*}(x_*)$. Suppose that $x \in N_{\epsilon_*}(x_*)$ and s is given by (3.12) and (3.14) for an arbitrary $\delta > 0$. Denote $s^{MP} \equiv -F'(x)^+F(x)$. s^{MP} is the unique global minimizer of the norm of the local linear model orthogonal to the null space of $F'(x)$. We claim that $\|s\| \leq \|s^{MP}\|$. Indeed, if $\|s^{MP}\| \leq \delta$ then

$s = s^{MP}$, otherwise the radius is too small to include s^{MP} and $\|s\| \leq \delta < \|s^{MP}\|$. If $s^{MP} = 0$, then $s = 0$ and (3.14) holds trivially for any Γ. If $s^{MP} \neq 0$ then we know that, since s minimizes $\|F(x) + F'(x)\bar{s}\|$ over all $\|\bar{s}\| \leq \delta$ with $\bar{s} \perp \mathrm{Null}(F'(x))$, it must be that $\|F(x) + F'(x)s\| \leq \|F(x) + F'(x)\frac{\|s\|}{\|s^{MP}\|}s^{MP}\|$; therefore,

$$
\begin{aligned}
\|F(x)\| - \|F(x) + F'(x)s\| &\geq \|F(x)\| - \|F(x) + F'(x)\tfrac{\|s\|}{\|s^{MP}\|}s^{MP}\| \\
&= \|F(x)\| - \|F(x) + F'(x)\tfrac{\|s\|}{\|s^{MP}\|}(-F'(x)^{+}F(x))\| \\
&= \|F(x)\| - \|F(x) - \tfrac{\|s\|}{\|s^{MP}\|}F(x)\| \\
&= \|F(x)\| + \tfrac{\|s\| - \|s^{MP}\|}{\|s^{MP}\|}\|F(x)\| \\
&= \tfrac{\|s\|}{\|s^{MP}\|}\|F(x)\|.
\end{aligned}
$$

Since $s^{MP} = -F'(x)^{+}F(x)$ and therefore,

$$
\begin{aligned}
\|s^{MP}\| &\leq \|F'(x)^{+}\|\|F(x)\| \\
&\leq 2K\|F(x)\|
\end{aligned}
$$

we have

$$\frac{1}{2K} \leq \frac{\|F(x)\|}{\|s^{MP}\|}.$$

Hence,

$$\|F(x)\| - \|F(x) + F'(x)s\| \geq \frac{\|s\|}{2K},$$

and (3.14) holds with $\Gamma \equiv 2K$ for all $x \in N_{\epsilon_*}(x_*)$. □

Lemma 9. *If x_* is not a stationary point of $\|F\|$, then there exist Γ, $\delta_* > 0$ and $\epsilon_* > 0$ such that s given by (3.12) satisfies (3.14) whenever $x \in N_{\epsilon_*}(x_*)$ and $0 < \delta < \delta_*$.*

Proof. Let $\epsilon > 0$ be such that if $x \in N_{\epsilon}(x_*)$, then $\|F(x)\| \geq \frac{1}{2}\|F(x_*)\|$. Let s_* be such that $\|F(x_*) + F'(x_*)s_*\| < \|F(x_*)\|$. Choose η_* such that

$$\|F(x_*) + F'(x_*)s_*\|/\|F(x_*)\| < \eta_* < 1.$$

Since F and F' are continuous, there exists $\epsilon_* \in (0, \epsilon]$ such that

$$\|F(x) + F'(x)s_*\| \leq \eta_* \|F(x)\|$$

whenever $x \in N_{\epsilon_*}(x_*)$. Choose $\delta_* \in (0, \|s_*\|)$. Suppose that $x \in N_{\epsilon_*}(x_*)$ and $0 < \delta \leq \delta_*$. For s given by (3.12), we have

$$\begin{aligned}
\|F(x)\| - \|F(x) + F'(x)s\| &\geq \|F(x)\| - \left\| F(x) + F'(x)\tfrac{\|s\|}{\|s_*\|}s_* \right\| \\
&\geq \|F(x)\| - \left(1 - \tfrac{\|s\|}{\|s_*\|}\right)\|F(x)\| \\
&\quad -\tfrac{\|s\|}{\|s_*\|}\|F(x) + F'(x)s_*\| \\
&\geq \tfrac{(1-\eta_*)\|F(x)\|}{\|s_*\|}\|s\| \\
&\geq \tfrac{(1-\eta_*)\|F(x_*)\|}{2\|s_*\|}\|s\|,
\end{aligned}$$

and (3.14) holds with

$$\Gamma \equiv \frac{2\|s_*\|}{(1-\eta_*)\|F(x_*)\|}.$$

□

Theorem 11. *Assume that $\{x_k\}$ is a sequence produced by Algorithm UTR. Then every limit point of $\{x_k\}$ is a stationary point of $\|F\|$. If x_* is a limit point of $\{x_k\}$ such that $\|F'(x_*)\|$ is of full rank, then $F(x_*) = 0$ and $x_k \to x_*$; furthermore, $s_k = -F'(x_k)^+F(x_k)$ whenever k is sufficiently large.*

Proof. Assume that x_* is a limit point of $\{x_k\}$ that is not a stationary point of $\|F\|$.

We claim that, for any $\delta > 0$, there exists an $\epsilon > 0$ such that if $x_k \in N_\epsilon(x_*)$ and k is sufficiently large, then $\delta_k \leq \delta$. If this were not true, then there would exist some

$\delta > 0$ and $x_{k_j} \subseteq x_k$ such that $x_{k_j} \to x_*$ and $\delta_{k_j} > \delta$ for each j. Then

$$\begin{aligned}
0 &= \lim_{j\to\infty} \{\|F(x_{k_j})\| - \|F(x_{k_{j+1}})\|\} \\
&\geq \lim_{j\to\infty} \{\|F(x_{k_j})\| - \|F(x_{k_j+1})\|\} \\
&= \lim_{j\to\infty} ared_{k_j}(s_{k_j}) \\
&\geq t \cdot \lim_{j\to\infty} pred_{k_j}(s_{k_j}) \\
&= t \cdot \lim_{j\to\infty} \{\|F(x_{k_j})\| - \|F(x_{k_j}) + F'(x_{k_j})s_{k_j}\|\} \\
&= t \cdot \lim_{j\to\infty} \{\|F(x_{k_j})\| - \min_{\|s\|\leq\delta_{k_j}} \|F(x_{k_j}) + F'(x_{k_j})s\|\} \\
&\geq t \cdot \lim_{j\to\infty} \{\|F(x_{k_j})\| - \min_{\|s\|\leq\delta} \|F(x_{k_j}) + F'(x_{k_j})s\|\} \\
&= t \cdot \{\|F(x_*)\| - \min_{\|s\|\leq\delta} \|F(x_*) + F'(x_*)s\|\}.
\end{aligned}$$

However, the last right–hand side must be positive since x_* is not a stationary point.

Now let Γ, δ_* and ϵ_* be as in Lemma 9. By the above claim, there exists $\epsilon \in (0, \epsilon_*]$ such that if $x_k \in N_\epsilon(x_*)$ and k is sufficiently large, then $\delta_k \leq \delta_*$. By Lemma 9, we have $\|s_k\| \leq \Gamma pred_k(s_k)$ for Γ independent of k whenever $x_k \in N_\epsilon(x_*)$ and k is sufficiently large. Then Lemma 7 implies that $x_k \to x_*$ and $\liminf_{k\to\infty} \delta_k > 0$. But since the claim implies that $\delta_k \to 0$, this is a contradiction. Hence, x_* must be a stationary point.

Suppose that x_* is a limit point of $\{x_k\}$ such that $F'(x_*)$ is of full rank. Since x_* must be a stationary point, we must have $F(x_*) = 0$. It follows from Lemma 8 that there exists a Γ independent of k for which $\|s_k\| \leq \Gamma pred_k(s_k)$ whenever x_k is sufficiently near x_*. Then Lemma 7 implies that $x_k \to x_*$, and there exists a $\delta > 0$ such that $\delta_k \geq \delta$ for sufficiently large k. Since $x_k \to x_*$ and $F(x_k) \to F(x_*) = 0$, we have that

$$\lim_{k\to\infty} \|F'(x_k)^+ F(x_k)\| \leq \lim_{k\to\infty} \|F'(x_k)^+\|\|F(x_k)\| = \|F'(x_k)^+\|\|F(x_k)\| = 0$$

implies that, for sufficiently large k, the Moore-Penrose step will be accepted. Thus $s_k = -F'(x_k)^+ F(x_k)$ whenever k is sufficiently large. $\square$

3.4.1 Under–Determined Dogleg Method

At each iteration, calculating the trust–region step requires the minimization of $\|F(x) + F'(x)s\|$ over all s such that $\|s\| \leq \delta$. Calculating the trust–region step to a high degree of accuracy is usually prohibitively expensive. Many methods of approximating the step have been developed. Examples include the locally constrained optimal "hook" step method, the dogleg step method and the double dogleg step method, see [4] and [34].

The dogleg method ([25, 24]) is the focus of this section. The method builds a piecewise linear curve, Γ^{DL}, which approximates the curve minimizing the local linear model in the trust–region. In the fully–determined case, the dogleg curve connects the current point to the *Cauchy point* and subsequently the Newton point. The Cauchy point is defined to be "the minimizer of $l(s) \equiv \frac{1}{2}\|F(x) + F'(x)s\|_2^2$ in the steepest descent direction $-\nabla l(0) = -F'(x)^T F(x)$, the steepest descent point,"[34]

$$s_k^{\mathrm{CP}} = -\frac{\| - F'(x)^T F(x)\|_2^2}{\|F'(x)F'(x)^T F(x)\|_2^2} F'(x)^T F(x).$$

Here, we replace the Newton point with the Moore–Penrose point. We denote the Cauchy point by s_k^{CP} and the Moore–Penrose point by s_k^{MP}. In the fully–determined case, the dogleg curve, Γ^{DL}, intersects the trust region boundary at a single point [4]. This result is easily extended to the under–determined case. The dogleg step is the step from the current point to the intersection point. We introduce the under-determined Dogleg method (UDL):

Algorithm UDL:

LET x_0, $0 < \theta_{\min} < \theta_{\max} < 1$ AND $0 < \delta_{\min} \le \delta$ BE GIVEN.

FOR $k = 0$ STEP 1 UNTIL ∞ DO:

CALCULATE $s_k^{\mathrm{MP}} = -F'(x_k)^+ F(x_k)$.

IF $\|s_k^{\mathrm{MP}}\| \le \delta$, $s_k = s_k^{\mathrm{MP}}$.

IF $\|s_k^{\mathrm{MP}}\| > \delta$ THEN DO:

COMPUTE $s_k^{\mathrm{CP}} = -\frac{\|-F'(x_k)^T F(x_k)\|_2^2}{\|F'(x_k)F'(x_k)^T F(x_k)\|_2^2} F'(x_k)^T F(x_k)$.

IF $\|s_k^{\mathrm{CP}}\| \ge \delta$, THEN $s_k = \frac{\delta}{\|s_k^{\mathrm{CP}}\|} s_k^{\mathrm{CP}}$.

IF $\|s_k^{\mathrm{CP}}\| < \delta$, THEN $s_k = s_k^{\mathrm{CP}} + \tau(s_k^{\mathrm{MP}} - s_k^{\mathrm{CP}})$,

WHERE τ IS UNIQUELY DETERMINED BY $\|s_k^{\mathrm{CP}} + \tau(s_k^{\mathrm{MP}} - s_k^{\mathrm{CP}})\|$.

WHILE $ared_k(s_k) < t \cdot pred_k(s_k)$ DO:

CHOOSE $\theta \in [\theta_{\min}, \theta_{\max}]$

UPDATE $\delta \leftarrow \max\{\theta\delta, \delta_{\min}\}$

REDETERMINE $s_k \in \Gamma_k^{DL}$

SET $x_{k+1} = x_k + s_k$ AND UPDATE δ.

The dogleg method chooses a step to minimize the linear model norm along the piecewise linear curve within the trust–region. The point where Γ^{DL} intersects the boundary is the minimizer within the trust–region, and can be computed analytically [20]. The following argument verifies this last statement.

Let s be a step from the current point, x_0, to a point along Γ^{DL}. We claim that the length of s increases as Γ^{DL} is traced from x_c to s^{CP} to s^{MP}. Indeed, $\|s\|_2^2$ increases as we move from x_0 to s^{CP}. To show that $\|s\|_2^2$ increases from s^{CP} to s^{MP} we define the function $s(\lambda) = s^{CP} + \lambda(s^{MP} - s^{CP})$ for $\lambda \in [0,1]$. We have $\frac{\partial\|s(\lambda)\|_2^2}{\partial\lambda} \ge 0$ because,

$$\begin{aligned}
\|s(\lambda)\|_2^2 &= \|s^{CP} + \lambda(s^{MP} - s^{CP})\|_2^2 \\
&= \|s^{CP}\|_2^2 + \lambda^2\|s^{MP} - s^{CP}\|_2^2 + 2\lambda(s^{CP})^T(s^{MP} - s^{CP})
\end{aligned}$$

implies

$$\frac{\partial\|s(\lambda)\|_2^2}{\partial\lambda} = 2\lambda\|s^{MP} - s^{CP}\|_2^2 + 2(s^{CP})^T(s^{MP} - s^{CP}),$$

and therefore $\frac{\partial\|s(\lambda)\|_2^2}{\partial\lambda} \geq 0$ if and only if $(s^{CP})^T(s^{MP} - s^{CP}) \geq 0$. We introduce $J \equiv F'(x)$ and $F \equiv F(x)$, and this notation will be used in subsequent equations. Now

$$\begin{aligned}
(s^{CP})^T(s^{MP} - s^{CP}) &= -\tfrac{\|J^TF\|_2^2}{\|JJ^TF\|_2^2}(J^TF)^T\left(-J^+F(x) + \tfrac{\|J^TF\|_2^2}{\|JJ^TF\|_2^2}(J^TF)\right) \\
&= \tfrac{\|J^TF\|_2^2}{\|JJ^TF\|_2^2}\left(F^TJJ^+F - \tfrac{\|J^TF\|_2^2}{\|JJ^TF\|_2^2}F^TJJ^TF\right) \\
&= \tfrac{\|J^TF\|_2^2}{\|JJ^TF\|_2^2}\left(\|F\|_2^2 - \tfrac{\|J^TF\|_2^4}{\|JJ^TF\|_2^2}\right) \\
&= \tfrac{\|J^TF\|_2^2\|F\|_2^2}{\|JJ^TF\|_2^2}\left(1 - \tfrac{\|J^TF\|_2^4}{\|JJ^TF\|_2^2\|F\|_2^2}\right),
\end{aligned}$$

yet

$$\begin{aligned}
\|J^TF\|_2^4 &= (J^TF)^T(J^TF)(J^TF)^T(J^TF) \\
&= F^TJJ^TFF^TJJ^TF \\
&= F^T\|JJ^TF\|_2^2F \\
&= \|JJ^TF\|_2^2\|F\|_2^2,
\end{aligned}$$

which implies

$$\frac{\|J^TF\|_2^4}{\|JJ^TF\|_2^2\|F\|_2^2} = 1,$$

and

$$(s^{CP})^T(s^{MP} - s^{CP}) = 0.$$

Therefore $\frac{\partial\|s(\lambda)\|_2^2}{\partial\lambda} \geq 0$. This proves the claim that the length of s increases as Γ^{DL} is traversed from x_0 to s^{CP} to s^{MP}.

We also claim $\|F(x)+J(x)s\|_2^2$ is decreasing along Γ^{DL}. It has been shown $\|F(x)+J(x)s\|_2^2$ decreases along the dogleg curve from x_0 to s^{CP} [4]. We must also show that $\|F(x) + J(x)s\|_2^2$ decreases from s^{CP} to s^{MP} along Γ^{DL}. Let

$$s(\lambda) = s^{CP} + \lambda(s^{MP} - s^{CP})$$

Then

$$\begin{aligned}
\|F + Js(\lambda)\|_2^2 &= F^TF + 2[J^TF]^T(s^{CP} + \lambda(s^{MP} - s^{CP})) \\
&\quad +(s^{CP} + \lambda(s^{MP} - s^{CP}))^TJ^TJ(s^{CP} + \lambda(s^{MP} - s^{CP})).
\end{aligned}$$

Taking the derivative with respect to λ,

$$\begin{aligned}
\frac{\partial \|F+Js(\lambda)\|_2^2}{\partial \lambda} &= 2(J^TF)^T(s^{MP}-s^{CP}) + (s^{CP})^TJ^TJ(s^{MP}-s^{CP}) + \\
&\quad (s^{MP}-s^{CP})^TJ^TJs^{CP} + 2\lambda(s^{MP}-s^{CP})^TJ^TJ(s^{MP}-s^{CP}) \\
&= 2[(J^TF)^T + (s^{CP})^TJ^TJ](s^{MP}-s^{CP}) + \\
&\quad \lambda(s^{MP}-s^{CP})^TJ^TJ(s^{MP}-s^{CP}) \\
&= 2[(J^TF)^T + (s^{CP})^TJ^TJ](s^{MP}-s^{CP}) + \\
&\quad \lambda\|J(s^{MP}-s^{CP})\|,
\end{aligned}$$

from which it follows that $\frac{\partial \|F+Js(\lambda)\|_2^2}{\partial \lambda}$ is an increasing function of λ. So if the right–hand side is not positive at $\lambda = 1$, then $\|F + Js(\lambda)\|_2^2$ is decreasing. However, we know $\|F + Js(\lambda)\|_2^2 = 0$ at s^{MP}, and so the right–hand side is negative for $\lambda \in [0, 1]$. Hence $\|F(x) + J(x)s\|_2^2$ is decreasing along Γ^{DL}.

Finally, we must verify that steps along the dogleg curve are orthogonal to the null space of the Jacobian. It is already known that the Moore–Penrose step is orthogonal to the null space. Assume t is an element of Null$(F'(x))$. The Cauchy step is $s^{CP} = -\frac{\|-F'(x)^TF(x)\|_2^2}{\|F'(x)F'(x)^TF(x)\|_2^2}F'(x)^TF(x)$. Then

$$\begin{aligned}
(s^{CP})^Tt &= \left(-\frac{\|-F'(x)^TF(x)\|_2^2}{\|F'(x)F'(x)^TF(x)\|_2^2}F'(x)^TF(x)\right)^Tt \\
&= -\frac{\|-F'(x)^TF(x)\|_2^2}{\|F'(x)F'(x)^TF(x)\|_2^2}F(x)^TF'(x)t \\
&= -\frac{\|-F'(x)^TF(x)\|_2^2}{\|F'(x)F'(x)^TF(x)\|_2^2}F(x)^T0 \\
&= 0
\end{aligned}$$

The dogleg curve consists of linear combinations of zero, the Cauchy step, and the Moore–Penrose step; therefore, we can conclude that steps along the dogleg curve are orthogonal to the null space of the Jacobian.

Chapter 4

Numerical Experiments

The numerical experiments do not provide an extensive comparison of the methods but are meant to highlight three things. First, these new methods are able to solve under-determined systems of nonlinear equations. Second, the methods achieve fast rates of convergence when near a solution of the problem. Finally, the inexact methods are computationally more efficient than the exact methods.

4.1 Test Problems

The test problems arise from typical test problems for nonlinear system solvers. Here, they are modified to be under–determined problems.

4.1.1 The Bratu Problem

The Bratu (or Gelfand) problem is a nonlinear eigenvalue problem of the form

$$\Delta u + \lambda e^{u} = 0, \text{ in } \Omega, \; u = 0 \text{ on } \partial\Omega. \tag{4.1}$$

A detailed description of the Bratu problem can be found in [8] and [19], with additional information on its solution in [6] and [33]. Figure 4.1, provided by H. Walker, it shows the solution space for the Bratu problem along with a typical solution. In practice, an initial u is calculated by fixing λ and then applying a nonlinear solver to the arising system. For our tests, we treat λ as an additional unknown, and solve the corresponding under–determined system.

For our tests, we assume that $\Omega = [0, 1] \times [0, 1]$. We discretized using centered differences on a 50×50 uniform grid. This leads to 2501 total unknowns in 2500 equations. For our tests we used an initial guess of

$$u = 2\sin(\pi x)\sin(\pi y),\ \ \lambda = 7.0.$$

Previous work [8] shows that the Newton equation is difficult to solve for fine grids. Therefore, preconditioning the linear system is often necessary. As done in [33], we right–precondition using a Poisson solver.

4.1.2 The Chan Problem

The Chan problem is a nonlinear eigenvalue problem similar to the Bratu problem. A description can be found in [1] with solutions in [34] and [33]. The problem is

$$\Delta u + \lambda\left(1 + \frac{u + u^2/2}{1 + u^2/100}\right) = 0,\ \text{ in } \Omega,\ u = 0 \text{ on } \partial\Omega. \tag{4.2}$$

Figure 4.2 shows a solution of the Chan problem. For our tests we assume that $\Omega = [0, 1] \times [0, 1]$ and λ is treated as an unknown. We discretized using centered differences on a 50×50 uniform grid. This leads to 2501 total unknowns in 2500 equations. The initial guess for the Chan problem is

$$u = 1.0,\ \ \lambda = 0.0$$

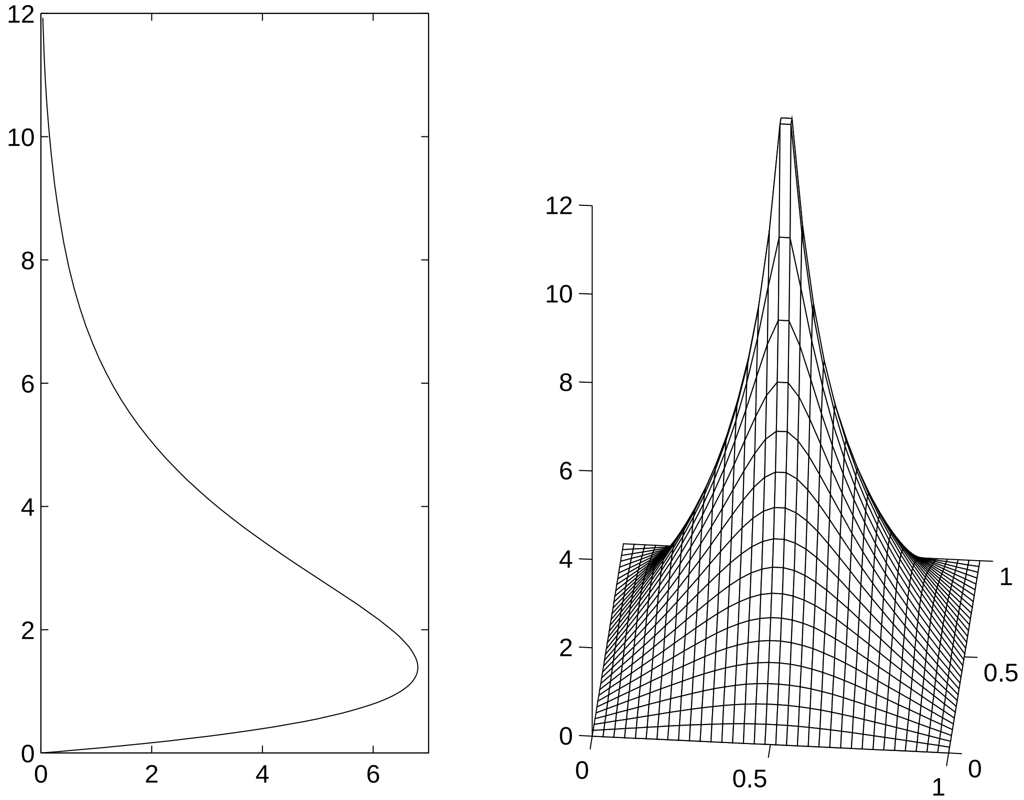

Figure 4.1: The left panel depicts the solution space of the discretized Bratu problem. The x-axis is the λ value and the y-axis is $\|u\|_\infty$. The right panel is a representative solution of the problem.

4.1.3 The Lid–Driven Cavity Problem

The lid–driven cavity problem involves a confined fluid flow in a square box. Circulation of the fluid is driven by a moving top boundary, or "lid". The two sides and bottom are held fixed while the top is moving from left to right. We use the stream function formulation of this problem. The Reynolds number is a nondimensional parameter; as the Reynolds number is increased, areas of counter circulation appear in the corners of the domain, and the problem becomes increasingly difficult to solve [22]. We denote the stream function by u, and the Reynolds number by Re. Usually, this problem is solved by fixing the Reynolds number and solving for the stream

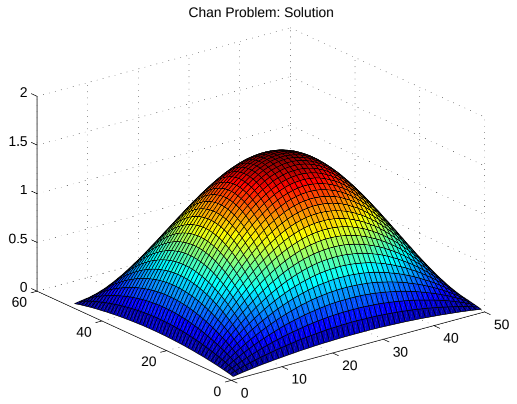

Figure 4.2: A plot of a solution of the Chan problem, at $\lambda = 7.740455$

function using a standard nonlinear solver. Here, we treat the Reynolds number as an additional unknown, and solve the corresponding under–determined system.

The linear problem is preconditioned using a biharmonic solver as in [33]. The discretization of the domain is a 40×40 equally–spaced grid. Treating the Reynolds number as an unknown leads to 1601 unknowns in 1600 equations. The code for this problem was provided H. Walker. The problem is a fourth order system

$$\frac{1}{Re}\Delta^2 u - (u_y \Delta u_x + u_x \Delta u_y) = 0 \text{ on } \Omega \tag{4.3}$$

with

$$\begin{aligned} u &= 0 \quad \text{on} \quad \partial\Omega, \\ \tfrac{\partial u}{\partial n} &= 0 \quad \text{on} \quad \text{the sides and bottom,} \\ \tfrac{\partial u}{\partial n} &= 1 \quad \text{on} \quad \text{the top} \end{aligned}$$

Here, Δ is the Laplacian operator, and Δ^2 is the biharmonic operator,i.e., the Laplacian applied twice. The initial data for the lid–driven cavity problem are

$$u = 0, \ Re = 1000.$$

Figure 4.3 contains a contour plot of a solution to the lid–driven cavity problem.

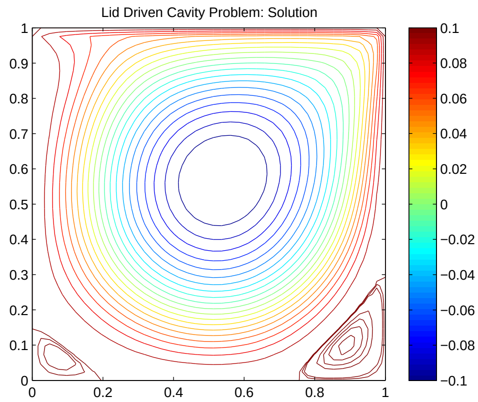

Figure 4.3: A plot of the stream function for the lid–driven cavity with $Re = 1000.00043784100$.

4.1.4 The 1D Brusselator Problem

The one–dimensional Brusselator problem [27] involves a coupled nonlinear system of equations derived from a hypothetical set of chemical reactions. The system of partial differential equations is

$$\frac{\partial u}{\partial t} = \frac{D_u}{L^2}\frac{\partial^2 u}{\partial x^2} + u^2 v - (B+1)u + A \tag{4.4}$$

$$\frac{\partial v}{\partial t} = \frac{D_v}{L^2}\frac{\partial^2 v}{\partial x^2} - u^2 v + Bu \tag{4.5}$$

where u and v represent the concentrations of different chemical species; the parameters A, B, D_u and D_v are fixed at values of 2, 5.45, .008 and .004 respectively. The characteristic length L is the bifurcation parameter of the problem. The Dirichlet boundary conditions are

$$u(t, x = 0) = u(t, x = 1) = A \tag{4.6}$$

$$u(t, x = 0) = v(t, x = 1) = B/A. \tag{4.7}$$

A steady-state solution of this problem is $u = A$ and $v = B/A$. From this we may form a trivial steady-state branch of the corresponding continuation problem. Hopf bifurcations occur at the parameter values of $L_k = 0.5130k,\ k = 1, 2, \ldots$. A Hopf bifurcation is characterized by "the appearance, from equilibrium state, of small–amplitude periodic oscillations."[14] Branches of periodic solutions extend from these bifurcation points. Our goal is to solve for a periodic solution somewhere along one of these branches. We denote the period of a solution by T. The domain is divided into 32 evenly spaced points. Solving for u, v, L, and T yields 64 total unknowns in 62 equations.The initial guess for this problem is the same as used by K. Lust et al. in [17]:

$$u = (sin(3\pi x) + \pi \sin(\pi x))/100 + 2,\ v = .3 \sin(\pi x) + 5.45/2,\ T = 3.017,\ L = .55551.$$

Figure 4.4 depicts part of the solution space for this problem along with an example solution.

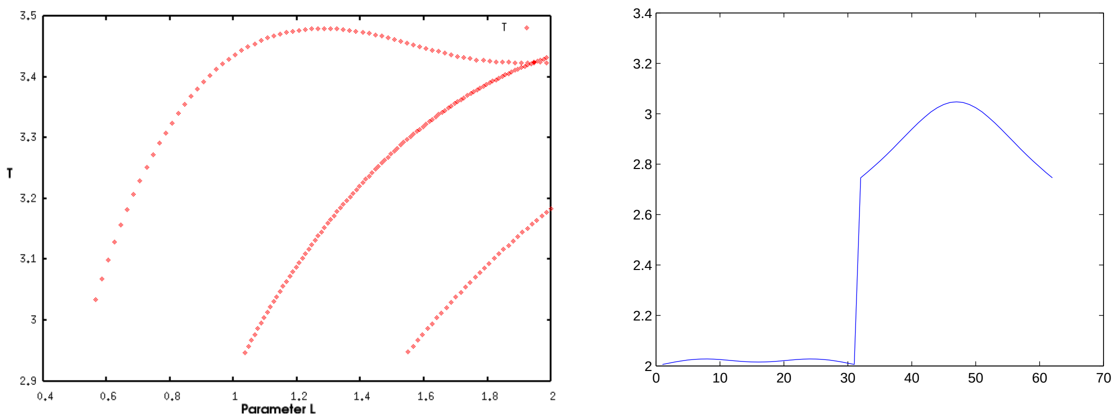

Figure 4.4: The left panel depicts the solution space for the 1D Brusselator problem. The right panel shows a plot of the initial concentration vector $[u, v]$ for a periodic solution with period $T = 3.020249$ at parameter $L = 0.557423$.

4.1.5 The 2D Brusselator Problem

The two–dimensional Brusselator problem is a version of the above chemical reaction occurring on a square grid [10, 11]. As for the previous problem, the goal is to find a periodic solution. However, here I fix the characteristic length and solve only for the two initial concentrations of reactants, u and v, and the period T. The partial differential equation system for this reaction–diffusion problem is:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + 1 + u^2 v - 4.4u \tag{4.8}$$

$$\frac{\partial v}{\partial t} = \alpha \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + 3.4u - u^2 v, \tag{4.9}$$

with $\alpha = .002$. On the boundary, we have Neumann boundary conditions:

$$\frac{\partial u}{\partial n} = 0, \tag{4.10}$$

$$\frac{\partial v}{\partial n} = 0. \tag{4.11}$$

We discretized on a 21×21 uniform grid. With the addition of T as an unknown, there are 883 unknowns in 882 equations. The initial conditions are given by

$$u = 0.5 + y, \;\; v = 1 + 5x, \;\; T = 7.5.$$

4.2 The Solution Algorithms

For the numerical tests, I coded four methods in MATLAB. They are NMU of section 2.3, INMU of section 3.1, the backtracking method of section 3.3 using quadratic backtracking (QINMU), and UDL of section 3.4.1. The methods are applied to each

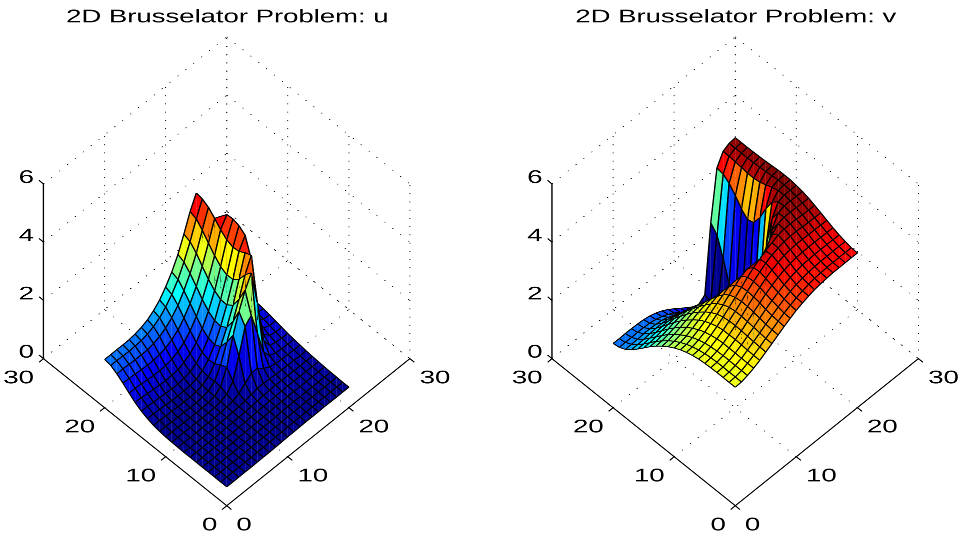

Figure 4.5: The initial concentrations u (left panel) and v (right panel) for a periodic solution of the 2D Brusselator problem with period $T = 7.47997$.

of the five test problems from the previous section. Remember, that NMU is not a new method. It is the model method for comparisons with the the subsequent methods and is used here as a control. Initial inexact Newton steps were calculated using the GMRES method described in Appendix A.

The methods discussed in the previous sections have many parameters that must be set, e.g., the GMRES restart value, maximum forcing term, etc. These parameters affect the performance of the methods. I chose parameters commonly used in the literature. This section lists some of the more important parameters and the values chosen.

- **GMRES**: For the Bratu Problem, the Chan Problem and the 1D Brusselator Problem, I used GMRES with a restart value of 20; the maximum allowed number of iterations is 100. For the 2D Brusselator and the Lid–Driven Cavity problem, the restart value is 50, with a maximum of 500 allowed iterations.

- **Forcing Terms**: For the forcing terms η_k used in both INMU and QINMU, I used *Choice 1* from [6], with $\eta_0 = .9$, $\eta_{max} = .9$, $\eta_{min} = .1$.

- **Backtracking Parameters**: I used $\theta_{min} = .1$, $\theta_{max} = .5$; the maximum number of backtracks allowed is 20.

Both inexact methods (INMU,QINMU) require an orthonormal basis of the null space of the Jacobian at each iteration. To accomplish this, we first build and store the full Jacobian at x_0, and use MATLAB's *null* command. The call returns an orthonormal basis for the null space of the Jacobian. It is calculated by a singular value decomposition. At each subsequent iteration, a corrector to the null space is calculated using the adapted GMRES method discussed in the appendix. Mathematically, if B is an orthonormal basis for the null space of F', with columns b_i, then, at each iteration we calculate correctors, Δb_i, by solving

$$F'(b_i + \Delta b_i) = 0,$$

or

$$F' \Delta b_i = -F' b_i,$$

and orthonormalizing the resulting updated vectors. For further discussion about the computational expense in the large–scale case, see the Summary chapter.

4.3 Results

NMU successfully found a solution for each of the Bratu, Chan, and 1D Brusselator problems. However, the method failed to solve the 2D Brusselator problem and the Driven Cavity problem. In the case of the former, NMU returned a trivial solution; a solution with zero period. INMU, QINMU, and UDL successfully found solutions to

Method/Problem	Bratu	Chan	Driven	1D Bruss	2D Bruss
NMU	2403	3130	25253	573	13661
INMU	722	1150	372	828	1746
QINMU	720	1174	541	874	1545
UDL	2347	3152	4068	572	21349

Figure 4.6: The total compute times, in seconds, for the five test problems.

Method/Problem	Bratu	Chan	Driven	1D Bruss	2D Bruss
NMU	801.00	782.50	252.53	191	525.42
INMU	90.25	115.00	10.05	118.29	47.19
QINMU	90.00	117.40	16.39	124.86	30.90
UDL	782.33	788.00	339.0	190.67	533.73

Figure 4.7: The compute times per nonlinear iteration, in seconds, for the five test problems.

every test problem given. They did not always find the same solution. For example, on the Bratu problem INMU converged to a solution with $\lambda = 6.569$ while UDL converged to a solution with $\lambda = 6.349$.

Table 4.6 shows the amount of time, in seconds, the methods needed to solve each problem. Table 4.7 takes the same times and scales them by the number of iterations. We see INMU and QINMU are indeed more computationally efficient than NMU. UDL has times comparable to NMU. This is expected because UDL calculates the exact Moore–Penrose step at each nonlinear iteration. Figure 4.8 plots the iteration number against the log of the nonlinear residual norm for each method and problem. The Bratu, Chan, and 1D Brusselator problems did not require the globalizations of QINMU and UDL. In these cases, the methods INMU and QINMU produce the same nonlinear residual norms. The methods NMU and UDL, also, produce the same values at each iteration. Because of the overlap in the plots, circles and diamonds are used for plotting the data from QINMU and UDL.

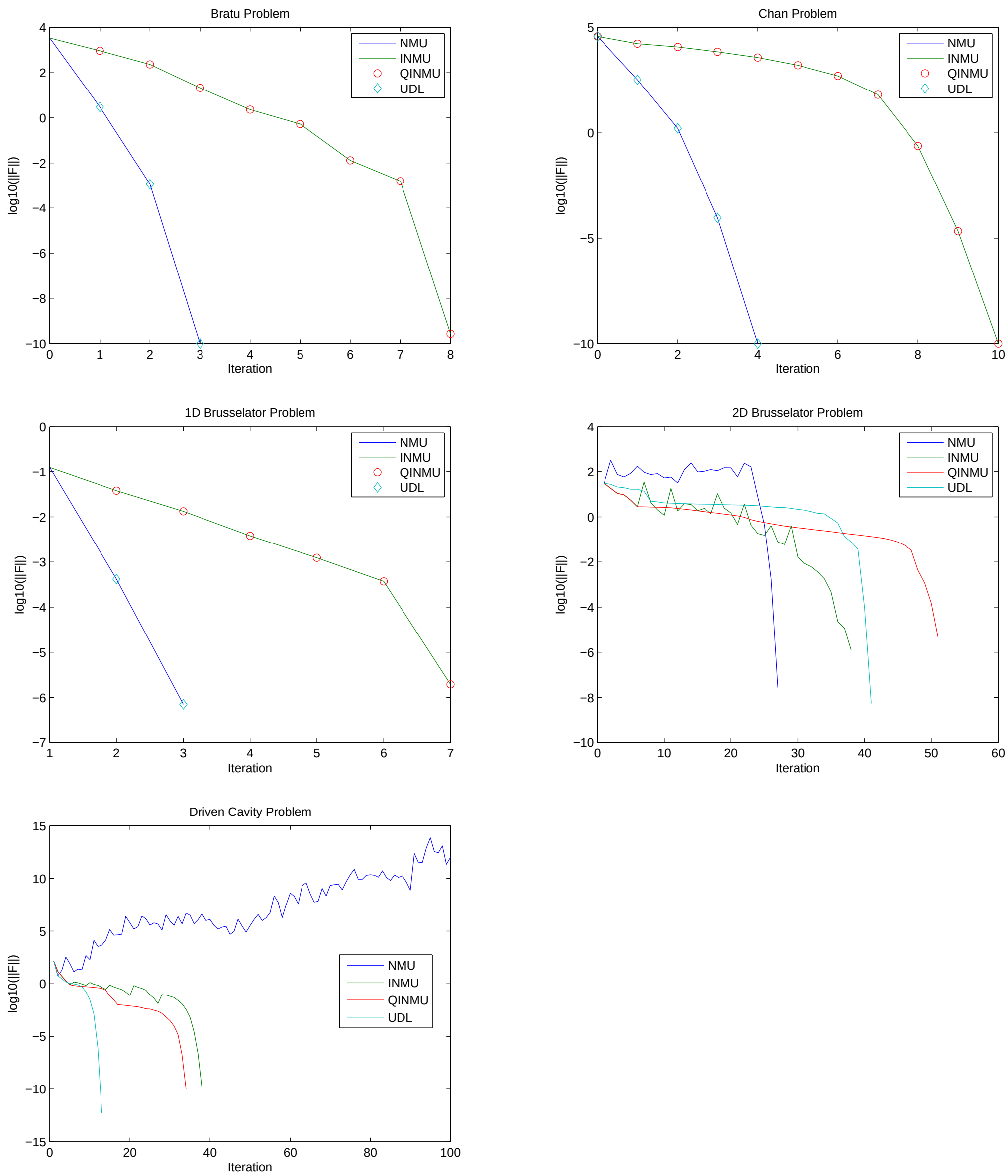

Figure 4.8: For each problem we plot the $\log_{10}(\|F\|)$ at each nonlinear iteration. Where the plots of different methods overlap, we use circles and diamonds to label the iterations.

Chapter 5

Summary

5.1 Summary

This work introduces three classes of methods for solving under–determined systems. Chapter 2 presents background material, including Newton's method for under–determined systems (NMU). Section 3.1 introduces inexact Newton methods for under–determined systems (INMU). Included in §3.1 is a local convergence theory for these new methods. We show, theoretically, that the methods have fast local convergence rates under appropriate assumptions on the forcing terms. In §3.2 we seek to improve the robustness of INMU by imposing a sufficient–decrease condition. This leads to the globalized inexact Newton methods for under–determined systems (GINMU) and, in §3.3, the backtracking method (BINMU). Important here is that BINMU becomes INMU close to a solution; therefore, BINMU can achieve fast local rates of convergence under appropriate assumptions on the forcing terms. Finally, in §3.4, we adapt a general trust–region method to solve an under–determined system. This section includes a discussion of the under–determined dogleg method (UDL), a specific under–determined trust–region method (UTR). A general convergence theory for these methods is presented.

Chapter §4 presents the results from preliminary numerical experiments. Five test problems were coded in MATLAB, and solved with each of four methods: NMU, INMU, QINMU (BINMU with quadratic step–length reduction), and UDL. We found that the three new methods all produced solutions of the problems; additionally, they often exhibited the predicted fast local rates of convergence. The two inexact methods, INMU and QINMU, were computationally more efficient than NMU and UDL, probably because each of the latter required a full linear solve for each nonlinear iteration.

5.2 Additional Applications

The methods presented in this dissertation were originally developed for solving under-determined systems arising from the discretization of parameter–dependent partial differential equations. Often, the dimension of the null space of the Jacobian in these problems is only of dimension one or two. However, these methods are applicable to problems with a null space of much larger dimension.

Another application is solving continuation, homotopy and bifurcation–tracking problems. The new methods may be utilized in two ways. First, one may use an under–determined system method to find an initial point in the solution set of one of these problems, as done above in the Bratu, Chan, and lid–driven cavity problems. Additionally, an under–determined system method may be used for the corrector iterations in a predictor/corrector method for tracing the solution set.

5.3 Future Work

The next logical step, for further study of these methods, is to code the methods in $C++$. This would allow for application to much larger problems. (The current

MATLAB code is limited in the number of total unknowns allowed.) Additionally, a parallel implementation would further increase the maximum allowable size of the problems.

An area which still must be explored is the efficient calculation of the vectors spanning the null space of the Jacobian. The k^{th} nonlinear iteration of our inexact Newton methods requires an orthonormal basis of the null space of $F'(x_k)$. Section 4.2 discusses the method used in our numerical tests. To recap, we build and store the full Jacobian at the initial guess, x_0 and use MATLAB's `null` command to calculate orthonormal spanning vectors. Each subsequent iteration updates the spanning vectors individually. In order to make these methods viable for large–scale simulations, we must find a more computationally efficient method of obtaining the initial null–space basis. A possibility is to begin with a random initial guess for each vector and use the update procedure outlined in §4.2, perhaps repeatedly.

Another possible avenue of research is inexact dogleg methods for under–determined systems. Previously, a general dogleg method was extended to the inexact Newton context in the case of a fully–determined system [23]. In [23], the second point of the dogleg curve (the Newton step) is replaced by an approximation. A similar idea may be applied to the under–determined case; the Moore–Penrose step would be replaced with an inexact–Newton step. A global convergence analysis similar to that in [23] could then be performed.

Appendices

Appendix A

Adaptation of GMRES

GMRES is a *Krylov subspace method*. These methods are designed to solve iteratively the linear problem: find $x \in \mathbf{R}^n$ such that $Ax = b$, $A \in \mathbf{R}^{n \times n}$, $b \in \mathbf{R}^n$. A Krylov subspace method begins with an initial x_0 and at the k^{th} step, determines an iterate x_k through a correction in the k^{th} *Krylov subspace*

$$\mathcal{K}_k \equiv span\{r_0, Ar_0, \ldots, A^{n-1}r_0\}, \tag{A.1}$$

where $r_0 \equiv b - Ax_0$ is the initial residual. A strong attraction of these methods is that implementations only require products Av and sometimes $A^T v$. Thus, within the methods, no direct access to or manipulation of the entries of A is required. GMRES uses only Av products. Detailed descriptions of GMRES and other Krylov subspace methods can be found in [3, 29, 28].

In [33], Walker adapts Krylov subspace methods to solve the problem: find $x \in \mathbb{R}^{n+1}$ such that $Ax = b$, $A \in \mathbb{R}^{n \times (n+1)}$, $b \in \mathbb{R}^n$. This involves imposing the additional constraint that $x \perp \text{Null}(A)$ on the solution to guarantee uniqueness.

It is our goal to extend his method to handle the more general case: find $x \in \mathbb{R}^{n+p}$ such that $Ax = b$, $A \in \mathbb{R}^{n \times (n+p)}$, $b \in \mathbb{R}^n$. First note that this linear system is underdetermined; an additional constraint must be imposed to have a unique solution.

Following the method in [33], assume the constraint is of the form

$$T^T x = 0, \ T \in I\!R^{(n+p)\times p}, \tag{A.2}$$

where the columns of T form an orthonormal basis of the null space of A. This condition is equivalent to choosing the solution of $Ax = b$ with minimum Euclidean norm.

The adapted Krylov methods work by imposing the constraint (A.2) directly on the iterations of standard Krylov methods. This is done as follows:

1. Find $Q \in I\!R^{(n+p)\times n}$ such that $Range(Q) = \{colspaceT\}^\perp$ and $\|Qy\|_2 = \|y\|_2$ for all $y \in I\!R^n$. Then $AQ \in I\!R^{n\times n}$.

2. Apply the Krylov subspace method to solve approximately $AQy = b$ for $y \in I\!R^n$. Then set $x = Qy$.

Just as in [33], $x = Qy$ satisfies (A.2) regardless of how well it approximately satisfies $Ax = b$. Use p Householder transformations to transform T into a triangular matrix.

$$P_p \cdots P_2 P_1 T = \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & x \\ 0 & \cdots & x & x \\ \vdots & \iddots & \vdots & \vdots \\ x & x & x & x \end{pmatrix}$$

The product of these transformations with the $(n+p)\times n$ identity matrix yields an acceptable Q. Algorithm HH forms our Householder transformation vectors [9].

Algorithm HH:

LET $T \in I\!R^{(n+p)\times p}$ BE GIVEN.
FOR $j = 1 : p$

$\beta(j) = \|T(1:n-j+1,j)\|;$

$T(n-j+1,j) = T(n-j+1,j) + sign(T(n-j+1,j)) * \beta(j);$

FOR $k = j+1 : p$

$T(1:n-j+1,k) = T(1:n-j+1,k)...$

$-2/\|T(1:n-j+1,j)\|^2 * T(1:n-j+1,j)...$

$*T(1:n-j+1,j)' * T(1:n-j+1,k);$

END

END

This method produces the Householder vectors u_i in the columns of T. The Householder transformation matrices are then formed by $P_i = I - \frac{2}{\|u_i\|_2^2} u_i u_i^T$. Let I_p denote the $(n+p) \times (n+p)$ identity matrix with the final p columns deleted. The matrix Q is then formed as $Q = P_1 \dots P_{p-1} P_p I_p$. Once Q has been created, it is straightforward to apply a Krylov method.

Appendix B

Matlab Files

NMU.m

```
function [x,resids]=NMU(x,f,jac,tol,maxits)
% Newton's Method for Under-determined systems
% Author: Joseph Simonis
% Latest update: 03-01-06
%
% x             initial guess of solution
% f             function to compute F(x)
% jac           function to compute J(x)
% tol           solution tolerance
% maxits        maximum number of nonlinear iterations
%-----------------------------------------------------

F=feval(f,x);
residual=norm(F);
its=1;
resids(its,1)=residual;
fprintf('\nIt.No.   ||F(u)||     GMRES Its. Lin Mod Norm  Eta \n');
fprintf('  %d   %e      %c        %e    %c\n', 0,residual,'*',0,'*');

while(residual > tol & its<maxits)
    J=feval(jac,x);
    s=-pinv(full(J))*F; % Solve the under-determined lin. sys.
    x=x+s; % Update x
    linnorm=norm(J*s+F);
    F=feval(f,x);
    residual=norm(F);
    fprintf('  %d   %e      %c        %e    %c\n',...
        its,residual,'*',linnorm,'*');
    its=its+1;
    resids(its,1)=residual;
end
```

QINMU.m

```
function [x,resids,fail_count]=QINMU(x,f,jac,jacv,tol,maxits,...
    use_precond,pcond_fun,eta_choice,eta0)
% This function was written as a simple implementation
% of the QINMU method from my dissertation.
% Inexact Newton Method for Under-determined systems
% using quadratic backtracking.
%
% Author: Joseph Simonis
% Latest update: 03-01-06
%
% x             initial guess of solution
% f             function to compute F(x)
% jac           function to compute J(x)
% jacv          function to compute J(x)*v
% tol           solution tolerance
% maxits        maximum number of iterations
% use_precond   flag for turning on preconditioning
% pcond_fun     function to perform preconditioning
% eta_choice    flag for choosing eta choice
% eta_0         the initial eta value
%   If eta_choice==0 eta=constant
%   If eta_choice==1 use eta=|F-linear_residual|/Fprev
%   If eta_choice==2 use eta=gamma*(F/Fprev)^alpha
%
%

%---------------------------------------------------
etamin=.1;
etamax=.9;
maxbtsteps=20;
thetamin=.1;
thetamax=.9;
gmres_restart=20;
gmres_max=100;
%---------------------------------------------------

%We will keep track of the number of backtracking failures
% in with the fail_count
fail_count=0;

% The first step is to compute the Null vectors of the
% Jacobian.
A=feval(jac,x);
t=null(full(A));
for j=1:size(t,2)
    t(:,j)=t(:,j)./norm(t(:,j)); % Scale the null vectors
end

```

```
% Begin Method
F=feval(f,x); % Evaluate F and the residual r=||F||.
residual=norm(F)
n=length(F);
its=1;  % Nonlinear iterations count.
resids(its,1)=residual; % For output
eta=eta0; % The initial forcing term.

fprintf('\nIt.No.   ||F(u)||     GMRES Its. Lin Mod Norm  Eta \n');
fprintf('  %d    %e     %d        %e    %e\n', 0,residual,0,0,eta0);

while(residual > tol & its<maxits)
    % Here I want to update the null vectors t.
    % To do this I solve J(deltat)=-Jt for a correction
    % to t for each direction in the null space.
    for k=1:size(t,2)
        temp = feval(jacv,x,t(:,k));
        [deltat(:,k),error,dummy]=GMRES_House(jacv,temp,x,gmres_restart,...
            (1.0e-3)/norm(temp),gmres_max,t,use_precond,pcond_fun);
        t(:,k)=t(:,k)+deltat(:,k);
    end
    for j=1:size(t,2)
        t(:,j)=t(:,j)./norm(t(:,j));% Scale t
    end % Scale t

    % Now call the under-determined GMRES method.
    [s,rho,ftjs,succ,linits]=GMRES_House(jacv,F,x,gmres_restart,eta,...
        gmres_max,t,use_precond,pcond_fun);

    % Catching errors...
    if ftjs >= 0, error('IN step is not a descent direction.'); end

    if (eta_choice==0) %constant eta
        [s,F,residual,Failure]=quadbt(x,residual,s,eta,ftjs,thetamin,...
            thetamax,f,n+1,maxbtsteps);
        x=x+s;
        residual=norm(F);
        if (Failure==1)  % If the linear solve failed to meet solve tol.
            fail_count=fail_count+1;
        end
    else
        Fprev=F;
        fnrmprev=residual;
        etaprv=eta;
        [s,F,residual,Failure,etaused]=quadbt(x,residual,s,eta,ftjs,...
            thetamin,thetamax,f,n+1,maxbtsteps);
        % Recalculate rho here.  The step has been
        % backtracked, so I need to give etaupdate \|F+J\theta*s\|, not
        % \|F+J*s\|.
        %------------------
        FpJS=Fprev+feval(jacv,x,s);
```

```
        fpjsnorm=norm(FpJS);
        %-------------------
        x=x+s;
        [eta]=etaupdate(eta_choice,fnrmprev,residual,fpjsnorm,...
            etamax,etamin,eta);
        if (Failure==1)
            fail_count=fail_count+1;
        end
    end
    fprintf('  %d   %e     %d        %e     %e\n',its,residual,...
        linits,rho,etaprv);
    its=its+1;
    resids(its,1)=residual;

end
```

INMU.m

```matlab
function [x,resids]=INMU(x,f,jac,jacv,tol,maxits,...
    use_precond,pcond_fun,eta_choice,eta0)
% Inexact Newton Method for Under-determined systems
% Author: Joseph Simonis
% Latest update: 03-01-06
%
% x             initial guess of solution
% f             function to compute F(x)
% jac           function to compute J(x)
% jacv          function to compute J(x)*v
% tol           solution tolerance
% maxits        maximum number of iterations
% use_precond   flag for turning on preconditioning
% pcond_fun     function to perform preconditioning
% eta_choice    flag for choosing eta choice
% eta_0         the initial eta value
%    If eta_choice==0 eta=constant
%    If eta_choice==1 use eta=|F-linear_residual|/Fprev
%    If eta_choice==2 use eta=gamma*(F/Fprev)^alpha
%_______________________________________________________
etamin=.1;
etamax=.9;
maxbtsteps=20;
thetamin=.1;
thetamax=.9;
gmres_restart=20;
gmres_max=100;
%_______________________________________________________

% The first step is to compute the Null vector of the
% Jacobian.
A=feval(jac,x);
t=null(full(A));
for j=1:size(t,2)
    t(:,j)=t(:,j)./norm(t(:,j));% Scale t
end

% Begin Method
F=feval(f,x); % Evaluate F and the residual r=||F||.
residual=norm(F)
n=length(F);
its=1;
resids(its,1)=residual;
eta=eta0; % The initial forcing term.

fprintf('\nIt.No.  ||F(u)||    GMRES Its. Lin Mod Norm  Eta \n');
fprintf('  %d   %e     %d        %e    %e\n', 0,residual,0,0,eta0);

```

```
while(residual > tol & its<maxits)
    % Here I want to update the null vectors t.
    % To do this I solve J(deltat)=-Jt for a correction
    % to t for each direction in the null space.

    for k=1:size(t,2)
        temp = feval(jacv,x,t(:,k));
        [deltat(:,k),error,dummy]=GMRES_House(jacv,temp,x,...
            gmres_restart,(1.0e-3)/norm(temp),gmres_max,t,...
            use_precond,pcond_fun);
        t(:,k)=t(:,k)+deltat(:,k);
    end
    for j=1:size(t,2)
        t(:,j)=t(:,j)./norm(t(:,j)); % Scale t
    end

    % Now call the under-determined GMRES method.
    [s,rho,ftjs,succ,linits]=GMRES_House(jacv,F,x,gmres_restart,eta,...
        gmres_max,t,use_precond,pcond_fun);
    fnrmprev = residual;
    x=x+s;
    F=feval(f,x);
    residual=norm(F);
    etaprv=eta;
    [eta]=etaupdate(eta_choice,fnrmprev,residual,rho,...
        etamax,etamin,eta);
    fprintf('  %d   %e     %d       %e    %e\n', its,residual,...
        linits,rho,etaprv);
    its=its+1;
    resids(its,1)=residual;
end
```

UDL.m

```
function [x,resids]=UDL(x,f,jac,tol,maxits)
% Under-Determined Dogleg method
% Author: Joseph Simonis
% Latest update: 03-01-06
%
% x             initial guess of solution
% f             function to compute F(x)
% jac           function to compute J(x)
% tol           solution tolerance
% maxits        maximum number of iterations
%
%_______________________________________________________
t=10^-4;
thetamin = 0.1;
thetamax = 0.5;
u=.75;
v=0.1;
inneritsmax=20;
%_______________________________________________________

% Algorithm
F=feval(f,x);
residual = norm(F);
fprintf('\nIt.No.   ||F(u)||    GMRES Its. Lin Mod Norm  Delta \n');
fprintf('  %d   %e      %c       %e    %e\n', 0,residual,'*',0,0);
its=1;
innerits=0;
resids(its,1)=residual;
while(residual > tol & its<maxits)
    J=feval(jac,x);
    % Calculate the Moore-Penrose step.
    snewt=-pinv(full(J))*F;
    snewtnorm=norm(snewt);
    if (its==1)
        delta=snewtnorm;
    end
    % Calculate the Dogleg step.
    dogleg_step = Dogleg(F,J,snewt,snewtnorm,delta);
    Fpls = feval(f,x+dogleg_step);
    Fplsn = norm(Fpls);
    Js = J*dogleg_step;
    lin_res = norm(F+Js);
    ared = residual-Fplsn;
    pred = residual-lin_res;
    % Inner Dogleg loop.
    while (ared<t*pred & innerits < inneritsmax);
        if (snewtnorm < delta)
            delta = snewtnorm;
```

```
        end
        d = Fplsn^2-residual^2-2*F'*Js;
        if (d <= 0)
            theta = thetamax;
        else
            theta = -(F'*Js)./d;
            if (theta > thetamax)
                theta = thetamax;
            end
            if (theta < thetamin)
                theta = thetamin;
            end
        end
        delta = theta*delta; % Update Delta
        % Recalculate dogleg step.
        dogleg_step = Dogleg(F,J,snewt,snewtnorm,delta);
        Fpls = feval(f,x+dogleg_step);
        Fplsn = norm(Fpls);
        Js = J*dogleg_step;
        lin_res = norm(F+Js);
        ared = residual-Fplsn;
        pred = residual-lin_res;
        innerits=innerits+1
    end
    innerits = 0;
    x=x+dogleg_step;
    F=Fpls;
    residual=Fplsn;
    fprintf('  %d   %e     %c        %e     %e\n', its,residual,...
        '*',lin_res,delta);
    its = its+1;
    resids(its,1) = residual;
    % Update delta
    if (ared > u.*pred & snewtnorm > delta)
        delta = 2.*delta;
    elseif (ared < v.*pred)
        delta = .5.*delta;
    end
end
```

Dogleg.m

```
function [dogleg_step]=Dogleg(F,J,snewt,snewtnorm,delta)
% Computes the dogleg step
% Inputs:
%   F = F(xcurrent)
%   J = J(xcurrent)
%   snewt = Moore-Penrose Step
%   snewtnorm = norm of Step
%   delta = current trust region radius

if (snewtnorm <= delta)
    dogleg_step = snewt;
else
    JTF=J'*F;
    JTFnorm=norm(JTF);
    JJTFnorm=norm(J*JTF);
    sdescent = -(JTFnorm./JJTFnorm)^2.*JTF;
    sdescentnorm = norm(sdescent);
    if (sdescentnorm >= delta)
        dogleg_step = (delta./sdescentnorm).*sdescent;
    else
        sdiff = snewt-sdescent;
        a = norm(sdiff).^2;
        b = sdescent'*sdiff;
        c = sdescentnorm.^2-delta.^2;
        tao = -c./(b+sqrt(b.^2-a*c));
        dogleg_step = sdescent + tao.*sdiff;
    end
end

```

GMRES_House.m

```
function [step,rho,ftjs,success,ittot] = GMRES_House(jacv,fval,...
    u,m,eta,itmax,T,preconflag,pcond_fun)
% This function is a GMRES routine for under-determined
% systems.  It solves J(u)*step=-fval for step.  We assume
% J(u):R(n+d)->R(n).  T is a normalized basis for the Null(J(u)).
% It contains d vectors.  This function uses a starting guess
% of zeros.
%
% INPUTS:
%     jacv = routine for computing jacobian-vector products.
%     fval = current function value F(u)
%     u = current approx. solution of F(u) = 0
%     m = GMRES restart value
%     eta = forcing term
%     itmax = maximum number of GMRES iterations
%     T = orthonormalized basis of Null(J(u))
%     preconflag = 0-> no preconditioning used
%                = 1-> preconditioning used
%     pcond_fun = preconditioning function
%
% OUTPUTS:
%     step = approx. solution of J(u)*step=-fval
%     rho = \|-fval-J(u)*step\|
%     ftjs = fval'*J(u)*step
%     success = 1 if tol was acheived
%     ittot = total number of gmres iterations

% Compute d Householder reflections and store them in T.
% Reference "An Adaptation of Krylov subspace methods to path
% following problems" H. Walker 1999
d=size(T,2);
n=size(u,1);
for j=1:d
    beta(j)=norm(T(1:n-j+1,j));
    T(n-j+1,j)=T(n-j+1,j)+sign(T(n-j+1,j))*beta(j);
    for k=j+1:d
        T(1:n-j+1,k)=T(1:n-j+1,k)-2/norm(T(1:n-j+1,j))^2*T(1:n-j+1,j)...
            *T(1:n-j+1,j)'*T(1:n-j+1,k);
    end
end

% The GMRES routine
% SETUP
n = size(u,1)-d; %the length of vectors in the range of J.
D = zeros(d,1); % used for appending zeros onto n-vectors.
r = -fval;
rho = norm(r);
tol = eta*rho;
```

```
step = zeros(size(u,1),1); % set the initial guess to zero
ftjs=0;
V = zeros(n,m+1); % Holds the Krylov subspace basis
R = zeros(m,m);
c = zeros(m,1); % s and c are used in computing the Givens rotations
s = c;
w = zeros(m+1,1);
ittot = 0;
% END SETUP

% OUTER LOOP
while (rho > tol & ittot < itmax)
  V(:,1) = r/rho;
  w(1) = rho;
% INNER LOOP
  for k = 1:m
    ittot = ittot + 1;
    if ittot > itmax, break, end

    % The first step is to calculate V_(k+1)=A*V_k
    % Multiplying a vector by A first requires
    % (maybe) multiplying by a preconditioner M^-1.
    % The second step is applying Q=P1...Pd*I
    % which means applying the Householder reflections
    % to the vector appended with zeros.  The final
    % step is to send it to the Jacv routine.

    % Step 1 apply preconditioner and append zeros
    % or just append zeros.
    if (preconflag == 1)
        Hv = feval(pcond_fun, V(:,k));
        Hv = [Hv;D];
    else
        Hv = [V(:,k);D];
    end

    % Step 2 apply Householder reflections
    for j=d:-1:1
        Hv(1:n+d-j+1,1)=Hv(1:n+d-j+1,1)-2/norm(T(1:n+d-j+1,j))^2 ...
        *T(1:n+d-j+1,j)*T(1:n+d-j+1,j)'*Hv(1:n+d-j+1,1);
    end

    % Step 3 Multiply by the Jacobian Matrix.
    V(:,k+1) = feval(jacv,u,Hv);

    % With V_(k+1) calculated it is time to continue with GMRES
    % as normal.
    for i = 1:k
      R(i,k) = V(:,k+1)'*V(:,i);
      V(:,k+1) = V(:,k+1) - R(i,k)*V(:,i);
    end
    for i = 1:k-1
```

```
      temp = R(i,k);
      R(i,k) = c(i)*temp + s(i)*R(i+1,k);
      R(i+1,k) = -s(i)*temp + c(i)*R(i+1,k);
    end
    tempnorm = norm(V(:,k+1));
    temp = sqrt(R(k,k)^2 + tempnorm^2);
    c(k) = R(k,k)/temp;
    s(k) = tempnorm/temp;
    R(k,k) = temp;
    w(k+1) = -s(k)*w(k);
    w(k) = c(k)*w(k);
    rho = abs(w(k+1));
    if rho <= tol, break, end
    V(:,k+1) = V(:,k+1)/tempnorm;
  end
  % END INNER LOOP

  % Solve for step using back substitution.
  for i = k:-1:1
    w(i) = w(i)/R(i,i);
    if i>1
      w(1:i-1) = w(1:i-1) - w(i)*R(1:i-1,i);
    end
  end

  % the step y which solves JQM^-1y=-F
  % is a linear combination of the V's.
  % Here we put y in the temp vector.
  tempvec = V(:,1:k)*w(1:k);

  % To now calculate our step correction we
  % apply M^-1 if neccessary, and then apply
  % Q.
  if (preconflag == 1)
      Hv = feval(pcond_fun,tempvec);
      Hv=[Hv;D];
  else
      Hv=[tempvec;D];
  end
  for j=d:-1:1
      Hv(1:n+d-j+1,1)=Hv(1:n+d-j+1,1)-2/norm(T(1:n+d-j+1,j))^2 ...
      *T(1:n+d-j+1,j)*T(1:n+d-j+1,j)'*Hv(1:n+d-j+1,1);
  end

  % Update the step.
  step = step + Hv;
  V(:,m+1) = feval(jacv,u,step);

  ftjs = fval'*V(:,m+1);
  if ittot > itmax, break, end
  r = - fval - V(:,m+1);
  rho = norm(r);
```

```
end
if (rho<tol)
    success = 1;
else
    success = 0;
end
% END OUTER LOOP
```

quadbt.m

```
function [step,trialf,trialn,Fail,eta]=quadbt(xcur,fcnrm,step,eta,...
    oftjs,thetamin,thetamax,fh,meshsize,maxbtsteps)
% Quadratic Backtracking Method
% Author: Joseph Simonis
% Latest update: 03-01-06
%Find a suitable step through bactracking, also return new eta.

% INPUT
% xcur       current value of x
% fcnrm      norm of F at xcur
% step       initial trial step
% eta        the forcing term
% oftjs      F'JS
% thetamin   the minimum scaling factor per iteration
% thetamax   the maximum scaling factor per iteration
% fh         handle for function evaluations
% meshsize   the size of the mesh
% maxbtsteps maximum allowable backtracking steps

% OUTPUT
% step       final step
% trialf     F at xcur+step
% trialn     ||trialf||
% Fail       1 if backtracking failed to produce an acceptable step
%            0 otherwise
%-------------------------------------------------------------------

Fail=0;
t=10^-4;
accept='no';
redfac=1.0;
ibt=0; %Bactracking iterations.
while (strcmp(accept,'no') & ibt<maxbtsteps)
    trials=xcur+step;  %Take the step
    trialf=feval(fh,trials);  %Determine f at the new value.
    trialn=norm(trialf);  %Find the norm
    % Uncomment the following for printing
%     fprintf('trialn=');
%     fprintf(' %e\n',trialn);
%     fprintf('(1-t*(1-eta))*fcnrm');
%     fprintf(' %e\n',(1-t*(1-eta))*fcnrm);

    if trialn<=(1-t*(1-eta))*fcnrm  %Test our condition residual reduction
        accept='yes';
    else
        ibt=ibt+1;
        %Find theta to reduce our step size.
        phi=trialn^2-fcnrm^2-2*oftjs*redfac;
```

```
        if phi <= 0
            theta=thetamax;
        else
            theta=-(oftjs*redfac)/phi;
        end
        % Keep theta within bounds.
        if theta<thetamin
            theta=thetamin;
        end
        if theta>thetamax
            theta=thetamax;
        end
        step=theta*step;
        %Increase eta.
        eta=1-theta*(1-eta);
        %Update the reduction factor
        redfac=theta*redfac;
        stepnorm=norm(step);
        if (stepnorm)<sqrt(eps) break;end
    end
end
if (strcmp(accept,'no'))
    % We have Backtracking failure, so we take the full step.
    step = (1/redfac)*step;
    disp('Backtracking Failure: Taking full step');
    Fail=1;
    trials=xcur+step;  %Take the step
    trialf=feval(fh,trials);  %Determine f at the new value.
    trialn=norm(trialf);  %Find the norm

end
```

Appendix C

Raw Data

C.1 Chan Problem Results

NMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.751216e+004$	$*$	$0.000000e+000$	$*$
1	$3.318422e+002$	$*$	$2.601388e-009$	$*$
2	$1.627407e+000$	$*$	$2.896797e-010$	$*$
3	$9.151679e-005$	$*$	$1.796905e-012$	$*$
4	$4.070212e-011$	$*$	$1.151032e-016$	$*$

INMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.751216e+004$	0	$0.000000e+000$	$9.000000e-001$
1	$1.700332e+004$	1	$1.700252e+004$	$9.000000e-001$
2	$1.186035e+004$	1	$1.185984e+004$	$8.432626e-001$
3	$6.940770e+003$	2	$6.938583e+003$	$7.589363e-001$
4	$3.719799e+003$	3	$3.714728e+003$	$6.399826e-001$
5	$1.608581e+003$	6	$1.595137e+003$	$4.857060e-001$
6	$4.998264e+002$	11	$4.810233e+002$	$3.108434e-001$
7	$6.449217e+001$	18	$6.253675e+001$	$1.509785e-001$
8	$2.355349e-001$	38	$2.413941e-001$	$3.912209e-003$
9	$2.137906e-005$	64	$2.049753e-005$	$9.085024e-005$
10	$8.904233e-011$	99	$7.830998e-011$	$3.742667e-006$

QINMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.751216e+004$	0	$0.000000e+000$	$9.000000e-001$
1	$1.700332e+004$	1	$1.700252e+004$	$9.000000e-001$
2	$1.186035e+004$	1	$1.185984e+004$	$8.432626e-001$
3	$6.940770e+003$	2	$6.938583e+003$	$7.589363e-001$
4	$3.719799e+003$	3	$3.714728e+003$	$6.399826e-001$
5	$1.608581e+003$	6	$1.595137e+003$	$4.857060e-001$
6	$4.998264e+002$	11	$4.810233e+002$	$3.108434e-001$
7	$6.449217e+001$	18	$6.253675e+001$	$1.509785e-001$
8	$2.355349e-001$	38	$2.413941e-001$	$3.912209e-003$
9	$2.137906e-005$	64	$2.049753e-005$	$9.085024e-005$
10	$8.904233e-011$	99	$7.830998e-011$	$3.742667e-006$

UDL

It.No.	$\|F(u)\|$	Inner Its.	Lin Mod Norm	Delta
0	$3.751216e+004$	$*$	$0.000000e+000$	$0.000000e+000$
1	$3.318422e+002$	0	$2.601388e-009$	$2.613503e+001$
2	$1.627407e+000$	0	$2.896797e-010$	$2.613503e+001$
3	$9.151679e-005$	0	$1.796905e-012$	$2.613503e+001$
4	$4.070212e-011$	0	$1.151032e-016$	$2.613503e+001$

C.2 Bratu Problem Results

NMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.391596e+003$	$*$	$0.000000e+000$	$*$
1	$2.984006e+000$	$*$	$2.878279e-010$	$*$
2	$1.141729e-003$	$*$	$3.152350e-012$	$*$
3	$9.795232e-011$	$*$	$1.518800e-015$	$*$

INMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.391596e+003$	0	$0.000000e+000$	$9.000000e-001$
1	$9.236813e+002$	1	$9.219854e+002$	$9.000000e-001$
2	$2.311795e+002$	1	$2.251650e+002$	$8.432626e-001$
3	$2.090284e+001$	2	$2.156339e+001$	$7.589363e-001$
4	$2.293760e+000$	2	$2.313649e+000$	$6.399826e-001$
5	$5.251453e-001$	1	$5.251154e-001$	$4.857060e-001$
6	$1.297609e-002$	2	$1.298191e-002$	$3.108434e-001$
7	$1.556908e-003$	2	$1.556908e-003$	$1.509785e-001$
8	$2.685922e-010$	7	$1.496858e-012$	$1.614431e-008$

QINMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.391596e+003$	0	$0.000000e+000$	$9.000000e-001$
1	$9.236813e+002$	1	$9.219854e+002$	$9.000000e-001$
2	$2.311795e+002$	1	$2.251650e+002$	$8.432626e-001$
3	$2.090284e+001$	2	$2.156339e+001$	$7.589363e-001$
4	$2.293760e+000$	2	$2.313649e+000$	$6.399826e-001$
5	$5.251453e-001$	1	$5.251154e-001$	$4.857060e-001$
6	$1.297609e-002$	2	$1.298191e-002$	$3.108434e-001$
7	$1.556908e-003$	2	$1.556908e-003$	$1.509785e-001$
8	$2.685922e-010$	7	$1.496858e-012$	$1.614431e-008$

UDL

It.No.	$\|F(u)\|$	Inner Its.	Lin Mod Norm	Delta
0	$3.391596e+003$	$*$	$0.000000e+000$	$0.000000e+000$
1	$2.984006e+000$	0	$2.878279e-010$	$2.883900e+000$
2	$1.141729e-003$	0	$3.152350e-012$	$2.883900e+000$
3	$9.795232e-011$	0	$1.518800e-015$	$2.883900e+000$

C.3 1D Brusselator Problem Results

NMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.235424e-001$	$*$	$0.000000e+000$	$*$
1	$4.163303e-004$	$*$	$9.072548e-016$	$*$
2	$7.057922e-007$	$*$	$1.641872e-018$	$*$

INMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.235424e-001$	0	$0.000000e+000$	$9.000000e-001$
1	$3.789989e-002$	2	$3.805739e-002$	$9.000000e-001$
2	$1.322895e-002$	2	$1.183413e-002$	$8.432626e-001$
3	$3.799495e-003$	2	$3.794278e-003$	$7.589363e-001$
4	$1.236069e-003$	2	$1.241420e-003$	$6.399826e-001$
5	$3.719770e-004$	2	$3.718548e-004$	$4.857060e-001$
6	$1.943773e-006$	3	$5.591030e-005$	$3.108434e-001$

QINMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.235424e-001$	0	$0.000000e+000$	$9.000000e-001$
1	$3.789989e-002$	2	$3.805739e-002$	$9.000000e-001$
2	$1.322895e-002$	2	$1.183413e-002$	$8.432626e-001$
3	$3.799495e-003$	2	$3.794278e-003$	$7.589363e-001$
4	$1.236069e-003$	2	$1.241420e-003$	$6.399826e-001$
5	$3.719770e-004$	2	$3.718548e-004$	$4.857060e-001$
6	$1.943773e-006$	3	$5.591030e-005$	$3.108434e-001$

UDL

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Delta
0	$1.235424e-001$	$*$	$0.000000e+000$	$0.000000e+000$
1	$4.163303e-004$	0	$9.072548e-016$	$1.166288e-001$
2	$7.057922e-007$	0	$1.641872e-018$	$1.166288e-001$

C.4 2D Brusselator Problem Results

NMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.091846e+001$	$*$	$0.000000e+000$	$*$
1	$3.158847e+002$	$*$	$9.229574e-013$	$*$
2	$7.476746e+001$	$*$	$2.272146e-012$	$*$
3	$5.793147e+001$	$*$	$4.698255e-013$	$*$
4	$8.420897e+001$	$*$	$5.588875e-013$	$*$
5	$1.735481e+002$	$*$	$1.384143e-012$	$*$
6	$9.341805e+001$	$*$	$2.204188e-012$	$*$
7	$7.433772e+001$	$*$	$6.375459e-013$	$*$
8	$8.187089e+001$	$*$	$5.886421e-013$	$*$
9	$5.272703e+001$	$*$	$5.943046e-013$	$*$
10	$5.709727e+001$	$*$	$4.390965e-013$	$*$
11	$3.137898e+001$	$*$	$4.711255e-013$	$*$
12	$1.231399e+002$	$*$	$4.549860e-013$	$*$
13	$2.421254e+002$	$*$	$2.145620e-012$	$*$
14	$9.701407e+001$	$*$	$1.564119e-012$	$*$
15	$1.035978e+002$	$*$	$4.093546e-013$	$*$
16	$1.224131e+002$	$*$	$5.522184e-013$	$*$
17	$1.091248e+002$	$*$	$5.061431e-013$	$*$
18	$1.479976e+002$	$*$	$4.815174e-013$	$*$
19	$1.469487e+002$	$*$	$2.120245e-012$	$*$
20	$5.887051e+001$	$*$	$8.264937e-013$	$*$
21	$2.342590e+002$	$*$	$6.491423e-012$	$*$
22	$1.613558e+002$	$*$	$2.085909e-012$	$*$
23	$7.850377e+000$	$*$	$1.231790e-012$	$*$
24	$4.364600e-001$	$*$	$5.738919e-014$	$*$
25	$1.731127e-003$	$*$	$2.919037e-015$	$*$
26	$2.760185e-008$	$*$	$8.454222e-018$	$*$

INMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.091846e+001$	0	$0.000000e+000$	$9.000000e-001$
1	$1.824003e+001$	3	$2.299055e+001$	$9.000000e-001$
2	$1.090754e+001$	1	$1.297936e+001$	$8.432626e-001$
3	$9.452493e+000$	4	$7.310140e+000$	$7.589363e-001$
4	$5.496000e+000$	2	$4.947952e+000$	$6.399826e-001$
5	$2.806162e+000$	2	$2.121726e+000$	$4.857060e-001$
6	$3.511690e+001$	7	$4.437237e-001$	$3.108434e-001$
7	$4.275360e+000$	2	$4.825854e+000$	$9.000000e-001$
8	$2.028126e+000$	2	$1.845342e+000$	$8.432626e-001$
9	$1.177837e+000$	2	$1.175095e+000$	$7.589363e-001$
10	$1.805096e+001$	4	$5.401817e-001$	$6.399826e-001$
11	$1.832720e+000$	2	$1.752990e+000$	$9.000000e-001$
12	$3.848034e+000$	3	$1.154060e+000$	$8.432626e-001$
13	$3.486983e+000$	1	$3.449961e+000$	$9.000000e-001$
14	$1.874952e+000$	2	$1.024239e+000$	$8.432626e-001$
15	$2.405483e+000$	3	$1.265773e+000$	$7.589363e-001$
16	$1.402440e+000$	2	$1.199856e+000$	$6.399826e-001$
17	$1.064270e+001$	5	$6.311441e-001$	$4.857060e-001$
18	$2.459337e+000$	2	$2.335608e+000$	$9.000000e-001$
19	$1.493057e+000$	1	$1.502597e+000$	$8.432626e-001$
20	$4.652817e-001$	2	$4.670447e-001$	$7.589363e-001$
21	$3.739754e+000$	7	$2.099171e-001$	$6.399826e-001$
22	$4.313060e-001$	2	$4.659618e-001$	$9.000000e-001$
23	$1.822714e-001$	2	$1.832953e-001$	$8.432626e-001$
24	$1.541191e-001$	5	$1.327950e-001$	$7.589363e-001$
25	$4.060784e-001$	7	$7.156117e-002$	$6.399826e-001$
26	$7.799654e-002$	2	$7.907521e-002$	$9.000000e-001$
27	$5.906230e-002$	6	$5.633737e-002$	$8.432626e-001$
28	$3.997319e-001$	8	$1.710852e-002$	$7.589363e-001$
29	$1.620284e-002$	2	$1.670986e-002$	$9.000000e-001$
30	$8.511825e-003$	2	$8.513613e-003$	$8.432626e-001$
31	$6.317359e-003$	5	$6.317077e-003$	$7.589363e-001$
32	$3.678229e-003$	7	$3.803187e-003$	$6.399826e-001$
33	$1.806651e-003$	8	$1.454114e-003$	$4.857060e-001$
34	$4.796231e-004$	8	$4.735908e-004$	$3.108434e-001$
35	$2.292151e-005$	9	$2.294165e-005$	$1.509785e-001$
36	$1.163682e-005$	501	$2.299250e-007$	$4.198811e-005$
37	$1.224380e-006$	1	$1.386998e-006$	$4.976503e-001$

QINMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$3.091846e+001$	0	$0.000000e+000$	$9.000000e-001$
1	$1.824003e+001$	3	$2.299055e+001$	$9.000000e-001$
2	$1.090754e+001$	1	$1.297936e+001$	$8.432626e-001$
3	$9.452493e+000$	4	$7.310140e+000$	$7.589363e-001$
4	$5.496000e+000$	2	$4.947952e+000$	$6.399826e-001$
5	$2.806162e+000$	2	$2.121726e+000$	$4.857060e-001$
6	$2.780205e+000$	7	$4.437237e-001$	$3.108434e-001$
7	$2.735780e+000$	7	$3.356503e-001$	$1.509785e-001$
8	$2.664124e+000$	9	$2.959626e-002$	$1.747666e-002$
9	$2.636932e+000$	6	$1.455871e-001$	$7.379728e-002$
10	$2.525357e+000$	6	$9.361902e-002$	$8.947840e-002$
11	$2.355887e+000$	6	$7.058900e-002$	$5.755350e-002$
12	$2.190047e+000$	6	$3.942118e-002$	$3.281008e-002$
13	$2.023608e+000$	6	$2.084467e-002$	$2.957726e-002$
14	$1.853639e+000$	6	$3.170430e-002$	$2.399261e-002$
15	$1.697146e+000$	8	$2.261963e-002$	$1.598077e-002$
16	$1.567189e+000$	8	$2.622484e-002$	$1.555968e-002$
17	$1.447049e+000$	8	$3.160480e-002$	$2.340045e-002$
18	$1.329648e+000$	9	$6.828215e-003$	$2.329787e-002$
19	$1.215426e+000$	9	$7.713509e-003$	$1.886568e-002$
20	$1.107978e+000$	9	$9.061426e-003$	$1.409218e-002$
21	$9.577904e-001$	9	$1.063965e-002$	$1.159057e-002$
22	$7.555593e-001$	9	$1.299614e-002$	$2.656106e-002$
23	$6.359854e-001$	9	$1.259148e-002$	$6.341761e-002$
24	$5.588588e-001$	8	$4.269848e-002$	$6.715535e-002$
25	$4.950346e-001$	9	$1.123415e-002$	$5.804991e-002$
26	$4.401510e-001$	9	$1.092148e-002$	$4.139510e-002$
27	$3.938462e-001$	9	$1.048434e-002$	$3.463319e-002$
28	$3.601882e-001$	10	$3.210334e-003$	$2.533135e-002$
29	$3.304893e-001$	10	$3.051547e-003$	$1.452817e-002$
30	$3.046598e-001$	10	$2.889043e-003$	$1.754565e-002$
31	$2.814855e-001$	10	$2.713605e-003$	$2.183909e-002$
32	$2.600533e-001$	10	$2.529738e-003$	$2.390682e-002$
33	$2.397411e-001$	10	$2.355331e-003$	$2.384653e-002$
34	$2.201039e-001$	10	$2.195256e-003$	$2.186321e-002$
35	$2.015751e-001$	10	$2.058672e-003$	$2.335307e-002$
36	$1.854003e-001$	10	$1.950568e-003$	$2.423853e-002$
37	$1.708713e-001$	10	$1.869464e-003$	$2.170470e-002$
38	$1.576414e-001$	10	$1.797993e-003$	$2.160147e-002$
39	$1.453774e-001$	10	$1.735107e-003$	$2.256991e-002$
40	$1.340715e-001$	10	$1.632873e-003$	$2.219430e-002$

41	$1.222396e-001$	10	$1.580034e-003$	$2.222070e-002$
42	$1.092141e-001$	10	$1.453238e-003$	$2.587778e-002$
43	$9.400847e-002$	10	$1.282814e-003$	$3.496839e-002$
44	$7.643301e-002$	9	$3.348808e-003$	$5.363697e-002$
45	$5.504397e-002$	9	$2.494263e-003$	$8.450906e-002$
46	$3.396409e-002$	8	$7.117362e-003$	$1.700008e-001$
47	$4.322774e-003$	2	$4.324437e-003$	$4.877323e-001$
48	$1.190457e-003$	8	$1.197434e-003$	$3.129444e-001$
49	$1.511449e-004$	9	$9.985930e-005$	$1.526331e-001$
50	$4.704174e-006$	10	$5.415516e-006$	$4.308057e-002$

UDL

It.No.	$\|F(u)\|$	Inner Its.	Lin Mod Norm	Delta
0	$3.091846e+001$	$*$	$0.000000e+000$	$0.000000e+000$
1	$2.764021e+001$	2	$2.933704e+001$	$1.361213e+001$
2	$2.104708e+001$	0	$1.683324e+001$	$2.722426e+001$
3	$1.938218e+001$	1	$1.881629e+001$	$1.166258e+001$
4	$1.662497e+001$	0	$1.522023e+001$	$1.166258e+001$
5	$1.658421e+001$	0	$1.438599e+001$	$1.166258e+001$
6	$1.371489e+001$	0	$1.382060e+001$	$5.831291e+000$
7	$4.962846e+000$	0	$5.060344e+000$	$1.166258e+001$
8	$4.576362e+000$	1	$4.545801e+000$	$2.332516e+000$
9	$4.127874e+000$	0	$4.022421e+000$	$4.665033e+000$
10	$4.044723e+000$	1	$4.011772e+000$	$1.790231e+000$
11	$3.937830e+000$	0	$3.913905e+000$	$1.790231e+000$
12	$3.832778e+000$	0	$3.720495e+000$	$3.580462e+000$
13	$3.756678e+000$	0	$3.643211e+000$	$3.580462e+000$
14	$3.706677e+000$	0	$3.579540e+000$	$3.580462e+000$
15	$3.650083e+000$	0	$3.535746e+000$	$3.580462e+000$
16	$3.623536e+000$	0	$3.472826e+000$	$3.580462e+000$
17	$3.551288e+000$	1	$3.502362e+000$	$1.776183e+000$
18	$3.477269e+000$	0	$3.415086e+000$	$1.776183e+000$
19	$3.409543e+000$	0	$3.337162e+000$	$1.776183e+000$
20	$3.349624e+000$	0	$3.277243e+000$	$1.776183e+000$
21	$3.284911e+000$	0	$3.232341e+000$	$1.776183e+000$
22	$3.208065e+000$	0	$3.172701e+000$	$1.776183e+000$
23	$3.118809e+000$	0	$3.098570e+000$	$1.776183e+000$
24	$2.921247e+000$	0	$2.884044e+000$	$3.552366e+000$
25	$2.782435e+000$	1	$2.766856e+000$	$1.823525e+000$
26	$2.622870e+000$	0	$2.463935e+000$	$3.647051e+000$
27	$2.581154e+000$	0	$2.325877e+000$	$3.647051e+000$
28	$2.387799e+000$	0	$2.306067e+000$	$3.647051e+000$
29	$2.133636e+000$	0	$2.016012e+000$	$3.647051e+000$
30	$1.990451e+000$	0	$1.643397e+000$	$3.647051e+000$
31	$1.714828e+000$	0	$1.524255e+000$	$3.647051e+000$
32	$1.433184e+000$	0	$1.213893e+000$	$3.647051e+000$
33	$1.366379e+000$	0	$7.124472e-001$	$3.647051e+000$
34	$8.629856e-001$	0	$7.511402e-001$	$1.823525e+000$
35	$5.383084e-001$	0	$1.861299e-002$	$3.647051e+000$
36	$1.344870e-001$	0	$9.540371e-015$	$3.647051e+000$
37	$7.786228e-002$	1	$6.388109e-002$	$9.282379e-001$
38	$3.788340e-002$	0	$2.598689e-015$	$1.856476e+000$
39	$9.294039e-005$	0	$2.597984e-016$	$1.856476e+000$
40	$5.494895e-009$	0	$1.083045e-018$	$1.856476e+000$

C.5 Driven Cavity Problem Results

NMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.387685e+002$	$*$	$0.000000e+000$	$*$
1	$5.735042e+000$	$*$	$3.420885e-012$	$*$
2	$1.890031e+001$	$*$	$7.398905e-014$	$*$
3	$3.458473e+002$	$*$	$3.043702e-013$	$*$
4	$8.033163e+001$	$*$	$1.812890e-011$	$*$
5	$1.336554e+001$	$*$	$3.041099e-012$	$*$
6	$2.467242e+001$	$*$	$2.197123e-013$	$*$
7	$2.105063e+001$	$*$	$7.524389e-013$	$*$
8	$4.786853e+002$	$*$	$1.942377e-012$	$*$
9	$1.962352e+002$	$*$	$3.414722e-010$	$*$
10	$1.305153e+004$	$*$	$2.564630e-010$	$*$
11	$3.389541e+003$	$*$	$2.129192e-008$	$*$
12	$4.610198e+003$	$*$	$2.328501e-009$	$*$
13	$1.398976e+004$	$*$	$3.194526e-008$	$*$
14	$1.342179e+005$	$*$	$1.560845e-008$	$*$
15	$4.072540e+004$	$*$	$8.236508e-008$	$*$
16	$4.355165e+004$	$*$	$1.762816e-008$	$*$
17	$4.971663e+004$	$*$	$3.584997e-008$	$*$
18	$2.388784e+006$	$*$	$8.283214e-008$	$*$
19	$6.077074e+005$	$*$	$4.291903e-005$	$*$
20	$1.536134e+005$	$*$	$7.614130e-007$	$*$
21	$2.557843e+005$	$*$	$1.608078e-007$	$*$
22	$2.627508e+006$	$*$	$2.900592e-007$	$*$
23	$1.434572e+006$	$*$	$4.251235e-006$	$*$
24	$3.601629e+005$	$*$	$1.419868e-006$	$*$
25	$6.004673e+005$	$*$	$1.173052e-006$	$*$
26	$4.591598e+005$	$*$	$1.068986e-006$	$*$
27	$1.201716e+005$	$*$	$4.266338e-007$	$*$
28	$3.519959e+006$	$*$	$3.517148e-007$	$*$
29	$8.830685e+005$	$*$	$5.416109e-006$	$*$
30	$3.382755e+005$	$*$	$6.210765e-007$	$*$
31	$2.362705e+006$	$*$	$1.467478e-006$	$*$
32	$4.716954e+005$	$*$	$2.572730e-006$	$*$
33	$4.901488e+006$	$*$	$7.081972e-007$	$*$
34	$3.151988e+006$	$*$	$1.635201e-004$	$*$
35	$4.982104e+005$	$*$	$2.406733e-006$	$*$
36	$1.190161e+006$	$*$	$1.191829e-006$	$*$
37	$4.392424e+006$	$*$	$2.236398e-005$	$*$
38	$1.006434e+006$	$*$	$4.174383e-006$	$*$
39	$1.245511e+006$	$*$	$1.647124e-006$	$*$
40	$3.505497e+005$	$*$	$7.244413e-007$	$*$

41	$1.515556e+005$	$*$	$1.028799e-006$	$*$
42	$2.333710e+005$	$*$	$4.176354e-007$	$*$
43	$2.785513e+005$	$*$	$8.325891e-007$	$*$
44	$4.777316e+004$	$*$	$1.781415e-007$	$*$
45	$9.024131e+004$	$*$	$4.171433e-008$	$*$
46	$1.331631e+006$	$*$	$3.132439e-007$	$*$
47	$2.955171e+005$	$*$	$4.152269e-006$	$*$
48	$7.854147e+004$	$*$	$5.566599e-007$	$*$
49	$3.272784e+005$	$*$	$9.704871e-008$	$*$
50	$1.259190e+006$	$*$	$4.591104e-007$	$*$
51	$3.763987e+006$	$*$	$6.825969e-006$	$*$
52	$9.754400e+005$	$*$	$2.413376e-005$	$*$
53	$1.763633e+006$	$*$	$2.974108e-006$	$*$
54	$5.675826e+006$	$*$	$6.444497e-006$	$*$
55	$2.200165e+008$	$*$	$3.366319e-005$	$*$
56	$5.380490e+007$	$*$	$4.667522e-003$	$*$
57	$1.871654e+006$	$*$	$6.290558e-005$	$*$
58	$3.282340e+007$	$*$	$4.993381e-006$	$*$
59	$4.013268e+008$	$*$	$6.476571e-004$	$*$
60	$1.960256e+008$	$*$	$7.260701e-004$	$*$
61	$3.854913e+007$	$*$	$2.406765e-003$	$*$
62	$2.079025e+009$	$*$	$6.637136e-004$	$*$
63	$3.871414e+009$	$*$	$3.215380e-001$	$*$
64	$3.525188e+008$	$*$	$6.157751e-001$	$*$
65	$5.643942e+007$	$*$	$1.351356e-001$	$*$
66	$6.873242e+007$	$*$	$1.824182e-002$	$*$
67	$1.139604e+009$	$*$	$1.532328e-002$	$*$
68	$2.144876e+008$	$*$	$3.442005e-002$	$*$
69	$2.117786e+009$	$*$	$1.981407e-002$	$*$
70	$2.528186e+009$	$*$	$3.424675e-001$	$*$
71	$2.823372e+009$	$*$	$2.272224e+000$	$*$
72	$8.248478e+008$	$*$	$2.947733e-001$	$*$
73	$4.844350e+009$	$*$	$4.508287e-001$	$*$
74	$2.263209e+010$	$*$	$7.982287e-001$	$*$
75	$7.256637e+010$	$*$	$1.000544e+001$	$*$
76	$8.348075e+009$	$*$	$1.404106e+001$	$*$
77	$8.432257e+009$	$*$	$2.815097e+001$	$*$
78	$1.946692e+010$	$*$	$6.677823e+000$	$*$
79	$2.239660e+010$	$*$	$8.095233e+000$	$*$

80	$1.999418e+010$	$*$	$1.183989e+001$	$*$
81	$1.317254e+010$	$*$	$1.157816e+001$	$*$
82	$5.228014e+010$	$*$	$2.656943e+002$	$*$
83	$1.224750e+010$	$*$	$1.000274e+002$	$*$
84	$6.358155e+009$	$*$	$1.370071e+004$	$*$
85	$2.151021e+010$	$*$	$1.532451e+003$	$*$
86	$1.258908e+010$	$*$	$2.253361e+004$	$*$
87	$1.762842e+010$	$*$	$5.158967e+002$	$*$
88	$4.604092e+009$	$*$	$6.402259e+003$	$*$
89	$7.586823e+008$	$*$	$5.424306e+001$	$*$
90	$2.281775e+012$	$*$	$6.921404e+000$	$*$
91	$3.363084e+011$	$*$	$4.578115e+003$	$*$
92	$3.237860e+011$	$*$	$1.702454e+002$	$*$
93	$7.828934e+012$	$*$	$9.129520e+001$	$*$
94	$7.233118e+013$	$*$	$2.487613e+002$	$*$
95	$3.441716e+012$	$*$	$4.657232e+003$	$*$
96	$2.711090e+012$	$*$	$1.230535e+002$	$*$
97	$1.225534e+013$	$*$	$3.231689e+001$	$*$
98	$2.170111e+011$	$*$	$5.628099e+001$	$*$
99	$1.022720e+012$	$*$	$1.554430e+005$	$*$

INMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.387685e+002$	0	$0.000000e+000$	$9.000000e-001$
1	$1.467934e+001$	1	$5.994659e+000$	$9.000000e-001$
2	$5.088052e+000$	2	$5.269833e+000$	$8.432626e-001$
3	$1.948880e+000$	3	$1.468477e+000$	$7.589363e-001$
4	$7.953244e-001$	4	$7.998205e-001$	$6.399826e-001$
5	$1.434503e+000$	14	$3.058027e-001$	$4.857060e-001$
6	$1.247216e+000$	1	$1.247240e+000$	$9.000000e-001$
7	$9.247471e-001$	2	$9.245476e-001$	$8.432626e-001$
8	$6.942710e-001$	8	$6.325317e-001$	$7.589363e-001$
9	$1.301061e+000$	16	$4.254660e-001$	$6.399826e-001$
10	$8.748019e-001$	2	$8.741979e-001$	$9.000000e-001$
11	$7.107287e-001$	3	$7.099932e-001$	$8.432626e-001$
12	$4.611341e-001$	6	$4.579713e-001$	$7.589363e-001$
13	$2.914770e-001$	9	$2.869586e-001$	$6.399826e-001$
14	$7.410431e-001$	18	$1.408562e-001$	$4.857060e-001$
15	$4.944766e-001$	1	$4.946433e-001$	$9.000000e-001$
16	$3.707567e-001$	4	$3.687394e-001$	$8.432626e-001$
17	$2.769892e-001$	5	$2.758771e-001$	$7.589363e-001$
18	$1.588865e-001$	7	$1.574060e-001$	$6.399826e-001$
19	$7.605753e-002$	15	$7.170555e-002$	$4.857060e-001$
20	$6.621468e-001$	30	$2.202318e-002$	$3.108434e-001$
21	$4.620741e-001$	1	$4.621615e-001$	$9.000000e-001$
22	$3.595404e-001$	4	$3.590442e-001$	$8.432626e-001$
23	$2.433560e-001$	5	$2.410920e-001$	$7.589363e-001$
24	$8.648880e-002$	6	$7.541220e-002$	$6.399826e-001$
25	$4.120744e-002$	8	$4.125833e-002$	$4.857060e-001$
26	$1.291785e-002$	21	$1.189594e-002$	$3.108434e-001$
27	$9.360790e-002$	73	$1.927928e-003$	$1.509785e-001$
28	$8.234608e-002$	2	$8.234292e-002$	$9.000000e-001$
29	$6.385121e-002$	4	$6.384078e-002$	$8.432626e-001$
30	$4.816169e-002$	5	$4.812405e-002$	$7.589363e-001$
31	$2.507515e-002$	6	$2.502919e-002$	$6.399826e-001$
32	$1.202238e-002$	11	$1.198733e-002$	$4.857060e-001$
33	$3.633745e-003$	23	$3.564754e-003$	$3.108434e-001$
34	$6.469801e-004$	35	$5.265970e-004$	$1.509785e-001$
35	$2.643337e-005$	79	$2.096590e-005$	$3.312919e-002$
36	$2.198351e-007$	180	$2.070378e-007$	$8.450742e-003$
37	$1.064836e-010$	216	$1.055586e-010$	$4.841336e-004$

QINMU

It.No.	$\|F(u)\|$	GMRES Its.	Lin Mod Norm	Eta
0	$1.387685e+002$	0	$0.000000e+000$	$9.000000e-001$
1	$1.467934e+001$	1	$5.994659e+000$	$9.000000e-001$
2	$5.088052e+000$	2	$5.269833e+000$	$8.432626e-001$
3	$1.948880e+000$	3	$1.468477e+000$	$7.589363e-001$
4	$7.953244e-001$	4	$7.998205e-001$	$6.399826e-001$
5	$6.605522e-001$	14	$3.058027e-001$	$4.857060e-001$
6	$6.032139e-001$	18	$1.973353e-001$	$3.108434e-001$
7	$5.680207e-001$	27	$6.931307e-002$	$1.509785e-001$
8	$5.263397e-001$	31	$4.349660e-002$	$9.197776e-002$
9	$4.879530e-001$	42	$1.670446e-002$	$3.423433e-002$
10	$4.474891e-001$	44	$1.367753e-002$	$3.352345e-002$
11	$4.087519e-001$	49	$7.570913e-003$	$1.865651e-002$
12	$3.545747e-001$	83	$5.741131e-003$	$1.426479e-002$
13	$2.438655e-001$	47	$1.004034e-002$	$3.082684e-002$
14	$6.845999e-002$	23	$4.460263e-002$	$1.865261e-001$
15	$2.875819e-002$	32	$1.189185e-002$	$1.744629e-001$
16	$1.037286e-002$	14	$1.035513e-002$	$3.694218e-001$
17	$9.414096e-003$	45	$2.068917e-003$	$1.996350e-001$
18	$8.535869e-003$	191	$4.526575e-005$	$4.857569e-003$
19	$7.744956e-003$	178	$7.009349e-005$	$8.334626e-003$
20	$7.027491e-003$	178	$5.989326e-005$	$8.281517e-003$
21	$6.233268e-003$	178	$5.124942e-005$	$7.983658e-003$
22	$5.294172e-003$	146	$7.815834e-005$	$1.256434e-002$
23	$4.137579e-003$	136	$1.311179e-004$	$2.516318e-002$
24	$3.948792e-003$	76	$2.524480e-004$	$6.500904e-002$
25	$3.002401e-003$	1	$3.002403e-003$	$9.000000e-001$
26	$2.379332e-003$	4	$2.379320e-003$	$8.432626e-001$
27	$1.508945e-003$	5	$1.508858e-003$	$7.589363e-001$
28	$6.849396e-004$	6	$6.848557e-004$	$6.399826e-001$
29	$3.098211e-004$	12	$3.097975e-004$	$4.857060e-001$
30	$9.599139e-005$	48	$9.574599e-005$	$3.108434e-001$
31	$1.438772e-005$	73	$1.406039e-005$	$1.509785e-001$
32	$1.742670e-007$	148	$1.724026e-007$	$1.271229e-002$
33	$1.001902e-010$	292	$1.003638e-010$	$5.893335e-004$

UDL

It.No.	$\|F(u)\|$	Inner Its.	Lin Mod Norm	Delta
0	$1.387685e+002$	$*$	$0.000000e+000$	$0.000000e+000$
1	$5.735042e+000$	0	$3.420885e-012$	$1.379772e+000$
2	$3.019314e+000$	1	$3.083392e+000$	$2.254444e-001$
3	$1.489764e+000$	0	$1.464496e+000$	$4.508888e-001$
4	$1.058307e+000$	0	$7.328333e-014$	$9.017777e-001$
5	$9.491956e-001$	0	$5.173882e-014$	$9.017777e-001$
6	$7.927133e-001$	1	$8.040361e-001$	$9.017777e-002$
7	$5.090053e-001$	0	$5.086269e-001$	$1.803555e-001$
8	$1.779627e-001$	0	$2.632293e-014$	$3.607111e-001$
9	$2.862144e-002$	0	$5.412018e-015$	$3.607111e-001$
10	$1.309040e-003$	0	$2.166697e-015$	$3.607111e-001$
11	$8.191130e-007$	0	$1.646266e-017$	$3.607111e-001$
12	$5.541761e-013$	0	$1.104148e-020$	$3.607111e-001$

Bibliography

[1] T. F. Chan. Newton-like pseudo-arclength methods for computing simple turning points. *SIAM J. Sci. Stat. Comput.*, 5:135–148, 1984.

[2] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[3] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[4] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[5] S. C. Eisenstat and H. F. Walker. Globally convergent inexact Newton methods. *SIAM J. Optimization*, 4:393–422, 1994.

[6] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput*, 17:16–32, 1996.

[7] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-stokes equations and a multigrid method. *J. Comput. Phys.*, 48:387–411, 1982.

[8] R. Glowinski, H. B. Keller, and L. Reinhart. Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems. *SIAM J. Sci. Statist. Comput.*, 6(4):793–832, 1985.

[9] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[10] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. Nonstiff problems.

[11] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996. Stiff and differential-algebraic problems.

[12] M. EL Hallabi and R. A. Tapia. A global convergence theory for arbitrary norm trust-region methods for nonlinear equations. Technical Report TR87-25, Department of Mathematical Schinces, Rice University, Houston, TX, 1987, Revised 1989.

[13] R. Glowinski H. B. Keller and L. Reinhart. Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems. *SIAM J. Sci. Stat. Comput.*, 6:793–832, 1985.

[14] Y. A. Kuznetsov. *Elements of Applied Bifurcation Theory.* Applied Mathematical Sciences. Springer, second edition, 1995.

[15] R. Lefever and G. Nicolis. Chemical instabilities and sustained oscillations. *J. Theor. Biol.*, 30:267–284, 1971.

[16] Yuri Levin and Adi Ben-Israel. A Newton method for systems of m equations in n variables. In *Proceedings of the Third World Congress of Nonlinear Analysts, Part 3 (Catania, 2000)*, volume 47, pages 1961–1971, 2001.

[17] K. Lust, D. Roose, A. Spence, and A. Champneys. An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.*, 19:1188–1209, 1998.

[18] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. Statist. Comput.*, 4:553–572, 1983.

[19] Jorge J. Moré. A collection of nonlinear model problems. In *Computational Solution of Nonlinear Systems of Equations (Fort Collins, CO, 1988)*, volume 26 of *Lectures in Appl. Math.*, pages 723–762. Amer. Math. Soc., Providence, RI, 1990.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer-Verlag, New York, NY, 1999.

[21] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970. Reissued in 2000 by SIAM Publications, Philadelphia, as Vol. 30 in the series Classics in Applied Mathematics.

[22] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker. Globalization techniques for Newton–Krylov methods and applications to the fully-coupled solution of the Navier–Stokes equations. Technical Report Sand2004-1777, Sandia National Laboratories, Albuquerque NM, 87185, Dec. 2003.

[23] R. P. Pawlowski, J. P. Simonis, H. F. Walker, and J. N. Shadid. Inexact newton dogleg methods. Technical Report MS-5-05-36, Worcester Polytechnical Institute, Worcester MA, 01609, May 2005.

[24] M. J. D. Powell. A FORTRAN subroutine for unconstrained minimization, requiring first derivatives of the objective function. Technical Report AERE-R. 6469, Mathematics Branch, A.E.R.E. Harwell, Berkshire, England, 1970.

[25] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–114. Gordon and Breach, London, 1970.

[26] M. J. D. Powell. General algorithms for discrete nonlinear approximation calculations. In C. K. Chui, L. L. Schumaker, and J. D. Ward, editors, *Approximation Theory IV*, pages 187–218. Academic Press, New York, 1983.

[27] I. Prigogine and R. Lefever. Symmetry breaking instabilities in dissipative systems. *J. Chem. Phys.*, 48(4):1695–1700, 1968.

[28] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, Manchester, 1992.

[29] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[30] R. Schreiber and H. B. Keller. Driven cavity flows by efficient numerical techniques. *SIAM J. Comput. Phys.*, 49:310–333, 1983.

[31] A. H. Sherman. On Newton-iterative methods for the solution of systems of nonlinear equations. *SIAM J. Numer. Anal.*, 15(4):755–771, 1978.

[32] G. A. Shultz, R. B. Schnabel, and R. H. Byrd. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM J. Numer. Anal.*, 22:47–67, 1985.

[33] H. F. Walker. An adaptation of Krylov subspace methods to path following problems. *SIAM J. Sci. Comput.*, 21(3):1191–1198 (electronic), 1999.

[34] H. F. Walker. Numerical methods for nonlinear equations. Technical Report MS-03-02-18, Worcester Polytechnical Institute, Worcester MA, 01609, May 2002.

[35] H. F. Walker and L. T. Watson. Least-change secant update methods for underdetermined systems. *SIAM J. Numer. Math*, 27:1227–1262, 1990.