Towards a New Implementation of a Nonlinear Solver for FEMLAB®

Michael Hanke
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

March 22, 2000

Contents

1 Introduction

Partial differential equations form the mathematical foundation for a host of important areas in engineering and physics. FEMLAB®[1] provides a powerful interactive environment for modeling and solving scientific and engineering problems which base on partial differential equations. Using FEMLAB®one can model strongly nonlinear coupled multi-physics applications with ease. There is no inherent limitation on the simultaneous simulation of many physical phenomena. The present version (the latest release 1.1 dates from October 1999) can handle (systems

[1] FEMLAB® is a registered trade mark of COMSOL AB.

of) second order partial differential equations in one and two space dimensions. The underlying discretization scheme is a finite element method based on piecewise linear triangular elements (and piecewise linear elements in one dimension, respectively).

The FEMLAB® system is implemented within the MATLAB®[2] environment. The latter is an interactive integrated technical computing environment that combines numerical computation, advanced graphics and visualization, and a high-level programming language. Its open design makes MATLAB® easily extensible, and these extensions are platform independent.

With the growing feedback from users of FEMLAB®, there were repeatedly met highly nonlinear applications (e.g., semiconductor device design, non-Newtonian flows) where the implemented nonlinear solver could not provide a converged solution. Therefore, the need for a new implementation arose which should be capable of handling highly nonlinear problems. The present report studies different algorithmic approaches. Section 2 provides an overview about different approaches for a robust and efficient realization of the Newton method for solving nonlinear systems of equations. Section 3 gives some implementational details and design decisions. Finally, in Section 4, we report about the performance of different algorithms for a test set of highly nonlinear problems whose selection was biased towards the final goal of implementing a solver within the FEMLAB® environment.

2 Methods for Highly Nonlinear Problems

The ultimate aim of the new design of the nonlinear solver is the ability to handle rather general, highly nonlinear problems, of course, motivated by finite element discretizations of partial differential equations. Therefore, there is no other possibility than using a variant of the Newton method.

2.1 The Basic Method: The Newton Method

Consider the following problem (P):

$$\begin{aligned} &\text{given:} && F : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n \\ &&& x^0 \in D \\ &\text{find:} && x^* \in D \text{ such that } F(x^*) = 0. \end{aligned} \tag{P}$$

Here, F is a given nonlinear function defined on an open domain $D \subseteq \mathbb{R}^n$. We assume that F is continuously differentiable on D. Moreover, we will assume that x^* is a regular solution to (P), i.e., $F'(x^*)$ is nonsingular. In order to solve (P), we start from the Taylor expansion of F:

$$0 = F(x^*) = F(x^0) + F'(x^0)(x^* - x^0) + o(\|x^* - x^0\|).$$

By neglecting all nonlinear terms we obtain a substitute problem: Find an x such that $G(x) = 0$ where

$$G(x) = F(x^0) + F'(x^0)(x - x^0). \tag{1}$$

[2] MATLAB® is a registered trade mark of The Math Works, Inc.

This gives rise to a new approximation, x^1, to x^*. Carrying out this construction recursively, we arrive at the Newton method

$$\begin{aligned} &\text{given } x^0; \\ &\text{for } k = 1, 2, \ldots \\ &\qquad \text{solve } F'(x^k)\Delta x^k = -F(x^k); \\ &\qquad x^{k+1} = x^k + \Delta x^k; \end{aligned} \tag{N}$$

This method has the following well-known properties:

- Local convergence: The sequence of iterates $\{x^k\}$ exists and converges towards x^* if x^0 is sufficiently close to x^*.
- Fast convergence: If the sequence converges, then it converges very fast, namely, quadratically:
$$\|x^{k+1} - x^*\| \leq C\|x^k - x^*\|^2.$$

The set of starting values such that the Newton method converges towards x^* is denoted by the *domain of attraction* of x^*.

The quadratic convergence leads to the following considerations concerning the computational complexity:

1. Assume that x^* belongs to the domain of attraction of x^*. Then the number k of iterations necessary in order to obtain an approximation x^k with a given accuracy ε, i.e., $\|x^k - x^*\| \leq \varepsilon$, is rather small. So the amount of operations is comparable to that of one solution of a linear system of equations. Therefore, such problems are considered to be mildly nonlinear.
2. If x^0 lies outside the domain of attraction of x^*, certain measures must be taken such that, eventually, an iterate x^k is obtained which belongs to the domain of attraction. These measures, known as *globalization methods*, are often by far more expensive in computational costs such that the overall complexity is no longer comparable to a linear solve. Such problems will be called *highly nonlinear problems.*

It should be noted that this classification depends not only on the properties of F but additionally on the quality of the provided initial guess x^0. This is the reason why the latter is included in the formulation of problem (P).

Before proceeding it appears to be necessary to substantiate one of the basic design decisions. When applying (N) within the framework of the solver FEMLAB®, then the bulk of the computational work is in two algorithmic phases:

- Computation of F and F' for a given x^k: This amounts to the assembly process within a finite element discretization. This functionality of FEMLAB®is implemented using MATLAB®m-files.
- Solution of the linear system of equations: MATLAB®provides sparse Gaussian elimination routines which are implemented in C.

Although the complexity of the Gaussian elimination grows much faster than that of the assembly process the former is faster for dimensions up to some 5,000 unknowns because of their different implementation strategies. This seems to be appropriate for medium sized two-dimensional problems. Therefore, the linear systems will be solved using the Gaussian elimination routines provided by MATLAB®.

2.2 Highly Nonlinear Problems: Globalization Strategies

Assume that x^0 does not belong to the domain of attraction of x^*, that is, we have a highly nonlinear problem at hand. The following globalization techniques are in common use:

(a) the steepest descent method,

(b) the Levenberg-Marquardt method;

(c) the damped Newton method;

(d) continuation methods.

The last type of globalization methods will not be considered here.

In the present section we assume that $\|\cdot\|$ denotes the Euclidean vector norm in $\mathbb{R}^n$. The starting point for the first three methods is the requirement that the iterates are approaching x^* successively:

$$\|x^{k+1} - x^*\| < \|x^k - x^*\| \text{ if } x^k \neq x^*.$$

Note that, for $\|x^k - x^*\|$ small enough, the Newton method has this property. Since x^* is not available, a substitute criterion must be introduced. Usually, instead of the norm of the error, the defect is used:

$$T(x) = \frac{1}{2}\|F(x)\|^2 = \frac{1}{2}F(x)^T F(x). \tag{2}$$

This choice is reasonable because $F(x) = 0$ if and only $T(x) = 0$. So our monotonicity requirement is now given by

$$T(x^{k+1}) < T(x^k). \tag{3}$$

Note that this substitution leads additionally to a stopping criterion for the Newton method. The aim is to compute an x^k such that

$$\|x^k - x^*\| \leq \varepsilon.$$

By using the substitute, the stopping criterion reads

$$T(x^k)^{1/2} \leq \varepsilon \tag{4}$$

such that the *defect* is measured instead of the error.

The steepest descent method dates back to CAUCHY [6]: He proposed to minimize (2) at a given iterate x^k along the negative gradient of T:

$$\begin{aligned} p^k &= -\operatorname{grad} T(x^k) = -F'(x^k)^T F(x^k), \\ x^{k+1} &= x^k + s_k p^k \end{aligned}$$

with a suitably chosen step length parameter s_k. It is easy to see that, for all sufficiently small $s > 0$,

$$T(x^k + sp^k) < T(x^k) \tag{5}$$

if $p^k \neq 0$. This leads to the problem of choosing an appropriate parameter s_k. Unfortunately, even if the optimal parameter s_k (i.e. $T(x^k + s_k p^k) = \min_{s>0} T(x^k + s^k)$) is used, only linear convergence is achieved. On the other hand, the Newton direction $p^k = \Delta x^k = -F'(x^k)^{-1}F(x^k)$ is also a descent direction, that is, (5) holds for all sufficiently small $s > 0$. So we are lead to the damped Newton method

$$\begin{aligned} F'(x^k)\Delta x^k &= -F(x^k) \\ x^{k+1} &= x^k + \lambda_k \Delta x^k \end{aligned} \tag{6}$$

where $\lambda \in (0,1]$ is a suitably chosen damping factor. In order to obtain the fast convergence of the Newton method back, $\lambda_k \equiv 1$ should be chosen if this is possible. This idea seems to be introduced by GLEYZAL [11]. Again, we are left with the problem of efficiently determining a parameter λ_k such that (3) holds true.

A different way of constructing a descent direction Δx^k was proposed by LEVENBERG [17] and MARQUARDT [18]. It was observed that a bad initial guess x^0 leads to a bad Newton direction. This is attributed to the fact that the linearization (1) is only reliable in a neighborhood of x^k. The starting point was the Gauß-Newton method: Instead of solving (1) directly, it is tried to minimize T with F replaced by $G(x^k)$: Compute Δx^k as the solution of

$$\|F(x^k) + F'(x^k)\Delta x^k\| \longrightarrow \min! \tag{7}$$

Obviously, Δx^k is nothing else but the Newton direction. Since G is considered to be reliable in a close neighborhood of x^k, the unconditional optimization problem is replaced by a constraint one: Minimize (7) subject to

$$\|\Delta x^k\| \leq \delta_k. \tag{8}$$

The ball $\{x \,|\, \|x - x^k\| \leq \delta_k\}$ around x^k is called the *trust region* while δ_k is the *trust region radius*. The minimization problem can be solved rather easily using the Lagrange multiplier $p = p(\delta_k) \geq 0$:

$$\begin{aligned} (F'(x^k)^T F'(x^k) + pI)\Delta x^k &= -F'(x^k)^T F(x^k), \\ x^{k+1} &= x^k + \Delta x^k. \end{aligned}$$

The descent criterion (3) is now $T(x^k + \Delta x^k(p)) < T(x^k)$ for all sufficiently large p. The limiting behavior of $\Delta x^k(p)$ is rather interesting:

- If the Newton correction is smaller than δ_k, then $p = 0$ and $\Delta x^k(0)$ is identical to the Newton direction.
- If p approaches infinity, then $p\Delta x^k(p) \to -\operatorname{grad} T(x^k)$, which is the direction of steepest descent.

Efficient δ-strategies (or, equivalently, p-strategies) can be developed. In the present context of rather large linear systems, the Levenberg-Marquardt method is to expensive. In order to find a usable p, linear systems with varying coefficient matrices must be solved at each iterate x^k.

A cheaper alternative for this type of methods (so-called model-trust region methods) is to consider $\Delta x^k(0)$ as a method of connecting the Newton direction ($p=0$) with the scaled direction of steepest descent ($p=\infty$) by a continuous path. The smaller δ_k (or, the larger p), the gradient direction is preferred. The idea is to define a path connecting both directions by an easily implementable method. POWELL [24] proposed his "dog-leg" method. The following "double" dog-leg variant was introduced by DENNIS and MEI [14].

The starting point is the quadratic model (7)

$$m(x) = \frac{1}{2}\|F(x^k) + F'(x^k)(x-x^k)\|^2. \tag{9}$$

Let x_c denote the Cauchy point of (9), that is the minimizer of m in the direction of steepest descent. Additionally, let x^+ be the Newton point $x^+ = x^k + \Delta x^k$. One can show that

$$\|x_c - x^k\| \leq \gamma \|\Delta x^k\|$$

with a certain $\gamma \leq 1$. Here, $\gamma = 1$ is only possible if $F'(x^k)$ is orthogonal. This case will be excluded in the following. Then one can show that, for every η with $\gamma < \eta < 1$, it holds:

- Along the piecewise linear curve connecting x^k, x_c, $x^k + \eta\Delta x^k$, and x^+, $m(x)$ is strictly decreasing.
- Along the same curve, $\|x - x^k\|$ is strictly increasing.

So the trust region method consists of determining a point x^{k+1} on the curve above such that $\|x^{k+1} - x^k\| = \delta_k$ if $\|\Delta x^k\| \geq \delta_k$. Otherwise, the Newton iterate $x^{k+1} = x^+$ is taken. Note that the number of operations has been drastically reduced compared to the Levenberg-Marquardt approach.

Let

$$\lambda_* = \frac{\|F'(x^k)^T F(x^k)\|^2}{\|F'(x^k)F'(x^k)^T F(x^k)\|^2}, \qquad \gamma = \frac{\lambda_*}{\|F(x^k)\|^2}.$$

Then one computes easily that

$$x_c = x^k - \lambda_* F'(x^k)^T F(x^k).$$

Choose now an $\eta \in (\gamma, 1)$. Then the new iterate is given by $x^{k+1} = x^k + s^k$ where

$$s^k = \begin{cases} \Delta x^k & \text{if } \|\Delta x^k\| \leq \delta_k, \\ \delta_k \frac{\Delta x^k}{\|\Delta x^k\|} & \text{if } \eta\|\Delta x^k\| \leq \delta_k < \|\Delta x^k\|, \\ -\lambda_* F'(x^k)^T F(x^k) + \bar{\lambda}(F'(x^k)^T F(x^k) + \eta\Delta x^k) & \text{if } \|\lambda_* F'(x^k)^T F(x^k)\| < \delta_k < \|\eta\Delta x^k\|, \\ -\delta_k \frac{F'(x^k)^T F(x^k)}{\|F'(x^k)^T F(x^k)\|} & \text{if } \delta_k \leq \|\lambda_* F'(x^k)^T F(x^k)\|. \end{cases} \tag{10}$$

The value of $\bar{\lambda}$ can be determined by solving the quadratic equation

$$\| - \lambda_* F'(x^k)^T F(x^k) + \lambda(F'(x^k)^T F(x^k) + \eta \Delta x^k)\|^2 = \delta_k^2.$$

All the methods described so far use (3) and the definition (2) immediately. However, there are two main objections against (3):

(i) The error of x^k is measured in the wrong space, namely, via the defect.

(ii) If the equation $F(x) = 0$ is scaled by a nonsingular matrix A such that the new system reads $AF(x) = 0$, then the Newton method provides exactly the same sequence of iterates as the unscaled version. But (2) is now defined according to

$$T(x|A) = \frac{1}{2}\|AF(x)\|^2 \tag{11}$$

which can change dramatically.

The property of the Newton method to be invariant under linear scalings is called the *affine invariance* of the Newton method. This is why DEUFLHARD [9] proposed to *require* that any globalization method should be formulated in affine invariant quantities. This approach strongly suggests to use the damped Newton method. As it turns out, the affine invariant formulation of the Levenberg-Marquardt method (7) – (8) is just

$$\|F'(x^k)^{-1}F(x^k) + \Delta x^k\| \longrightarrow \min!$$

subject to the constraint

$$\|\Delta x^k\| \leq \delta_k.$$

If the constraint is not active, Δx^k is just the ordinary Newton correction. Otherwise, $\Delta x^k(\delta_k) = \lambda(\delta_k)\Delta x^k$ is just a damped Newton step. So we obtain once more the damped Newton method (6).

For a given nonsingular matrix A, let

$$G(x|A) := \{z \in D | T(z|A) \leq T(x|A)\}$$

be the level set of (11). The descent condition (3) for (11) is

$$T(x^{k+1}|A) < T(x^k|A). \tag{12}$$

It is fulfilled if x^{k+1} is an inner point of $G(x^k|A)$. Therefore, it is natural to consider

$$\bar{G}(x) := \bigcap_{A \in GL(n)} G(x|A).$$

DEUFLHARD was able to show that, under certain rather general conditions, $\bar{G}(x^0)$ is a topological path $\bar{x}: [0,1] \to D$ called the *Newton path* which has the following properties:

(a) $F(\bar{x}(\lambda)) = (1-\lambda)F(x^0)$,
$T(\bar{x}(\lambda)|A) = (1-\lambda)^2 T(x^0|A)$.
All level functions decrease along $\bar{x}$, $\lambda \in [0,1]$. This property is independent of the special choice of A.

(b) $\dfrac{d\bar{x}}{d\lambda} = -F'(\bar{x})^{-1}F(x^0)$,
$\bar{x}(0) = x^0, \quad \bar{x}(1) = x^*$.
One could follow this path by numerically integrating the differential equation. But the latter problem can be extremely ill-conditioned. So this possibility is no real alternative.

(c) $\frac{d\bar{x}}{d\lambda}|_{\lambda=0} = -F'(x^0)^{-1}F(x^0) = \Delta x^0$
The tangent direction to $\bar{x}$ at $\lambda = 0$ is just the Newton direction. So this direction is outstanding even far away from the solution x^*. In view of (b), its length is to large. So we are once again naturally led to use a damped Newton method (6).

Let us note in passing that the equation $H(x,\lambda) = F(x) - (1-\lambda)F(x^0) = 0$ is often called the defect reducing homotopy for solving (P). In this interpretation, (b) represents the Davidenko differential equation belonging to H ([7], see also [1]). The use of this information gives rise to so-called *continuation methods.* Such methods will be implemented in future releases of FEMLAB®.

Up to now we do not known which A should be chosen in the damped Newton method. Taking the affine invariance principle as the guiding idea, we are led to

$$A = F'(x^k)^{-1} \tag{13}$$

since $T(\cdot|F'(x^k)^{-1})$ is affine invariant for a given x^k. DEUFLHARD calls this choice the *natural monotonicity criterion* because of the following outstanding features:

(i) The Newton direction Δx^k is simultaneously the direction of steepest descent for $T(\cdot|F'(x^k)^{-1})$.

(ii) If F is twice continuously differentiable,

$$T(x|F'(x^*)^{-1}) = \frac{1}{2}\|x-x^*\|^2 + o(\|x-x^*\|^2)$$

such that $T(x^{k+1}|F'(x^k)^{-1})$ provides an error estimation for x^k.

(iii) The damping factor λ can be chosen larger than for other choices of A.

Note that the matrix A is no longer constant but changes with k. This has the consequence that convergence cannot be guaranteed theoretically any longer! A pathological counterexample is provided in [3]. Nevertheless, practical evidence, especially in the context of the numerical solution of two-point boundary value problems, has shown that the natural monotonicity function

yield rather robust nonlinear solvers. An explanation can be found in the cited paper [3]. One has to pay for the nice properties also algorithmically. It holds

$$T(x^k|F'(x^k)^{-1} = \frac{1}{2}\|F'(x^k)^{-1}F(x^k)\|^2 = \frac{1}{2}\|\Delta x^k\|^2,$$
$$T(x^{k+1}|F'(x^k)^{-1}) = \frac{1}{2}\|F'(x^k)^{-1}F(x^{k+1})\|^2 = \frac{1}{2}\|\overline{\Delta x}^{k+1}\|^2$$

where Δx^k denotes the ordinary Newton correction while $\overline{\Delta x}^{k+1}$ is the modified Newton correction for the $(k+1)$-st step. Assuming that an LU-decomposition of $F'(x^k)$ is available from the computation of the Newton correction, the additional cost is that of one back-substitution. In the present context this should be negligible. This is no longer true if the linear system is solved by other means, e.g., iterative methods.

2.3 Step Selection Strategies

The performance of all globalization methods depends essentially on a carefully selected step length parameter. A correct choice of this parameter decides not only about the efficiency but also about the success of a code. A step length selection strategy contains two phases: A *prediction strategy* for estimating an initial guess, and a *correction device* for adjusting the parameter if the monotonicity criterion is not fulfilled.

A simple but rather efficient algorithm was proposed by ARMIJO [2]: Choose the parameter λ_k such that

$$\lambda_k \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}, \tag{14}$$
$$T(x^k + \lambda_k \Delta x^k) \leq \left(1 - \frac{\lambda_k}{2}\right) T(x^k). \tag{15}$$

Obviously, this strategy can also be applied in the natural monotonicity criterion $T(\cdot|A)$. In (14), the largest possible λ_k should be chosen in order to obtain the undamped Newton method as soon as possible. So the obvious prediction strategy is to choose $\lambda_k^{(0)} = 1$ as the first trial value and, if (15) is not fulfilled, to retry using $\lambda_k^{(j+1)} = 0.5\lambda_k^{(j)}$. Since it is highly probable that a small λ_{k-1} leads to a small λ_k during the next step, a more conservative prediction is $\lambda_k^{(0)} = 2\lambda_{k-1}$.

For the affine invariant Newton method, more refined damping strategies are possible [9, 10, 4]. An optimal prediction strategy is given by

$$\begin{aligned} \lambda_k^{(0)} &= \min\{1, \mu_k^{(0)}\}, \\ \mu_k^{(0)} &= \frac{\|\Delta x^{k-1}\|}{\|\overline{\Delta x}^k - \Delta x^k\|} \times \frac{\|\overline{\Delta x}^k\|}{\|\Delta x^k\|} \times \lambda_{k-1}. \end{aligned} \tag{16}$$

Since $\lambda_k^{(0)}$ is defined recursively, an initial guess $\lambda_0^{(0)}$ must be provided. If the monotonicity test

fails, a reduction strategy

$$\begin{aligned}\lambda_k^{(j+1)} &= \min\{\frac{\lambda_k^{(j)}}{2}, \mu_k^{(j)}\} \\ \mu_k^{(j)} &= \frac{\|\Delta x^k\|}{\|\overline{\Delta x}^{k+1}(\lambda_k^{(j)}) - (1-\lambda_k^{(j)})\Delta x^k\|} \times \frac{(\lambda_k^{(j)})^2}{2}\end{aligned} \tag{17}$$

is invoked. Practical experience shows that the reduction step is, in many cases, executed only once such that it is rather efficient. For very highly nonlinear problems, it is useful to modify the above mentioned strategies by replacing $\mu_k^{(j)}$ by $\mu_k^{(j)}/2$.

Another step length strategy for (3) is the back-tracking line search proposed by DENNIS and SCHNABEL [15]. Here, the function

$$f(\lambda) := T(x^k + \lambda \Delta x^k)$$

is modeled by quadratic and cubic polynomials. The prediction step chooses

$$\lambda_k^{(0)} = 1.$$

If the descent condition

$$T(x^k + \lambda_k^{(j)}) \leq (1 - \alpha\lambda_k^{(j)})T(x^k) \tag{18}$$

for a fixed $0 < \alpha, 1$ is not fulfilled the so-called back-tracking is invoked. For $j = 0$, $f(0), f'(0)$, and $f(1)$ are known after evaluating (18). These two values are interpolated by a quadratic polynomial. The minimum of this polynomial is chosen as the next trial value

$$\lambda_k^{(1)} = \frac{-f'(0)}{2(f(1) - f(0) - f'(0))}. \tag{19}$$

If (18) is not fulfilled for $\lambda_k^{(1)}$, all further reduction steps are computed using a cubic model defined by $f(0), f'(0)$, as well as $f(\lambda_k^{(j-1)})$ and $f(\lambda_k^{(j)})$. The local minimizer of this polynomial is chosen to provide $\lambda_k^{(j+1)}$). It is rather straightforward to obtain the formula

$$\begin{aligned}\lambda_k^{(j+1)} &= \frac{-b + \sqrt{b^2 - 3af'(0)}}{3a} \\ \begin{pmatrix} a \\ b \end{pmatrix} &= \frac{1}{\lambda_k^{(j)} - \lambda_k^{(j-1)}} \begin{pmatrix} \frac{1}{(\lambda_k^{(j)})^2} & \frac{-1}{(\lambda_k^{(j-1)})^2} \\ \frac{-\lambda_k^{(j-1)}}{(\lambda_k^{(j)})^2} & \frac{\lambda_k^{(j)}}{(\lambda_k^{(j-1)})^2} \end{pmatrix} \begin{pmatrix} f(\lambda_k^{(j)}) - f(0) - f'(0)\lambda_k^{(j)} \\ f(\lambda_k^{(j-1)}) - f(0) - f'(0)\lambda_k^{(j-1)} \end{pmatrix}.\end{aligned} \tag{20}$$

In order to obtain a real $\lambda_k^{(j+1)}$, α must be chosen smaller than 1/4.

The step length strategy for the dog-leg method follows the modification of Powell's method according to DENNIS and SCHNABEL [15]. The controlling parameter in trust region methods is the trust region radius δ_k and not a damping parameter λ_k. This difference in the construction

of the method leads also to a difference in the parameter search: First, a parameter δ_k must be selected such that a descent criterion is fulfilled (the former correction strategy), afterwards a prediction for the following step is made. Assume that s^k is computed according to (10) for a given trust region radius. The descent condition reads now

$$T(x^k + s^k) \leq T(x^k) - \alpha F(x^k)^T F'(x^k) s^k.$$

This condition is identical to (18) if $s^k = \Delta x^k$. If it is fulfilled, the new iterate is accepted. If it is not fulfilled, a new trial value is chosen using the quadratic model (19):

$$\delta_k := \lambda_k^{(1)} \delta_k.$$

Note that the cubic interpolation is no longer justified such that (20) is *not* used.

Assume now, that δ_k is accepted. If $s^k = \Delta x^k$ is the Newton step, $\delta_{k+1} = \|\Delta x^k\|$ is selected. Otherwise, it is proposed in [15] to try a larger step starting from x^k first because of a possible reduction of the computational effort. Note that, if a larger step is possible, the basic effort is to compute a new element on the dog-leg curve. To decide about the possibility of doing so, the accuracy of the quadratic model (9) is estimated by comparing the predicted reduction $f_{\text{pred}} = m(x^k + s^k) - T(x^k)$ with the actual reduction $\Delta T = T(x^k + s^k) - T(x^k)$. If either

$$|f_{\text{pred}} - \Delta T| \leq \beta |\Delta T|$$

or we have a region of negative curvature,

$$T(x^k + s^k) \leq T(x^k) - F(x^k)^T F'(x^k) s^k,$$

the current step is retried with $\delta_k := 2\delta_k$.

After these "correction" steps, the resulting $x^{k+1} = x^k + s^k$ is accepted. It remains to predict the trust region radius. Here, the strategy is rather simple:

$$\delta_{k+1} = \begin{cases} 2\delta_k & \text{if } \Delta T \leq f_{\text{pred}}, \\ \delta_k / 2 & \text{if } \Delta T > 0.1 f_{\text{pred}}, \\ \delta_k & \text{otherwise.} \end{cases}$$

2.4 Quasi-Newton Updates

It is usually very costly to compute the Jacobian $F'(x^k)$ and to perform the LU-decomposition. One method to avoid these computations are quasi-Newton methods. Especially, the Broyden updating procedure [5] has been frequently applied. This method has even superlinear convergence. Computational experience shows that this method should only be applied if the undamped Newton method converges, i.e. $\lambda_k \equiv 1$. So the method takes the form

$$\begin{aligned} A_k s^k &= -F(x^k), \\ x^{k+1} &= x^k + s^k. \end{aligned} \tag{21}$$

Obviously, if $A_k = F'(x^k)$, the s^k is the usual Newton correction. After each step, the matrix A_k is updated to yield a new A_{k+1}. According to BROYDENs proposal, A_{k+1} is computed by

$$A_{k+1} = A_k + \frac{(y^k - A_k s^k)(s^k)^T}{(s^k)^T s^k}, \tag{22}$$

$$y^k = F(x^{k+1}) - F(x^k). \tag{23}$$

In this form, the update is not efficient. The updating matrix is a full matrix destroying the sparseness of the stiffness matrix in the finite element method. Although there are proposals for using only those elements of the second term in (22) which refer to nonzero entries in A_k [25] and thus preserving the sparsity, the necessity of an LU-decomposition remains. The updating can also be applied to the LU-decomposition immediately. Nevertheless, the sparsity preserving update of the LU-decomposition show a serious performance degradation of the method according to the author's experience [12]. Therefore, one should try to use (22) – (23) as far as possible.

The following nice idea is due to DEUFLHARD [10] (cf. also [23]). The starting point is the observation that we need to compute only one or two vectors, namely the Broyden correction

$$s^k = -A_k^{-1} F(x^k)$$

and, for the affine invariant Newton method, the modified Broyden update

$$\bar{s}^{k+1} = -A_k^{-1} F(x^{k+1}).$$

Equation (22) can be rewritten by using (21), (23):

$$y^k - A_k s^k = F(x^{k+1}) - F(x^k) + F(x^k) = F(x^{k+1}) = -A_k \bar{s}^{k+1}.$$

Hence, the updating formula reads

$$A_{k+1} = A_k \left(I - \frac{\bar{s}^{k+1}(s^k)^T}{(s^k)^T s^k} \right) \tag{24}$$

or, equivalently,

$$A_{k+1} = A_k - \frac{A_{k+1} s^{k+1}(s^k)^T}{(s^k)^T s^k}.$$

The latter representation leads to

$$A_{k+1} \left(I + \frac{s^{k+1}(s^k)^T}{(s^k)^T s^k} \right) = A_k,$$

which yields

$$A_{k+1}^{-1} = \left(I + \frac{s^{k+1}(s^k)^T}{(s^k)^T s^k} \right) A_k^{-1}. \tag{25}$$

Assume now that, up to a certain index k_0 the standard Newton method is used such that $A_k = F'(x^k)$ for $k \le k_0$. Moreover, $s^{k_0} = \Delta^{k_0}$ and $\bar{s}^{k_0+1} = \overline{\Delta x}^{k_0+1}$ are computed using an LU-decomposition of A_{k_0}. In the following we assume without loss of generality that $k_0 = 0$.

Let $z \in \mathbb{R}^n$ be a given vector. Define $v^j = A_j^{-1} z$. Using (25) we obtain

$$\begin{aligned} A_{k+1}^{-1} z = v^{k+1} &= \left(I + \frac{s^{k+1}(s^k)^T}{(s^k)^T s^k} \right) A_k^{-1} z \\ &= v^k + s^{k+1} \frac{(s^k)^T v^k}{(s^k)^T s^k} \\ &= v^k + \alpha_k(v^k) s^{k+1} \end{aligned} \tag{26}$$

where $\alpha_k(v^k)$ can easily be computed. Unfortunately, s^{k+1} is unknown up to now. Remember that $v = s^{k+1}$ solves the equation $A_{k+1} v = -F(x^{k+1})$. Hence, letting $z = -F(x^{k+1})$ in (26) yields

$$s^{k+1} = v^k + \alpha_k(v^k) s^{k+1}$$

such that s^{k+1} can be computed. This gives rise to the following simple recursion for computing the quasi-Newton correction:

$$\begin{aligned} v^0 &= -A_0^{-1} F(x^{k+1}) && \text{by using the available LU-decomposition} \\ v^{j+1} &= v^j \alpha_j(v^j) s^{j+1} && \text{for } j = 1, \ldots, k-1 \\ s^{k+1} &= \frac{v^k}{1 - \alpha_k(v^k)} \end{aligned}$$

Once s^{k+1} is known, $A_{k+1}^{-1} z$ can be computed for any z if the last equation is replaced by

$$A_{k+1}^{-1} z = v^k + \alpha_k(v^k) s^{k+1}.$$

For a realization of this algorithm, the quasi-Newton corrections s^k must be stored. With the exception of the back-substitution, the computational amount is proportional to $k \cdot n$.

If we intend to use the monotonicity criterion (4), one has to provide $A_{k+1} z$ for a given $z \in \mathbb{R}^n$ additionally. This can be done along the same lines. Equation (24) leads to

$$\begin{aligned} A_{k+1} z &= A_k \left(I - \frac{\bar{s}^{k+1}(s^k)^T}{(s^k)^T s^k} \right) z \\ &= A_k \left(z - \bar{s}^{k+1} \frac{(s^k)^T z}{(s^k)^T s^k} \right) \\ &= A_k (z - \beta_k(z) \bar{s}^{k+1}) \end{aligned}$$

Therefore, $A_{k+1} z$ can be computed by

$$\begin{aligned} v^{k+1} &= z \\ v^j &= v^{j+1} - \beta_j(v^{j+1}) \bar{s}^{j+1} \text{ for } j = k, \ldots, 0 \\ A_{k+1} z &= A_0 v^0 \end{aligned}$$

once $\bar{s}^{k+1}$ is known. This recursion is by far more expensive than that before since both the quasi-Newton correction and the modified quasi-Newton direction must be computed and stored. So the computational expense doubles roughly.

Finally, it is sometimes useful to have a similar algorithm for the computation of $A_{k+1}^T z$. One readily obtains

$$\begin{aligned} v^0 &= A_0^T z \\ v^{j+1} &= v^j - \gamma_j(v^j)s^j \text{ for } j = 0,\ldots,k \\ A_{k+1}^T z &= v^{k+1} \end{aligned}$$

where $\gamma_j(v^j) = \frac{(\bar{s}^{j+1})^T v^j}{(s^j)^T s^j}$. Again, the computational complexity amounts to doubly that of the first algorithm.

3 Model Implementations

In order to get an estimation about the behavior of the different methods we realized the damped Newton method with backtracking line-search, the double dog-leg method, and the affine invariant Newton method within the MATLAB®environment in the function m-files `btds`, `trds`, and `dnw`. The design was chosen as close as possible in order to enable a comparison.

The guiding idea within these implementations is completely determined by the final goal of realizing a new nonlinear solver within the FEMLAB®environment:

- The nonlinear solver is situated rather deep within the program hierarchy. This implies that the user cannot supply much information to the solver. Consequently, most of the control must be done by the algorithm, i.e. automatically.
- The starting point for the need of a new implementation was the occurrence of very highly nonlinear problems. Therefore, robustness and reliability are the ultimate goal, possibly with a certain loss of efficiency.
- For simple problems, the loss of efficiency due to the added complexity should be tolerable.

One of the most important issues in the implementation of methods for solving nonlinear equations is that of a correct scaling (see [19, 21, 13]). A given nonlinear equation

$$F(x) = 0$$

can be equivalently written down as

$$\bar{D}^{-1}F(D(D^{-1}x)) = 0$$

with nonsingular diagonal matrices $\bar{D}$ and D. The scaling D of the unknowns has an influence on the criterion (12) with (13) for the affine invariant Newton method as well as on the trust region

method. Similarly, the scaling $\bar{D}$ in the space of residuals controls the step-size selection strategy for the backtracking line-search and the trust region method via (5).

According to our first guideline, the scalings should be chosen automatically. According to the results in [19] (in the case of the Levenberg-Marquardt method), these scalings should be chosen adaptively within the course of iteration, i.e., instead of choosing fixed matrices $\bar{D}$, D one should adapt them according to

$$D_k = D_k(x^k), \bar{D}_k = \bar{D}_k(x^k).$$

Let x^k be given. Then the equation for the $(k+1)$-st iterate reads

$$\bar{D}_k^{-1} F(D_k(D_k^{-1}x)) = 0 \tag{27}$$

with the Jacobian

$$A_k = \bar{D}_k^{-1} F'(x^k) D_k. \tag{28}$$

Consider first the scalings of the unknowns. MORÉ [19] compares different strategies which use the norm of the partial derivative $\frac{\partial}{\partial x_i} F(x^k)$ as a *measure of change of* F with respect to the i-th unknown. Therefore, the i-th diagonal element of D_k is taken to be equal to this norm:

$$\begin{aligned} D_k &= \operatorname{diag}(d_1^{(k)}, \ldots, d_n^{(k)}), \\ d_i^{(0)} &= \| \frac{\partial}{\partial x_i} F(x^0) \|^{-1}, \qquad i = 1, \ldots, n \\ d_i^{(k)} &= \max\{ d_i^{(k-1)}, \| \frac{\partial}{\partial x_i} F(x^k) \|^{-1} \}, \qquad i = 1, \ldots, n. \end{aligned} \tag{29}$$

Another proposal is used NOWAK and WEIMANN [23]. Their choice is motivated by the error control. Since $\| \cdot \|_D = \| D^{-1} \cdot \|$ is used in the error control, the component-wise relative error is adequately represented by

$$\| D_k^{-1} \Delta x^k \|, D_k = \operatorname{diag}(x_1^*, \ldots, x_n^*).$$

This leads to the following choice:

$$\begin{aligned} D_k &= \operatorname{diag}(d_1^{(k)}, \ldots, d_n^{(k)}), \\ d_i^{(0)} &= \max\{ |x_i^0|, t \}, \qquad i = 1, \ldots, n \\ d_i^{(k)} &= \max\{ \frac{1}{2}(|x_i^{k-1}| + |x_i^k|), t \}, \qquad i = 1, \ldots, n, \end{aligned} \tag{30}$$

where t denotes a given positive threshold value. According to a classification provided by the user,

$$t = \begin{cases} \tau, & \text{if the problem is highly nonlinear,} \\ 1, & \text{if the problem is mildly nonlinear,} \end{cases}$$

with the error tolerance τ imposed by the user. Hence, for highly nonlinear problems, the control is biased towards a relative error.

Although a scaling of the residuals is of a similar importance as the scaling of the unknowns, the author is not aware of any dynamical strategy for choosing $\bar{D}_k$. DENNIS and SCHNABEL [15] propose to use a constant scaling matrix to be provided by the user such that the components of $\bar{D}^{-1}F$ are roughly equilibrated away from the solution. In the present context we have to use an adaptive scaling. In [23], the scaling matrix is computed by

$$\begin{aligned} \bar{D}_k &= \operatorname{diag}(\bar{d}_1^{(k)}, \ldots, \bar{d}_n^{(k)}), \\ \bar{d}_i^{(k)} &= \max_{1 \le j \le n} |(F'(x^k) D_k)_{ij}|, \qquad i = 1, \ldots, n. \end{aligned} \tag{31}$$

This choice seems to aim at a minimal condition number for the matrix A_k in (28). Note that this scaling does not have any influence on the affine invariant Newton method. According to our experience, this choice leads to such drastical changes in the scaling such that the convergence of the other methods is badly influenced. Therefore, we prefer the following choice of $\bar{D}_k$ for the backtracking damped Newton method and the double dog-leg method:

$$\begin{aligned} \bar{D}_k &= \operatorname{diag}(\bar{d}_1^{(k)}, \ldots, \bar{d}_n^{(k)}), \\ \bar{d}_i^{(0)} &= \sum_{j=1}^{n} |(F'(x^0))_{ij}|, \qquad i = 1, \ldots, n \\ \bar{d}_i^{(k)} &= \frac{1}{2}(\bar{d}_i^{(k-1)} + \sum_{j=1}^{n} |(F'(x^k))_{ij}|), \qquad i = 1, \ldots, n. \end{aligned} \tag{32}$$

The termination criteria base on the scaled quantities. The affine invariant Newton method is considered to be successful if

$$\|D_k^{-1} \overline{\Delta x}^{k+1}\| \le \tau, \qquad \lambda_k = 1$$

hold true. For the other two methods, a successful return is indicated by a small (quasi-) Newton correction:

$$\|D_k^{-1} \Delta x^k\| \le \tau \text{ and } \lambda_k = 1.$$

This decision was taken in order to make the methods more comparable to each other. Additionally, certain security checks are applied in order to ensure that the recent iterate, x^k, is within the domain of quadratic convergence.

Failure of the program is indicated in two circumstances:

1. A user-provided number of iterations is exceeded;
2. the damping factor λ_k (and the trust region radius δ_k, respectively) fall below a minimal given value.

The algorithm for computing the damping factor and the trust region radius, respectively, are safeguarded against meaningless predictions by assuming that

$$u\lambda_k^{(j)} \le \lambda_k^{(j+1)} \le o\lambda_k^{(j)}$$

and, similarly,

$$u\delta_k^{(j)} \leq \delta_k^{(j+1)} \leq o\delta_k^{(j)}.$$

According to the proposals in [23, 15], it holds

$$u = 0.1, o = 0.5 \text{ (backtracking and trust region methods)}$$
$$u = 0.1, o = 10 \text{ (affine invariant Newton method).}$$

The property that x^k does not belong to the domain of definition of F can be signaled to the algorithms. They react by halving the damping factor until a trial value within the domain of definition is found.

For simple problems, the quasi-Newton methods according to Section 2.4 are implemented. In order to ensure robustness, Broyden updates are only used if (a) the user allows so, (b) at least two successful complete Newton steps hade been performed, and (c) in the case of the affine invariant Newton method, a certain prediction of the expected approximation quality is sufficiently small. The user can also limit the number of successive quasi-Newton steps before the Jacobian is reevaluated. If the damping factor is less than 1, the next step will be carried out using the exact Jacobian.

4 Examples

4.1 A Set of Test Examples

The Bratu Problem. Let $\Omega = [0,1]^2 \subset \mathbb{R}^2$ denote the unit square in $\mathbb{R}^2$. For a given $\lambda \in \mathbb{R}$, we are interested in finding a solution to the problem

$$-\Delta u = \lambda e^u \text{ in } \Omega, \quad u = 0 \text{ on } \Gamma = \partial\Omega.$$

Here, Δ denotes the Laplacian and Γ is the boundary of Ω. This problem is widely used for testing different aspects of numerical methods, cf. [20].

If one has $\lambda \leq 0$, the the left hand sides constitutes a strongly monotone operator such that a large number of numerical methods will work quite well. So the interesting case is $\lambda > 0$. The set of solutions form a curve parameterized by λ. It has a turning point near $\lambda_c = 6.80812\ldots$. For $\lambda < \lambda_c$, there exist exactly two solution while there are no solutions for $\lambda > \lambda_c$.

The partial differential equation is discretized by the usual five-point discretization on a square grid with grid size $h = 1/N$ in both coordinate directions. The initial guess was chosen to be equal to the trivial solution.

The nonlinear problem is mildly nonlinear for values of λ even rather close to λ_c. So we expect even the Broyden method to work quite well.

A Tubular Chemical Reactor Model. The steady state of o non-adiabatic chemical reactor in which there is processed a simple, first-order, exothermic chemical reaction can be described by

the following second order two-point boundary value problem (see [20] and the references cited there):

$$\frac{1}{Pe_m}u'' - u' - Da\, f(u,v;\gamma) = 0, 0 < x < 1$$
$$\frac{1}{Pe_h}v'' - v' - \beta(v - v_0) + BDa\, f(u,v;\gamma) = 0, 0 < x < 1$$

subject to the boundary conditions

$$u' = Pe_m(u-1), v' = Pe_h(v-1) \text{ at } x = 0$$
$$u' = v' = 0 \text{ at } x = 1.$$

Here, $f(u,v;\gamma) = u\exp(\gamma - \gamma/v)$ is the reaction rate derived from Arrhenius kinetics.

This problem is often used in the context of testing path following methods because it has a rather complicated bifurcation diagram depending on the parameters Pe_m, Pe_h (mass and heat Peclet numbers, respectively), Da (the Damköhler number), B (the heat of rection), β (the heat transfer coefficient), and γ (the activation energy). In [16], a number of different parameter constellations are provided:

No	Pe_h	Pe_m	B	β	Da	γ	v_0	n
1	2.5	5	15	0	0.07	20	0	1
2	291.3	291.3	20	0	0.05826	20	0	3
3	4	8	15	2	0.07	20	0	3
4	5	10	15	2	0.07	20	0	5
5	22	44	15	2	0.07	20	0	5
6	50	100	15	2	0.12	20	0	3
7	70	140	15	2	0.123	20	0	3
8	25	50	15	2	0.12	20	0	1

Here, n denotes the number of (known) solutions. For higher values of the Peclet numbers, the solutions exhibit boundary layers. Even for lower Peclet numbers, the problem is often very difficult to solve.

The problem is discretized on an equidistant grid with step size $h = 1/N$ using standard symmetric stencils. Different solution guesses are used in an attempt to compute different sets of solutions.

A Semiconductor Problem. In [8], DEN HEIJER and RHEINBOLDT proposed to use the following two-point boundary value problem:

$$-u'(t) = \lambda c_a e^{\lambda\beta(u_a - u)} - \lambda c_b e^{\lambda\beta(u-u_b)} + d(t), \qquad a < t < b$$
$$d(t) = \begin{cases} -\lambda c_a, & \text{for } t \le 0 \\ \lambda c_b, & \text{for } t > 0 \end{cases}$$

subject to the boundary conditions

$$u(a) = \lambda u_a, \qquad u(b) = \lambda u_b.$$

On a given mesh, the problem is discretized by piecewise linear finite elements.

Particularly interesting and computationally challenging values of the parameters are

$$a = -9 \times 10^{-5}, b = 10^{-5}, u_a = 0, u_b = 700$$
$$\beta = 40, c_a = 10^{12}, c_b = 10^{13}.$$

A simpler set of parameters is given by

$$a = -9 \times 10^{-3}, b = 10^{-3}, u_a = 0, u_b = 25$$
$$\beta = 20, c_a = 10^{6}, c_b = 10^{7}.$$

The parameter λ is introduced artificially. The problem to be solved is obtained for $\lambda = 1$, while it is trivial for $\lambda = 0$. So in [8], continuation methods are proposed. In our context, it is interesting to compare the iterations for different values of λ starting at the trivial solution.

The mesh is given in the following table:

i	0	1	2	3	4	5	6	7	8	9	10
$10^3 t_i$	-9	-7	-5	-4.5	-4.25	-4	-3.75	-3.5	-3	-1	1

Supersonic Transport Air Pollution. This example is a straightforward extension of an instationary 1D model for the pollution of the stratosphere by supersonic transports [26] (cf. also [22]). The equations are given by

$$\begin{aligned}
-D\Delta u_1 &= k_{11} - k_{12}u_1 + k_{13}u_2 + k_{14}u_4 - k_{15}u_1u_2 - k_{16}u_1u_4 \\
-D\Delta u_2 &= k_{21}u_1 - k_{22}u_2 + k_{23}u_1u_2 - k_{24}u_2u_3 \\
-D\Delta u_3 &= -k_{31}u_3 + k_{32}u_4 + k_{33}u_1u_4 - k_{34}u_2u_3 + 800 + S \\
-D\Delta u_4 &= -k_{41}u_4 + k_{42}u_2u_3 - k_{43}u_1u_4 + 800
\end{aligned}$$

on the unit square and

$$S(x,y) = \begin{cases} 3250 & \text{if } (x,y) \in [0.5, 0.6]^2 \\ 360 & \text{otherwise} \end{cases}.$$

The boundary conditions are homogeneous Neumann boundary conditions. The constants are given by

$$\begin{aligned}
D &= 0.5 \times 10^{-9}, \\
k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16} &= 4 \times 10^5, 272.443800016, 10^{-4}, 0.007, 3.67 \times 10^{-16}, 4.13 \times 10^{-12}, \\
k_{21}, k_{22}, k_{23}, k_{24} &= 272.4438, 1.00016 \times 10^{-4}, 3.67 \times 10^{-16}, 3.57 \times 10^{-15}, \\
k_{31}, k_{32}, k_{33}, k_{34} &= 1.6 \times 10^{-8}, 0.007, 4.1283 \times 10^{-12}, 3.57 \times 10^{-15}, \\
k_{41}, k_{42}, k_{43} &= 7.000016 \times 10^{-3}, 3.57 \times 10^{-15}, 4.1283 \times 10^{-12}.
\end{aligned}$$

This example is a badly scaled problem. The initial values

$$u_1(x,y) = 10^9, u_2(x,y) = 10^9, u_3(x,y) = 10^{13}, u_4(x,y) = 10^7$$

create a highly nonlinear problem. In contrast to that, the starting approximations

$$u_1(x,y) = 1.306028 \times 10^6, u_2(x,y) = 1.076508 \times 10^{12},$$
$$u_3(x,y) = 6.457715 \times 10^{10}, u_4(x,y) = 3.542285 \times 10^{10}$$

yield a mildly nonlinear problem.

The differential equation is discretized on a square grid using a grid size of $h = 1/N$.

4.2 Results

The following tables provide a certain impression about the behavior of the different algorithms. Because we have tried to implement them using identical techniques, we expect them to be comparable. The programs were run without any tuning, some exceptions are given if they are simply applicable. In that case, the parameters are changed for all programs in an identical fashion. In order to have a certain comparison of what can be obtained by rather sophisticated techniques we added a straightforward implementation of the ARMIJO damping strategy (14)-(15). The convergence criterion bases solely on the residual measured in the Euclidean norm. This is essentially the strategy used in the solver `femnlin` within version 1.1 of FEMLAB®.

For the scaling of the unknowns the strategy (30) is used. An example where we succeeded with (29) is given additionally. In general, (32) seems to be much worse than (29).

The abbreviations within the following tables are as follows:

Algorithms	`nlin`	straightforward implementation
	`dnw`	affine invariant Newton method
	`trds`	trust region method
	`btds`	Newton method with backtracking line search
Statistics	NF	number of function evaluations
	NJ	number of Jacobian evaluations (equal to the number of LU-decompositions)
	NBS	number of backsubstitutions
	cond	condition number estimation of the scaled Jacobian at the solutions "$a.bc(d)$" should be read as $a.bc \times 10^d$.
	Result	indication of the success of the method • ok error criterion fulfilled • fm maximal number of allowed Newton steps exceeded • fd underflow of the damping factor An "(ok)" indicates that a solution was obtained but the algorithm did not recognize this.

The Bratu Problem The following results are obtained for a value of $\lambda = 6.8$ which is rather close to the turning point. Nevertheless, all methods are converging rather fast.

	NF	NJ	NBS	cond	Result
`nlin`	9	8	8	7.64(3)	ok
`dnw`	9	8	16	9.51(3)	ok
`trds`	10	9	9	1.52(4)	ok
`btds`	10	9	9	1.52(4)	ok

Because of the nice results above, the next test was carried out using the quasi-Newton methods proposed in the previous paragraph. It turns out that the security check which is additionally implemented in the affine invariant Newton method seems to see the nearby singularity such that Broyden steps are only allowed in a close neighborhood of the solution. So only two Broyden updates are carried out, and the number of iteration steps is not increased compared to the "pure" Newton method. In the present case, the almost uncritical use of Broyden updates in the last two methods leads to a more efficient algorithm. Since the convergence breaks down, the Jacobian is reevaluated in these cases.

	NF	NJ	NBS	cond	Result
`nlin`	9	8	8	7.64(3)	ok
`dnw`	9	6	16	8.72(3)	ok
`trds`	12	2	11	1.18(4)	ok
`btds`	12	3	11	1.47(4)	ok

A Tubular Chemical Reactor Model This model gave rise to a lot of experiments. Surprisingly, we were not able to compute a solution using the most simple initial guess of taking all unknowns identical to one. The following tables are computed using the parameter set 6. The number of discretization points was chosen to be 1000 in order to be able to resolve at least moderate boundary layers. The first table shows that the problem is rather simple if a constant value of 0.5 is taken as an initial guess.

	NF	NJ	NBS	cond	Result
`nlin`	6	5	5	1.76(6)	ok
`dnw`	5	4	8	1.11(5)	ok
`trds`	6	5	5	2.43(5)	ok
`btds`	6	5	5	6.95(5)	ok

The next experiments contains the results for a constant starting value of 10. The solution computed is different from the previous one. The solution v contains a very narrow initial layer, now. It was not possible to take a much larger value because the exponential term overflows. We were not able to compute the third solution by choosing different initial guesses.

	NF	NJ	NBS	cond	Result
nlin	19	16	16	1.59(10)	ok
dnw	22	21	42	2.24(8)	ok
trds	33	18	18	4.00(8)	fd(ok)
btds	19	17	17	5.76(9)	ok

Since the first initial guess lead to a rather simple problem, we allowed Broyden steps to be taken. As expected, everything went well. Even the affine invariant method used the Broyden update from the beginning in contrast to the Bratu problem.

	NF	NJ	NBS	cond	Result
nlin	6	5	5	1.75(6)	ok
dnw	7	2	12	9.38(4)	ok
trds	9	2	8	2.43(5)	ok
btds	9	3	8	3.23(5)	ok

A Semiconductor Problem This problem turned out to be a rather hard problem. The first two experiments are carried out using the more simple parameter set with $\lambda = 1$. The first table contains the results for the standard options.

	NF	NJ	NBS	cond	Result
nlin	64	17	17	7.22(8)	fd(ok)
dnw	194	101	294	1.76(3)	fm
trds	38	14	14	4.40(9)	ok
btds	21	13	13	3.02(13)	ok

A second run was made in order to see if the “highly nonlinear” strategy for the affine invariant Newton method gives some gain. Indeed, the switch to the relative error together with the more conservative damping factor selection gave better results.

	NF	NJ	NBS	cond	Result
nlin	64	17	17	7.22(8)	fd(ok)
dnw	27	21	47	1.67(2)	ok
trds	45	14	14	4.40(9)	ok
btds	21	13	13	3.02(13)	ok

The last for tables contain results for the hard parameters. Taking all standard options and $\lambda = 1$ no method was able to solve the problem.

	NF	NJ	NBS	cond	Result
nlin	37	4	4	2.23(13)	fd
dnw	65	6	12	1.05(3)	fd
trds	16	3	3	2.35(10)	fd
btds	22	3	3	2.35(10)	fd

Therefore, during the following experiments we used the parameter $\lambda = 10^{-3}$. Setting the initial damping factor to 0.01, the following results are obtained:

	NF	NJ	NBS	cond	Result
`nlin`	53	14	14	3.96(8)	fd(ok)
`dnw`	33	32	64	9.28(2)	ok
`trds`	60	29	29	4.95(10)	ok
`btds`	402	101	101	4.92(15)	fm

The highly nonlinear options does not have any influence. Instead, the initial damping factor is important in the present problem. For an initial factor of 1 we have:

	NF	NJ	NBS	cond	Result
`nlin`	53	14	14	3.96(8)	fd(ok)
`dnw`	29	24	52	9.28(2)	ok
`trds`	39	19	19	4.76(10)	ok
`btds`	402	101	101	4.92(15)	fm

Supersonic Transport Air Pollution The present example shows clearly the importance of a correct scaling. In the first runs, the initial guess leading to a mildly nonlinear problem is chosen. With the standard options we got:

	NF	NJ	NBS	cond	Result
`nlin`	902	101	101	6.27(11)	fm
`dnw`	5	4	8	1.92(12)	ok
`trds`	108	101	101	3.89(9)	fm
`btds`	504	101	101	1.10(10)	fm

In order to test the scaling (29) we ran the same example using this formula.

	NF	NJ	NBS	cond	Result
`nlin`	902	101	101	6.27(11)	fm
`dnw`	13	6	18	2.31(12)	fd
`trds`	108	101	101	2.44(12)	fm
`btds`	504	101	101	4.93(12)	fm

The next two examples are identical to the previous ones. The only difference is that the hard initial guesses are provided now.

	NF	NJ	NBS	cond	Result
`nlin`	16	2	2	5.37(10)	fd
`dnw`	16	15	30	1.92(12)	ok
`trds`	4	2	2	9.93(10)	fd
`btds`	13	2	2	9.93(10)	fd

	NF	NJ	NBS	cond	Result
nlin	16	2	2	5.37(10)	fd
dnw	28	18	45	2.08(12)	fd(ok)
trds	8	2	2	5.89(9)	fd
btds	13	2	2	5.89(9)	fd

Again, the scaling (29) turns out be be bad. Note that it was not necessary to switch to the conservative damping strategies here.

5 Conclusions

During our experiments, the affine invariant Newton method and the trust region method turned out to be the most robust methods among our candidates. They were able to solve most of the more extreme problems. The penalty for the higher complexity in easy problems is not high. This is especially true if the quasi-Newton method via recursive Broyden updates is used. While the trust region method is a little bit more efficient for problems where both it and the affine invariant method apply, the latter has its real power for very highly nonlinear problems and badly scaled problems. The latter seems to be important if the user is not able to supply correctly scaled data.

Acknowledgment. The author is indebted to Dr. Ulrich Nowak, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany, for discussions about the topic and to Prof. Dr. Peter Deuflhard, Konrad-Zuse-Zentrum für Informatinstechnik, for providing a draft version of [10].

References

[1] Eugene L. Allgower and Kurt Georg. *Numerical continuation methods. An introduction*, volume 13 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin etc., 1990.

[2] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific J. Math.*, 16:1–3, 1966.

[3] U. Ascher and M.R. Osborne. A note on solving nonlinear equations and the natural criterion function. *JOTA*, 55(1):147–152, 1987.

[4] H.G. Bock. Nuerical treatment of inverse problems in chemical reaction kinetics. In K.H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Ser. Chem. Phys.*, pages 102–125. Springer-Verlag, Berlin etc., 1981.

[5] C.G. Broyden. A class of solving nonlinear simultaneous equations. *Math. Comp.*, 19:577–593, 1965.

[6] A. Cauchy. Méthode générale pour la résolution des sỳstems d'équations simultanes. *C.R. Acad. Sci. Paris*, 25:536–538, 1847.

[7] D.F. Davidenko. On a new method for the numerical solution of systems of nonlinear equations (in Russian). *Dokl. Akad. Nauk SSSR*, 88:601–602, 1953.

[8] C. den Heijer and W.C. Rheinboldt. On steplength algorithms for a class of continuation methods. *SIAM J. Numer. Anal.*, 18:925–948, 1981.

[9] P. Deuflhard. A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with applications to multiple shooting. *Numer. Math.*, 22:289–315, 1974.

[10] Peter Deuflhard. Newton techniques for highly nonlinear problems – theory and algorithms. To be published with Academic Press, Inc.

[11] A. Gleyzal. Solution of nonlinear equations. *Quart. J. Appl. Math.*, 17:95–96, 1959.

[12] Michael Hanke, René Lamour, and Renate Winkler. The program system RWA for the solution of two-point boudary value problems – foundations, algorithms, comparisions. Seminarbericht 67, Humboldt-Univ., Sektion Mathematik, Berlin, 1984.

[13] K.L. Hiebert. An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Transactions Math. Software*, 8:5–20, 1982.

[14] Jr. J.E. Dennis and H.H.W. Mei. Two new unconstraint optimization algorithms which use function and gradient values. *J. Optim. Theory Appl.*, 28:453–482, 1979.

[15] Jr. J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstraint Optimization and Nonlinear equations*. Prentice Hall, Englewood Cliffs, N.J., 1983.

[16] M. Kubíček, V. Hlaváček, and M. Holodniok. Test examples for the comparision od codes for nonlinear boundary value problems in ordinary differential equations. In B. Childs, M. Scott, J.W. Daniel, E. Denman, and P. Nelson, editors, *Codes for Boundary-Value Problems in Ordinary Differential Equations*, volume 76 of *Lecture Notes in Computer Science*, pages 325–346, Berlin, 1979. Springer-Verlag. Proceedings of a Working Conference, May 14–17, 1978.

[17] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[18] D. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[19] Jorge J. Moré. The Levenberg-Marquardt algorithm: Implementtion and theory. In G.A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116, Berlin, 1978. Springer-Verlag. Prooceddings od the Biannual Conference held at Dundee, June 28–July 1, 1977.

[20] Jorge J. Moré. A collection of nonlinear model problems. In Eugene L. Allgower and Kurt Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, volume 26 of *Lectures in Applied Mathematics*, pages 723–762. American Mathematical society, Providence, R.I., 1990.

[21] Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstrom. Testing unconstraint optimization software. *ACM Transactions Math. Software*, 7:17–41, 1981.

[22] U. Nowak and L. Weimann. GIANT — a software package for the numerical solution of very large systems of highly nonlinear equations. Technical Report TR-90-11, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1990.

[23] U. Nowak and L. Weimann. A family of Newton codes for systems of highly nonlinear equations. Technical Report TR-91-10, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1991.

[24] M.J.D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical methods for nonlinear algebraich equations*. Gordon and Breach, London, 1970.

[25] L.K. Schubert. Modification of a quasiNewton method for nonlinear equations with a sparse Jacobian. *Math. Comp.*, 24:27–30, 1970.

[26] R.F. Sincovec and N.K. Madsen. Software for nonlinear partial differential equations. *ACM Trans. on Math. Software*, 1:232–260, 1975.