Menck, J.

An Approximate Newton-Like Coupling of Subsystems

Complex technical systems are often assembled from well-studied subsystems. Here, we elaborate on the obvious idea of coupling existing subsystem solvers to solve a coupled system. More specifically, we present a matrix-free iterative method that is inspired by a Newton type coupling of the subsystems but aims at efficiently controlled linear convergence. We prove its convergence and propose a control mechanism to optimize its efficiency. We illustrate the method's properties with the help of a numerical example.
Note: (As yet) we only deal with the stationary case; mathematically speaking, we are looking for roots of systems of nonlinear equations.

Address. *Jürgen Menck*, Technische Universität Hamburg-Harburg, Arbeitsbereich Mathematik, Kasernenstraße 12, D-21073 Hamburg, Federal Republic of Germany, `http://www.tu-harburg.de/mat/`, `menck@tu-harburg.de`

Keywords. Newton's method, approximate Newton's method, block-structured Newton's method, coupled systems, stationary process simulation, work control.

1 Introduction

Complex technical systems are often assembled by coupling well-known subsystems together. Typically, all of these units can be attacked with reliable solvers, maybe even specialized software packages, as long as they stand alone. It may however be doubtful how to handle them if they all interact. There are basically two substantially different approaches to the problem:

- *The analytic approach:* Re-analyse the units, recover explicit sets of equations, assemble a set of equations for the coupled system, then use (possibly optimized) standard solvers (e. g. Newton).
- *The synthetic approach:* Try to combine the given subsystem solvers to get a solver for the coupled system.

The analytic approach is for example the basis of the well-established flowsheeting software SPEEDUP [5]. While it certainly has a lot of advantages, there can still be good reasons to stick to the synthetic approach:

- It may seem preferable to keep a *modular* view of the coupled system because it helps judging the validity of the results.
- It may be too *expensive* to re-analyse the units or it may be unreasonable to ignore the development costs represented by the subsystem solvers.
- It may seem *dangerous* to re-analyse the units. While the subsystem solvers are probably well-tested and tuned, the new equations may be incorrect or technically awkward.
- It may even be *impossible* to re-analyse the units. In fact, they may be represented by some sort of *black box* solvers, particularly some sort of commercial software that forbids access to the source code.

In many applications the synthetic approach to the problem has thus been favoured. Unfortunately, the most popular way to couple the solvers is also the most naive: Everything is chained in some sort of block Gauss-Seidel process or block Jacobi process, which may result in divergence or very slow convergence of the hybrid solver. Sometimes the convergence can be improved by re-ordering the systems in a clever way, but this requires extensive testing.

Consequently, attempts have been made to couple the solvers in a more sophisticated way, thus ending up with a locally convergent process. Our own process is based on a well-established concept which we will call T(angential)

B(lock) N(ewton). Basically, TBN computes Newton steps for the coupled system while retaining the given block structure. As this interpretation of the method suggests, TBN is guaranteed to converge under quite general assumptions, and it will even converge quadratically if executed exactly. Numerical experience however shows that if the subsystems are handled with linearly convergent solvers, it will typically be too expensive to force quadratic convergence on the hybrid method. Consequently we concentrate on controlling the behaviour of a *linearly* convergent approximation of TBN and making it as efficient as possible.

Tony F. Chan ([7], [8]) seems to have been the first to apply the TBN concept to our basic problem. His method, however, differs significantly from ours in some respects, and a proof of convergence required restrictive assumptions. The present paper is closely related to the work of Artlich and Mackens ([2], [3], [4]) although the emphasis on *linear* convergence is a novel feature.

2 Problem

Let us now outline the details of our setting: Suppose we are given $k \in \mathbb{N}$ subsystems, each depending on a set $x_i \in \mathbb{R}^{k_i}$ of k_i internal variables and a set $y \in \mathbb{R}^{k_c}$ of common external (or "coupling") variables. Each system is represented by its respective solver, which is assumed to be an iterative process

$$x_i^{n+1} \quad := \quad \Phi_i(x_i^n, y) \ , \quad n \in \mathbb{N} \tag{1}$$

(Note that a direct solver will also qualify.) To keep notations simple, we merge the subsystems into a large system

$$x \quad = \quad \Phi(x, y) \ , \quad x = (x_1, \ldots, x_k) \ , \quad \Phi = (\Phi_1, \ldots, \Phi_k) \tag{2}$$

Actually, the characteristic orders of magnitude of the $\|x_i\|$ and the contraction numbers of the Φ_i may differ considerably between the different units. So in practice it may be advisable to balance the systems by rescaling the internal variables and executing some of the slower solvers repeatedly, thus using

$$x_i^{n+1} \quad := \quad \delta_i \Phi_i^{\nu_i}(\delta_i^{-1} x_i^n, y) \ , \quad \delta_i \text{ a positive diagonal matrix} \ , \quad \nu_i \in \mathbb{N} \tag{3}$$

instead of (1). For the purpose of the present paper, consider everything balanced in a reasonable way. Our composite system reads

$$x \quad = \quad \Phi(x, y) \ , \tag{4}$$

$$g(x, y) \quad = \quad 0 \ , \tag{5}$$

where g represents the coupling of the units. Setting

$$f(x, y) \quad := \quad x - \Phi(x, y) \ , \tag{6}$$

we can reformulate this as a root finding problem:

$$f(x, y) \quad = \quad 0 \ , \tag{7}$$

$$g(x, y) \quad = \quad 0 \ . \tag{8}$$

We assume that g consists of exactly as much equations as y has components, such that the coupled system is "square". The aim of our method will be to reduce suitable norms of the residual errors in (7) and (8). Thus we finally end up with an optimization problem,

$$\max(\|f(x, y)\|, \|g(x, y)\|) \quad = \quad \min! \ . \tag{9}$$

If there is a (locally) unique solution of (7) and (8), this system will be locally equivalent to (9).

Suppose our starting point lies in the vicinity of a solution $(\hat{x}, \hat{y})$ of (7), (8) at which the joint Jacobian of f and g is nonsingular. Intermediate iterates of our method may move away from the solution a bit but our estimates will make it easy to contain them inside a suitable region U if the starting point is "good enough". We assume that f and g are at least $\mathcal{C}^2$ inside U, that their joint Jacobian $\begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix}$ remains nonsingular and that Φ is uniformly contractive in the sense that $\|D_x \Phi(x, y)\| \leq q < 1$ holds for all (x, y).

Of course the last assumption restricts the choice of norms for f. Typically one would use a combination of norms which make the Φ_i contractive. In the sequel, suppose that adequate norms have been chosen. Matrix norms will always be assumed to be the subordinate matrix norms with respect to these vector norms, i.e.

$$\|A\| \quad = \quad \sup_{w \neq 0} \frac{\|Aw\|}{\|w\|} \ . \tag{10}$$

3 Exact Tangential Block-Newton

Our method is based on what we call the T(angential) B(lock) N(ewton) Algorithm. This is basically a blocked Newton's iteration for solving coupled systems of the form (7), (8). Variants of this method have been known and used for many years, see for example [6], [7], [14], [15], [16], [17], [18], [21], [25]. One TBN iteration step consists of one Newton step for f with respect to x,

$$\Delta x \quad := \quad -D_x f(x_n, y_n) f(x_n, y_n) \ , \tag{11}$$
$$x^+ \quad := \quad x_n + \Delta x \ , \tag{12}$$

and one Newton step for g along the tangential space of the manifold $M := \{(x,y) \mid f(x,y) = f(x^+, y_n)\}$ through (x^+, y_n),

$$\Delta y \quad := \quad -S^{-1}(x^+, y_n) g(x^+, y_n) \ , \tag{13}$$
$$x_{n+1} \quad := \quad x^+ - C\Delta y \ , \tag{14}$$
$$y_{n+1} \quad := \quad y_n + \Delta y \ . \tag{15}$$

The above matrices C and S are defined as follows:

$$C(x,y) \quad := \quad (D_x f(x,y))^{-1} D_y f(x,y) \tag{16}$$
$$S(x,y) \quad := \quad -D_x g(x,y) C(x,y) + D_y g(x,y) \tag{17}$$

C is the "correction" matrix that generates the tangential directions of M in the sense that they are all of the form $(-C\Delta y, \Delta y)$. The matrix S is the total derivative of g with respect to these directions. It is also the Schur complement of f_x in the abovementioned Jacobian J (thus the letter "S").

One might say that the TBN Algorithm achieves a one-sided decoupling of f and g: While the first part of one composite step may perturb $\|g\|$ considerably, the second one will not interfere with $\|f\|$ up to at least first order terms of Δy. Thus in a neighbourhood of the solution, both $\|f\|$ and $\|g\|$ will be reduced by the composite step. In fact, up to a quadratically small perturbation exact TBN resembles Newton's method for the large system. (This can be seen by executing a formal block Gaussian elimination step in the linear system defining the Newton step.) Consequently the exact TBN scheme will even converge quadratically (cf. [18]).

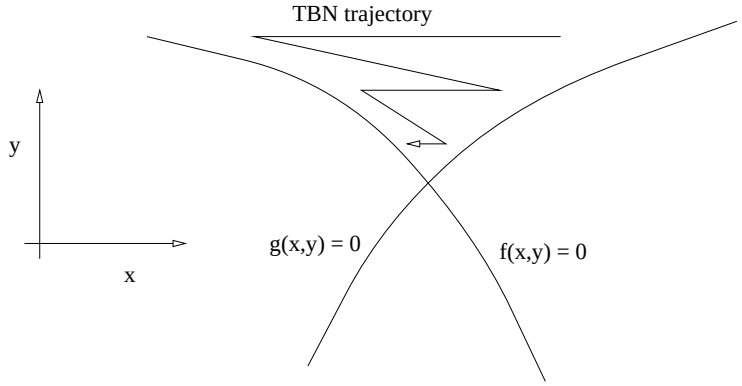

Figure 1: qualitative behaviour of TBN in (x,y)-space

4 Approximate Tangential Block-Newton

In [18] it was recognized that the key processes involved in TBN,

1. computing the Newton step $x^+ - x_n$,
2. multiplying y type vectors by C,
3. solving the linear system $S\Delta y = -g$

may in some sense be perturbed quadratically without losing the convergence properties of TBN. Taking this into account, [2], [3], and [4] proposed to use Φ iterations for 1. and 2. and to replace the exact linear solver in 3. by a suitable Krylov space iteration (more specifically, BiCGStab). While the iterative approach is still the basis of our modified algorithm, numerical results have shown that maintaining quadratic convergence within this framework may in fact be practically not feasible, as it may lead to a prohibitive number of Φ iterations per step. Since Φ itself is only linearly convergent, it seems wiser to aim at an overall *linearly* convergent variant of TBN. Unfortunately this also means abandoning the security of Newton's method. Considerably more sophisticated control mechanisms may be required to ensure convergence, but in the long run the effort will pay off. In this section we will introduce the framework of our modified method, the A(pproximate) T(angential) B(lock) N(ewton) algorithm. Details about its actual implementation, particularly about the setting of the method parameters, will follow in the next sections.

4.1 The f Step

We substitute the original Newton step for f by Φ iterations. It is more or less intuitively clear that this makes sense, since both variants are designed to make $\|f\|$ smaller. In fact, if we only executed enough Φ iterations, the resulting step Δx would be identical with the Newton step up to second order perturbations. But as we have already pointed out, this is *not* our intention. Instead we design the f step such that its accuracy and length can be controlled using step parameters. To this end we introduce

$$0 < \alpha \leq 1 \ , \quad \kappa_1 \in \mathbb{N},$$

where κ_1 denotes the number of successive Φ iterations used and α allows to damp the resulting step:

$$\Delta x := \Phi^{\kappa_1}(x_n, y_n) - x_n \ , \tag{18}$$

$$x^+ := x_n + \alpha \Delta x \ . \tag{19}$$

In terms of efficiency, κ_1 helps balancing the quality and the costs of Δx. α will turn out to be the natural damping parameter of the composite ATBN step if the parameter β introduced below is adapted reasonably. For a local analysis of ATBN, α can thus be assumed to be 1. However, we prefer to keep the value variable for later globalization purposes.

4.2 The g Step

In the g step, we replace C and S by approximations based on the Neumann series of f_x^{-1}. Furthermore, instead of computing (13) exactly, we approximate the solution of the associated linear system $S(x^+, y_n)\Delta y = -g(x^+, y_n)$ using an iterative solver. The modified step has three parameters,

$$0 < \beta \leq 1 \ , \quad \kappa_2 \in \mathbb{N} \ , \quad 0 < \varepsilon_1 \ll 1 \ ,$$

where κ_2 denotes the number of Φ iterations used for approximating C, ε_1 is the error bound for the relative residuum of the linear equation, and β offers the opportunity of damping the computed g step Δy. Our modified matrices are the following:

$$\tilde{C}(x, y) := \sum_{i=0}^{\kappa_2} (D_x \Phi(x, y))^i D_y f(x, y) \ , \tag{20}$$

$$\tilde{S}(x, y) := -D_x g(x, y)\tilde{C}(x, y) + D_y g(x, y) \ . \tag{21}$$

As mentioned above, $\tilde{C}(x,y)$ is derived from C by replacing f_x^{-1} by a truncated Neumann series, and $\tilde{S}$ is the resulting approximation of S. Note that the above expressions are only intended for analytical use, *not* for use in the actual implementation. Many applications of ATBN will involve large numbers of variables, which can make computing and storing the matrices quite expensive. On the other hand, if we approach (22) using a "transpose free" iterative method, the only operations needed from these matrices are in fact matrix-vector products, and we will point out in section 7 how to achieve those matrix free. It will also become clear in that section why we refer to κ_2 as the number of Φ iterations used for $\tilde{C}$.

With the above matrix approximations, the g step takes the form

$$\begin{aligned} \text{Find } \Delta y \quad \text{s.t.} \quad & \|\tilde{S}(x^+, y_n)\Delta y + g(x^+, y_n)\| \leq \varepsilon_1 \|g(x^+, y_n)\| \ , & (22)\\ x_{n+1} \quad := \quad & x^+ - \beta \tilde{C} \Delta y \ , & (23)\\ y_{n+1} \quad := \quad & y_n + \beta \Delta y \ . & (24) \end{aligned}$$

For (22), use BiCGStab or some other transpose free iterative solver. (For details, refer to section 7.)

While κ_2 can be viewed as balancing the accuracy of the tangential space and the computational costs, ε_1 defines the required quality of the Newton step for given $\tilde{C}$. This will become clear in section 5. β is again a damping parameter. We will propose an optimal choice of β depending, among other things, on the value of α. Surprisingly, due to the formulation of the minimization problem (9) and the approximations involved, $\alpha = 1$ will *not* imply $\beta = 1$. Numerical experiments suggest that typically β will instead be slightly smaller than 1.

5 Analysis of the ATBN Step

In this section we analyse the ATBN step as defined in section 4. First we study the effect of the f step on the norms of f and g, then the effect of the g step. Finally we combine the results to get estimates for the effect of the combined step. This section may in part seem too technical but it is a necessary basis for the convergence proof in section 6 and for the development of a reasonable control mechanism in section 7.

In the sequel, f_n and g_n will denote the values of f and g at (x_n, y_n), respectively. Similarly, we shall denote the values at (x^+, y_n) by f^+ and g^+.

First of all we will show that the f step basically reduces f as in

$$\|f^+\| \quad \leq \quad ((1-\alpha) + \alpha q^{\kappa_1})\|f_n\| + \alpha \mathcal{O}(\|f_n\|^2) \ . \tag{25}$$

This is indeed what we intended: The undamped step ($\alpha = 1$) will reduce the norm of f by a factor of q^{κ_1}, and damping can be roughly described as linearly interpolating the values for $\alpha = 0$ and $\alpha = 1$. In this and the following estimates, we keep track of the damping parameters in the remainder terms because we want to stress how damping can force convergence even if the norms of f and g are not yet particularly small. See the convergence theorem in section 6 for an illustration. Basically, this is already a nod in the direction of globalization.

To prove the result, we note that by the implicit function theorem there is a mapping $y \mapsto x_* = x_*(y)$ s.t. in a neighbourhood of $(\hat{x}, \hat{y})$, (x_*, y) solves $f = 0$. An easy induction proves

$$\Phi^\nu(x_n, y_n) \quad = \quad x_*(y_n) + (D_x\Phi(x_*(y_n), y_n))^\nu (x_n - x_*(y_n)) + \mathcal{O}(\|f_n\|^2) \quad \forall \nu \in \mathbb{N} \ , \tag{26}$$

and (suppressing subscripts n for once) we get

$$\begin{aligned} f^+ &= f + \alpha D_x f(x,y)\Delta x + \alpha^2 \mathcal{O}(\|\Delta x\|^2) \\ &= f + \alpha (I - D_x\Phi(x,y))(\Phi^\nu - x) + \alpha^2 \mathcal{O}(\|\Delta x\|^2) \\ &= f + \alpha (I - D_x\Phi(x_*, y) + \mathcal{O}(\|f\|)) \left[((D_x\Phi(x_*,y))^\nu - I)\ (x - x_*) + \mathcal{O}(\|f\|^2) \right] + \alpha^2 \mathcal{O}(\|f\|^2) \\ &= f + \alpha (I - D_x\Phi(x_*, y))\ ((D_x\Phi(x_*,y))^\nu - I)\ (x - x_*) + \alpha \mathcal{O}(\|f\|^2) \\ &= f + \alpha\ ((D_x\Phi(x_*,y))^\nu - I)\ (I - D_x\Phi(x_*, y))(x - x_*) + \alpha \mathcal{O}(\|f\|^2) \\ &= [(1-\alpha)I + \alpha D_x\Phi(x_*, y)^\nu]\, f + \alpha \mathcal{O}(\|f\|^2) \ , \end{aligned}$$

which proves (25). To estimate $\|g^+\|$, we first note that the definition of Δx directly implies

$$\|\Delta x\| \leq \frac{1-q^{\kappa_1}}{1-q}\|f_n\| , \tag{27}$$

and since

$$g^+ = g_n + D_x g(x_n, y_n)\Delta x + \alpha^2 \mathcal{O}(\|\Delta x\|^2) , \tag{28}$$

we get

$$\|g^+\| \leq \|g_n\| + \alpha\mu\frac{1-q^{\kappa_1}}{1-q}\|f_n\| + \alpha^2\mathcal{O}(\|f_n\|^2) , \quad \text{where} \tag{29}$$

$$\mu := \max_{(x,y)\in U} \|D_x g(x,y)\| . \tag{30}$$

This result can be interpreted as follows: Up to first order terms the change of $\|g\|$ will be proportional to the step length. This is mirrored by the factor α in the second right hand side term. To eliminate the unknown quantity $\|\Delta x\|$ from the estimate, we have used an upper bound which clearly shows that the step length will depend on the residuum $\|f_n\|$ before the step and on the number of iterations κ_1. Due to the contractivity of Φ, there is a saturation as $\kappa_1 \to \infty$, i.e. the Φ iteration steps most harmful to $\|g\|$ are the first few. The factor μ is a measure for the sensitivity of $\|g\|$ w.r.t. changes of x. If f and g were decoupled, μ would be 0.

Let us now have a look at the g step. It's effect on the norm of g itself is readily computed: Using (21), (23) and (24), we get

$$g_{n+1} = g^+ + \beta\tilde{S}(x^+, y_n)\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) , \tag{31}$$

and since multiplying (22) by $\tilde{S}^{-1}$ yields

$$\|\Delta y + \tilde{S}^{-1}(x^+, y_n)g^+\| \leq \varepsilon_1\|\tilde{S}^{-1}(x^+, y_n)\| \cdot \|g^+\| , \tag{32}$$

the y step size is bounded according to

$$\|\Delta y\| \leq (1+\varepsilon_1)\|\tilde{S}^{-1}(x^+, y_n)\| \cdot \|g^+\| . \tag{33}$$

This implies that we can replace $\mathcal{O}(\|\Delta y\|^2)$ by $\mathcal{O}(\|g^+\|^2)$, s.t. we can combine (22) and (31) to get

$$\|g_{n+1}\| \leq ((1-\beta)+\beta\varepsilon_1)\|g^+\| + \beta^2\mathcal{O}(\|g^+\|^2) . \tag{34}$$

To better understand this estimate, let us first note that the damping parameter β again achieves a linear interpolation between the extremes $\beta = 0$ and $\beta = 1$. From now on, consider the undamped case $\beta = 1$. In this case the g step basically reduces $\|g^+\|$ by a factor of ε_1. This means that up to first order terms the parameter ε_1 is the contraction rate that we demand from the approximate Newton step for g. As $\varepsilon_1 \to 0$, the behaviour of the step will resemble superlinear convergence, which hints at the fact that Δy will converge against the exact Newton step for given $\tilde{C}$. (Obviously, (22) quite literally *becomes* the equation of the Newton step.) Note that like in the case of the f step, quadratic decrease of the error would increase the computational costs; thus it is not a feature that we really strive for in our actual implementation.

If you are puzzled by the fact that the accuracy of $\tilde{C}$ has no effect on the improvement of $\|g^+\|$, remember that the approximate directions used in the definition of (22) are also used for the actual update of (x_{n+1}, y_{n+1}). Consequently the undamped step will (up to first order terms) actually reduce $\|g\|$ as predicted in (22), and the error in $\tilde{C}$ will only affect $\|f_{n+1}\|$.

To compute an estimate for $\|f_{n+1}\|$, we use $\tilde{C} - C = -(D_x\Phi)^{\kappa_2+1}C$ to write the increment as

$$\begin{aligned}
f_{n+1} - f^+ &= -\beta(D_x f(x^+, y_n)\tilde{C}(x^+, y_n) - D_y f(x^+, y_n))\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) \\
&= -\beta D_x f(x^+, y_n)(\tilde{C}(x^+, y_n) - C(x^+, y_n))\Delta y - \\
&\quad \beta(D_x f(x^+, y_n)C(x^+, y_n) - D_y f(x^+, y_n))\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) \\
&= -\beta D_x f(x^+, y_n)(\tilde{C}(x^+, y_n) - C(x^+, y_n))\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) \\
&= \beta D_x f(x^+, y_n)(D_x\Phi(x^+, y_n))^{\kappa_2+1}C(x^+, y_n)\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) \\
&= \beta(D_x\Phi(x^+, y_n))^{\kappa_2+1}D_x f(x^+, y_n)C(x^+, y_n)\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2)
\end{aligned}$$

$$= \quad \beta(D_x\Phi(x^+, y_n))^{\kappa_2+1} D_y f(x^+, y_n)\Delta y + \beta^2\mathcal{O}(\|\Delta y\|^2) \ , \tag{35}$$

and using (33), we finally get

$$\|f_{n+1}\| \quad \leq \quad \|f^+\| + \beta(1+\varepsilon_1)q^{\kappa_2+1}\lambda\|g^+\| + \beta^2\mathcal{O}(\|g^+\|^2) \ , \quad \text{where} \tag{36}$$

$$\lambda \quad := \quad \max_{(x,y)\in U} \|D_y f(x,y)\| \cdot \|\tilde{S}^{-1}(x,y)\| \ . \tag{37}$$

In most respects this estimate resembles (29) with the roles of f and g reversed: Up to first order terms, the change of $\|f^+\|$ will be proportional to the damping factor β, and there is an analogous relation to the size of $\|\Delta y\|$ which gets obscured by the fact that we have replaced this unknown quantity by other known quantities. The size of $\|g^+\|$ has basically the same effect as the size of $\|f_n\|$ in the aforementioned estimate.

The basic new feature here is the factor q^{κ_2+1}. It shows how improving the quality of $\tilde{C}$ will lessen the sensitivity of f with respect to the g step. For $\kappa_2 \to \infty$, $\tilde{C}$ would converge towards the exact matrix C, and the one-sided decoupling mentioned in the context of exact TBN would become apparent, since up to first order terms the g step would not affect $\|f^+\|$ at all.

It may be noted that while κ_2 was not involved in (34), the present estimate is also very nearly independent of ε_1. Thus the effects of these parameters are quite easy to separate, which helps understanding the way in which ATBN will work.

Finally, let us have a look at the constant λ. In analogy to μ, this quantity can be described as a measure for the sensitivity of $\|f\|$ w.r.t. certain variations. More precisely, λ consists of two components: $\|\tilde{S}^{-1}\|$ measures the response of g to variations in the approximate tangential directions. If for example f and g are strongly coupled, small angles may occur between tangential directions of $\{\, f \equiv \text{const.} \,\}$ and tangential directions of $\{\, g \equiv \text{const.} \,\}$ with the unfortunate consequence that $\|g\|$ may react poorly to variations along the direction used for the g step. In such a situation $\|\tilde{S}^{-1}\|$, and probably the Newton step as well, will become large. In keeping with this interpretation, $\|\tilde{S}^{-1}\|$ has entered our formula via (33).

The second component of λ is $\|D_y f(x,y)\|$: This quantity measures the sensitivity of f w.r.t. variations in y that are not accompanied by any tangential correction in x. It has to be multiplied by the aforementioned factor q^{κ_2+1} to indicate the sensitivity w.r.t. the directions generated using $\tilde{C}$.

In the sequel, let m_n denote $m_n := \max(\|f_n\|, \|g_n\|)$. Then, by combining the above inequalities, we can finally estimate the effect of the combined step on the norms of f and g:

$$\begin{aligned} \|f_{n+1}\| \quad \leq \quad & \left((1-\alpha) + \alpha q^{\kappa_1}\right)\|f_n\| + \beta(1+\varepsilon_1)q^{\kappa_2+1}\lambda\|g_n\| + \alpha\beta q^{\kappa_2+1}\frac{1-q^{\kappa_1}}{1-q}\lambda\mu\|f_n\| + \\ & \alpha^2\beta\mathcal{O}(m_n^2) + \beta^2\mathcal{O}(m_n^2) \ , \end{aligned} \tag{38}$$

$$\begin{aligned} \|g_{n+1}\| \quad \leq \quad & \left((1-\beta) + \beta\varepsilon_1\right)\left(\|g_n\| + \alpha\frac{1-q^{\kappa_1}}{1-q}\mu\|f_n\|\right) + \\ & \alpha^2\mathcal{O}(m_n^2) + \beta^2\mathcal{O}(m_n^2) \ . \end{aligned} \tag{39}$$

Roughly speaking, the first right-hand side term in (38) represents the (intended) reduction of $\|f\|$ by the f step, the second one (with factor β) accounts for the perturbation of $\|f\|$ that would be expected from the g step if executed *before* the f step, and the third term (with factor $\alpha\beta$) is some sort of a feedback in that it models the additional perturbation of $\|f\|$ caused by having to start the g step with an already perturbed g. The factor $((1-\beta)+\beta\varepsilon_1)$ on the right-hand side of (39) accounts for the (intended) reduction of $\|g\|$ in the g step, while the perturbation in the other bracket (with factor α) is the unwelcome effect of the f step on g that is again being mirrored in the feedback term of (38). For further details, please refer to the preceding discussion of the separate f and g steps.

6 Linear Convergence

In the present section we will prove the local linear convergence of ATBN on the condition that the method parameters are set in a reasonable way. Since the result will not rely on a specific control mechanism, we will postpone presenting ours to section 7.

To avoid complicating matters by dealing with different cases, we first replace $\|f_n\|$ and $\|g_n\|$ by the maximum $m_n = \max(\|f_n\|, \|g_n\|)$ on the right hand sides of (38) and (39). Our actual control mechanism will actually be more subtle than that, but for the purpose of the proof the simplified model suffices:

$$\begin{aligned}
\|g_{n+1}\| &\leq q_{g,2} m_n + \alpha^2 \mathcal{O}(m_n^2) + \beta^2 \mathcal{O}(m_n^2) \ , \quad \text{where} && (40)\\
q_{g,1} &:= 1 + \alpha \frac{1-q^{\kappa_1}}{1-q} \mu \ , && (41)\\
q_{g,2} &:= ((1-\beta) + \beta\varepsilon_1) q_{g,1} \ , \quad \text{and} && (42)\\
\|f_{n+1}\| &\leq q_{f,2} m_n + \alpha^2 \beta \mathcal{O}(m_n^2) + \beta^2 \mathcal{O}(m_n^2) \ , \quad \text{where} && (43)\\
q_{f,1} &:= (1-\alpha) + \alpha q^{\kappa_1} \ , && (44)\\
q_{f,2} &:= q_{f,1} + \beta(1+\varepsilon_1) q^{\kappa_2+1} \lambda q_{g,1} \ . && (45)
\end{aligned}$$

Viewing the factors $q_{f,2}$ and $q_{g,2}$ as functions of the damping parameters, we find

$$\begin{array}{rclcrcl}
\frac{\partial}{\partial\alpha} q_{g,2} &\geq& 0 \ , & & \frac{\partial}{\partial\beta} q_{g,2} &<& 0 \ , \\
\frac{\partial}{\partial\alpha} q_{f,2} &<& 0 \ , & & \frac{\partial}{\partial\beta} q_{f,2} &\geq& 0 \ .
\end{array} \qquad (46)$$

Particularly, increasing β will increase $q_{f,2}$ and decrease $q_{g,2}$. Thus, supposing all other parameters are fixed, the optimal choice of β will be

$$\begin{aligned}
\beta &= \max(0, \min(1, \beta_*)) \ , \quad \text{where} && (47)\\
\beta_* &:= \frac{q_{g,1} - q_{f,1}}{q_{g,1}((1+\varepsilon_1) q^{\kappa_2+1} \lambda + (1-\varepsilon_1))} = \alpha \frac{(1-q^{\kappa_1}) \left(1 + \frac{\mu}{1-q}\right)}{q_{g,1}((1+\varepsilon_1) q^{\kappa_2+1} \lambda + (1-\varepsilon_1))} && (48)
\end{aligned}$$

because β_* achieves $q_{f,2} = q_{g,2}$. (Moreover, β_* is the only value of β to achieve this as long as $\lambda \neq 0$.) Elementary calculations using $1 \leq q_{g,1} \leq (1+\mu)/(1-q)$ yield

$$\begin{aligned}
K_- \alpha &\leq \beta_* \leq K_+ \alpha \ , \quad \text{where} && (49)\\
K_- &= \frac{(1-q)^2}{(1+\mu)(1+2\lambda q)} \ , && (50)\\
K_+ &= \frac{1+\mu}{(1-q)(1-\varepsilon_1)} \ . && (51)
\end{aligned}$$

Thus basically β_* and α are proportional and $\beta_* > 0$ is satisfied for every $0 < \alpha \leq 1$. Our choice of β now implies

$$m_{n+1} \leq \max(q_{f,2} \ , \ \varepsilon_1 q_{g,1}) m_n + \alpha^2 \mathcal{O}(m_n^2) \ , \qquad (52)$$

and using the upper bound K_+, a short calculation yields

$$\begin{aligned}
\frac{m_{n+1}}{m_n} &\leq \max(q_\alpha, q_{\min}) + \alpha^2 \mathcal{O}(m_n) \ , \quad \text{where} && (53)\\
q_\alpha &= (1-\alpha) + \alpha \left(q^{\kappa_1} + q^{\kappa_2} \cdot 2q\lambda \frac{(1+\mu)^2}{(1-q)^2(1-\varepsilon_1)} \right) \ , && (54)\\
q_{\min} &= \frac{\varepsilon_1}{1-\varepsilon_1} \frac{(1+\mu)^2}{(1-q)^2} \ . && (55)
\end{aligned}$$

As the names suggest, q_α is the reduction factor associated with β_* while $q_{\min}$ relates to the case $\beta = 1 < \beta_*$. For $\kappa_1, \kappa_2 \to \infty$ and $\varepsilon_1 \to 0$ the estimate simply reads

$$\frac{m_{n+1}}{m_n} \leq (1-\alpha) + \alpha^2 \mathcal{O}(m_n) \ . \qquad (56)$$

This illustrates that asymptotically the ATBN step indeed behaves like a damped Newton step; in particular, the residuum will be reduced quadratically as $\alpha \to 1$. If we introduce an $\varepsilon_1 > 0$ again, (56) will hold true as long as $(1-\alpha)$ remains larger than $q_{\min}$: As we have already remarked, ε_1 is a bound for the reduction of $\|g\|$ in the g step. On the other hand, if we keep $\varepsilon_1 = 0$ and let κ_1 or κ_2 be a finite number, the lower bound $q_{\min}$ will disappear but instead q_α will fail to approximate 0 as $\alpha \to 1$: Obviously, computing the g step exactly will not yet ensure quadratic convergence. Also note that the formula (54) confirms quite clearly that the necessity of using large κ_2 depends very much on λ and μ, as should have been expected.

More importantly, (53) shows that for any combination of sufficiently large κ_1, κ_2 and sufficiently small ε_1 the ATBN algorithm will exhibit linear convergence. Furthermore it is apparently possible and reasonable to choose $\alpha = 1$ near the solution; α is thus a true damping parameter for the composite ATBN step which will only be needed for globalization purposes. Finally the attainable factor m_{n+1}/m_n is only bounded by $\mathcal{O}(m_n)$ from below, which means that (theoretically) any convergence rate can be achieved. This is not to be misunderstood as a recommendation: Always bear in mind that the convergence rate m_{n+1}/m_n refers to the composite ATBN step as a black box solver. It does not allow for an evaluation of the computational costs involved. We will come back to this point in the following theorem:

Theorem (Convergence of Approximate Tangential Block-Newton)

Let the assumptions of section 2 hold true.

a) ATBN can be made to converge linearly with any given convergence rate $0 < q_{\text{comp.}} < 1$. More precisely, there is a neighbourhood of the solution such that for sufficiently large κ_1, κ_2 and sufficiently small ε_1

$$\frac{m_{n+1}}{m_n} \quad \leq \quad (1-\alpha) + \alpha q_{\text{comp.}} \tag{57}$$

will hold for all $0 \leq \alpha \leq 1$ and suitably chosen $\beta = \beta(\alpha)$. In particular, the convergence rate $q_{\text{comp.}}$ is achieved for $\alpha = 1$. Using smaller α will in most cases enlarge the domain of attraction of the solution (as well as the domain of validity for (57)).

b) ATBN can be made to achieve an *effective* convergence rate $q \leq q_{\text{eff.}} < 1$ in the following sense: Assume that the iterative solver always succeeds in solving the modified g equation (22) and that the number of iteration steps needed is bounded for any given ε_1. Let $\kappa = \kappa(x, y, \kappa_1, \kappa_2, \varepsilon_1, \alpha)$ denote the number of Φ evaluations needed to compute the ATBN step. Then there is a neighbourhood of the solution and a $q \leq q_{\text{eff.}} < 1$ such that for $\alpha = 1$ and suitably chosen κ_1, κ_2, ε_1 and β the ATBN step will satisfy

$$\left(\frac{m_{n+1}}{m_n}\right)^{1/\kappa} \quad \leq \quad q_{\text{eff.}} \ . \tag{58}$$

The proof will be sketched below. To convince yourself that the assumption of b) on the linear solver is reasonable, suppose $w_0 = 0$ is used as the start approximation. The corresponding linear residuum $\tilde{S}w_0 + g^+$ will then be g^+, and (22) can be interpreted as a condition to reduce the linear residuum by a (fixed) factor of ε_1. *Any* acceptable iteration process should achieve that within a finite number of steps, and in fact there should also be a bound on that number which (in a neighbourhood of $(\hat{x}, \hat{y})$) only depends on ε_1. Unfortunately it is known that for the kind of solvers we want to use, namely BiCGStab or comparable Krylov subspace methods, the convergence behaviour is very hard to predict. Even if the notorious breakdowns of Lanczos based algorithms are excluded, the methods may fail to converge in finite precision arithmetic, see [12]. On the other hand, even if exact arithmetic is assumed *and* the iterates of the method at hand satisfy a minimization property, the convergence may still be intolerably slow, see for example [11]. In other words, the typical iteration process will *not* be acceptable for *all* kinds of $\tilde{S}$. The problem seems intrinsic, and the best help is probably providing alternative solvers to choose from if the default method should fail. Note that if one Krylov subspace method fails to solve a problem efficiently, another one may still succeed ([19]).

Proof. Part a) of the theorem is an immediate consequence of (53), which is almost also true for part b): Under the assumptions of b) the number of Φ evaluations needed to compute the step inside a given neighbourhood of the solution can be bounded in dependence of the parameters κ_1, κ_2 and ε_1 only,

$$\kappa(x, y, \kappa_1, \kappa_2, \varepsilon_1, \alpha) \quad \leq \quad \hat{\kappa}(\kappa_1, \kappa_2, \varepsilon_1) \ . \tag{59}$$

Thus, choosing a neighbourhood, a $q_{\text{comp.}}$ and suitable κ_1, κ_2 and ε_1 according to a) yields

$$\left(\frac{m_{n+1}}{m_n}\right)^{1/\kappa} \quad \leq \quad \left(\frac{m_{n+1}}{m_n}\right)^{1/\hat{\kappa}} \quad \leq \quad (q_{\text{comp.}})^{1/\hat{\kappa}} \ , \tag{60}$$

which proves that indeed a $q_{\text{eff.}}$ exists and $(q_{\text{comp.}})^{1/\hat{\kappa}}$ is a possible choice. Optimizing the value of $q_{\text{eff.}}$ will be the task of an intellingent control mechanism (cf. the following section).

7 Practicalities

The present section is dedicated to the actual implementation of ATBN. It divides into two subsections: The first, "Matrix Implementation", addresses the handling of the matrices $\tilde{C}$ and $\tilde{S}$ as defined in section 4. It points out how computing and storing these matrices can be avoided by interpreting matrix-vector products as directional derivatives. In the second subsection, "Parameters", we develop a control mechanism for the method parameters β, κ_1, and κ_2 which aims at minimizing the effective error reduction q_{eff} defined in the convergence theorem, or rather the effective reduction derived from (38) and (39). Note that there is a difference due to certain simplifications involved in computing (53).

7.1 Matrix Implementation

As mentioned in section 4, we do not recommend computing the matrices $\tilde{C}$ and $\tilde{S}$ as defined. Instead we propose to organize ATBN such that only matrix-vector products with these matrices are needed, and to compute these products with the help of appropriate differencing schemes. One way to do this is solving (22) with the help of Krylov subspace methods.

This outline presumes that the number k_c of coupling variables (i.e. the number of components of y) is reasonably large. Otherwise it might be preferable to assemble the matrix $\tilde{S}$ and use a direct solver to compute Δy from $\tilde{S}\Delta y = -g^+$. One way to achieve this would be to compute the columns of $\tilde{S}$ as $\tilde{S}e_i$ for $i = 1, \ldots, k_c$ using the differencing scheme described below.

Krylov subspace methods, however hard to analyse in the case of nonsymmetric system matrices, are widely popular and well-respected. Although systems can be constructed which make them fail or perform very poorly, it is generally believed that they will work well for certain classes of large systems with "a lot of structure", particularly in combination with an appropriate preconditioner. However, since in general our matrix $\tilde{S}$ will not be symmetric, many of the usual methods, like CGNE, CGNR, LSQR, BiCG, BiORes, or QMR, will not only require matrix-vector products with $\tilde{S}$ but also matrix-vector products with its transpose $\tilde{S}^T$, and since it is not clear how to compute these via clever differencing schemes, we restrict ourselves to the class of *transpose free* algorithms. These are mainly orthogonalization methods based on the Arnoldi process, like GMRES ([20]) or truncated versions of it (e. g. GMRES(m)), or "squared" methods based on the two-sided Lanczos process, like CGS ([23]), BiCGStab ([24]), BiCGStab(ℓ) ([22]), or TFQMR ([10]). (For an overview of Lanczos-type solvers, one can also refer to [13].)

Assuming we use one of these solvers, we only need to know how to approximate matrix-vector products of $\tilde{C}$ and $\tilde{S}$. In the case of $\tilde{C}$, this can be done using the differencing scheme

$$\tilde{C}(x,y)w \simeq \Psi_w^{\kappa_2+1}(0;x,y) \ , \quad \text{where} \tag{61}$$

$$\Psi_w(r;x,y) := \frac{\Phi(x+h_2 r, y) - \Phi(x,y)}{h_2} + \frac{f(x, y+h_1 w) - f(x,y)}{h_1} \ . \tag{62}$$

The formula is an immediate consequence of $\Psi_w(r;x,y) \simeq \Phi_x(x,y)r + f_y(x,y)w$. It is clear from this argument that h_1 and h_2 are supposed to be small numbers suited to approximate the implied directional derivatives. As a rule of thumb, they should be of the order of $\sqrt{\varepsilon}$, where ε is the machine precision. More precisely, we use

$$h_1 := \sqrt{\varepsilon}\,\frac{\max(\|y\|,1)}{\max(\|w\|,\varepsilon_2)} \ , \tag{63}$$

$$h_2 := \sqrt{\varepsilon}\,\frac{\max(\|x\|,1)}{\max(\|r\|,\varepsilon_2)} \ . \tag{64}$$

Here ε_2 is a very small positive number designed to avoid zerodivide errors. The choice of h_i in these formulae is inspired by [9].

Assuming that $\tilde{C}$ is handled as shown, we can approximate matrix-vector products of $\tilde{S}$ according to

$$\tilde{S}(x,y)w \simeq \frac{g(x - h_3\tilde{C}(x,y)w, y + h_3 w) - g(x,y)}{h_3} \ . \tag{65}$$

This follows from the fact that if we define

$$\tilde{g}_{(x,y)}(\Delta y) \quad := \quad g(x - \tilde{C}(x,y)\Delta y, y + \Delta y) \tag{66}$$

to represent g on the approximate tangential space, directional derivatives with respect to the coordinate directions will correspond to products of $\tilde{S}$ and the respective direction vectors:

$$\frac{\partial}{\partial w}\tilde{g}_{(x,y)}(0) \quad = \quad \tilde{S}(x,y)w \ . \tag{67}$$

Similar to h_1 and h_2, we choose as h_3

$$h_3 \quad := \quad \sqrt{\varepsilon}\,\frac{\max(\|(x,y)\|,1)}{\max(\|(\tilde{C}w,w)\|,\varepsilon_2)} \ . \tag{68}$$

7.2 Parameters

We will now discuss the control mechanism for ATBN's method parameters. In keeping with the previous sections, we shall concentrate on the local convergence behaviour of the method. Consequently, we suppose that the remainder terms of our estimates will be negligible and that it will be possible to let $\alpha = 1$ throughout. For the sake of later generalizations we will develop our control mechanism for arbitrary given α, but adaptation strategies for α itself as well as other aspects of globalization will be postponed to a forthcoming paper.

The following control mechanism may be enhanced by including special strategies for certain special situations: For example, the cases $\|f\| \ll \|g\|$ and $\|f\| \gg \|g\|$ can be treated separately by executing only the g or the f step, respectively; these cases may actually occur during a startup phase, depending on the starting values.

Our control mechanism basically works like this:

1. $\alpha = 1$ unless complications occur (in which case α may have to be damped).
2. ε_1 is provided by the user. κ_1 and κ_2 will adapt to ε_1 but as a rule of thumb, ε_1 should be chosen small but not *too* small to avoid unnecessary costs in computing (22). It may be a good idea to warn the user if the given value of ε_1 does not seem reasonable, or to adapt the value automatically.
3. We supply a formula to choose an optimal β provided all the other parameters are fixed.
4. We eliminate β from the model by inserting this optimal value. Thus we can consider the estimated effective reduction factor $q_{\text{eff.}}$ as a function of κ_1 and κ_2 . We minimize this function.
5. We use the values κ_i and the associated optimal β that we have just computed. At this point a damping strategy can attack if the true results differ significantly from those of the model.

The decision to let the user supply ε_1 stems from the observation that there is a certain redundancy among the ATBN method parameters and for the typical case of a fairly poor contraction rate q the values of κ_i will be much more significant for the quality of the step than ε_1. Furthermore it would be difficult to predict the number of iteration steps associated with a given value of ε_1.

Let us now work out the details of β and κ_i: If we view the right-hand sides of the estimates (38) and (39) as functions of α and β, a similar argument as in (46) will show that if everything else is fixed, the optimal choice of β again will be

$$\beta \quad = \quad \min(1, \beta_*) \tag{69}$$

where β_* is the value of β for which the right-hand sides of (38) and (39) coincide. Note that this is not exactly the β_* from (48) because we have not substituted m_n for the values of $\|f_n\|$ and $\|g_n\|$. To find an approximation, we interpolate $\|f\|$ and $\|g\|$ linearly between (x^+, y_n) and $(x^+ - \tilde{C}\Delta y, y_n + \Delta y)$, that is, between $\beta = 0$ and $\beta = 1$. With "++" denoting the values for $\beta = 1$, the result takes the form

$$\beta_* \quad = \quad \frac{\|g^+\| - \|f^+\|}{(\|f^{++}\| - \|f^+\|) - (\|g^{++}\| - \|g^+\|)} \ , \tag{70}$$

$$m_{n+1} \simeq \frac{\|f^{++}\| \cdot \|g^{+}\| - \|f^{+}\| \cdot \|g^{++}\|}{(\|f^{++}\| - \|f^{+}\|) - (\|g^{++}\| - \|g^{+}\|)} \ . \tag{71}$$

(The formula for m_{n+1} has been included because it can be helpful in checking the validity of the linear interpolation.) The choice of β can be fine-tuned with respect to special situations (like e. g. , $\|f^{++}\| \leq \|f^{+}\|$) but β_* is a reasonable standard.

To choose κ_1 and κ_2, we now provide a model for the composite step: We use the right hand sides of (25), (36), (29), (34) to model the norms of f and g. We eliminate β from the expressions by inserting the optimal value just computed. Then we get:

$$\|f^{+}\| \simeq ((1-\alpha) + \alpha q^{\kappa_1})\|f_n\| \ , \tag{72}$$

$$\|g^{+}\| \simeq \|g_n\| + \alpha\mu \frac{1-q^{\kappa_1}}{1-q}\|f_n\| \ , \tag{73}$$

$$m_{n+1} \simeq \max(\varepsilon_1\|g^{+}\|, m^{*}_{n+1}) \ , \quad \text{where} \tag{74}$$

$$m^{*}_{n+1} := \frac{(1-\varepsilon_1)\|f^{+}\| + (1+\varepsilon_1)\lambda q^{\kappa_2+1}\|g^{+}\|}{(1-\varepsilon_1) + (1+\varepsilon_1)\lambda q^{\kappa_2+1}} \ . \tag{75}$$

If we want to evaluate these estimates for given values of κ_i, we will have to supply q, λ, and μ. To this purpose, we again refer to our estimates (25), (29), and (36). Supposing we have computed $\|f^{+}\|$, $\|g^{+}\|$, and $\|f^{++}\|$ in the current step, we can conclude:

$$q \simeq \left(\frac{\|f^{+}\| - (1-\alpha)\|f_n\|}{\alpha\|f_n\|}\right)^{1/\kappa_1} \ , \tag{76}$$

$$\lambda \simeq \frac{|\,\|f^{++}\| - \|f^{+}\|\,|}{\beta q^{\kappa_2+1}(1+\varepsilon_1)\|g^{+}\|} \quad \text{(with the above } q \text{ estimate)}, \tag{77}$$

$$\mu \simeq \frac{|\,\|g^{+}\| - \|g_n\|\,|}{\alpha\|\Delta x\|} \ . \tag{78}$$

Once we have computed these values, we can feed them back into the model and optimize the next step. It is also possible and probably reasonable to use convex combinations of the current approximations and earlier values to smoothen out oscillations between the steps. None of this is of course possible for the very first step: Here we will have to work with an educated guess or some default values.

Finally we suppose that the number ℓ of (outer) steps that the linear solver needs to satisfy (22) remains approximately constant, which enables us to estimate the number κ of Φ evaluations belonging to a given pair (κ_1, κ_2). Of course κ depends on the linear solver and on details of the actual implementation but a typical number for a BiCGStab based approach would be

$$\kappa \simeq (3+\kappa_1) + 2(\ell+1)(\kappa_2+1) \ . \tag{79}$$

Using the m_{n+1} estimate (74), we optimize the κ-th root of the reduction factor m_{n+1}/m_n for $1 \leq \kappa_1, \kappa_2 \leq \kappa_{\max}$, where $\kappa_{\max}$ is some upper bound for the admissible number of Φ iterations in a row (cf. part b) of the convergence theorem). Again, note that the present estimate of the reduction factor is sharper than the one in (53) because the latter involved several simplifications that we have avoided here.

8 A Numerical Example

In the present section we shall demonstrate some properties of ATBN and our control mechanism by studying a well-known model problem. We chose the Bratu problem on the unit square,

$$\begin{aligned} -\Delta u(\xi_1,\xi_2) &= \sigma\exp(u(\xi_1,\xi_2)) \quad \text{for} \quad 0 < \xi_1, \xi_2 < 1 \ , \\ u(\xi_1,\xi_2) &= 0 \qquad\qquad\qquad\quad \text{if} \quad \xi_1 = 0 \text{ or } \xi_1 = 1 \text{ or } \xi_2 = 0 \text{ or } \xi_2 = 1 \ . \end{aligned} \tag{80}$$

In terms of Chemical Engineering, (80) can be viewed as a model problem for diffusion and exothermic reaction (cf. [1]). We used the standard 5 point discretization of the Laplacian on a uniform grid. We divided the unit square into four identical squares; this substructuring will naturally produce a coupled system if the interior nodes

of the subsquares are interpreted as interior variables and the common nodes as coupling variables. We treated the parameter σ (the so-called Thiele modulus) as a further coupling variable and introduced the equation

$$u(0.5, 0.5) \quad = \quad u_{\max} \tag{81}$$

with a prespecified $u_{\max}$ to compensate for it. As the name suggests, $u_{\max}$ will be the maximum value of $u(\xi_1, \xi_2)$; technically, however, it is just a parameter of our problem. By separating the diagonal part of the discretized Laplacian from the rest of the equations, we constructed an obvious Jacobi type iteration process Φ. It is known that this Φ will be contractive as long as $u_{\max}$ is small enough. (The substructuring of the square allows for larger values than would be admissible without it.)

For the following computations we used $(2 \cdot 7 + 1)^2 = 225$ interior grid points and set $u(0.5, 0.5) = 8$. The following figures are supposed to demonstrate the behaviour of the method as well as some specifics of the control mechanism.

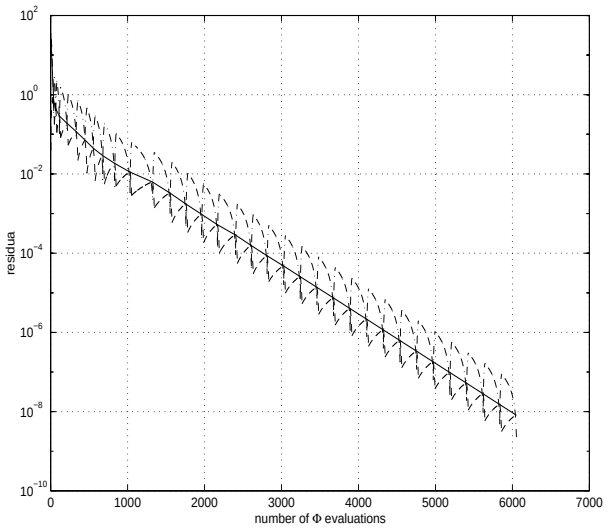

Figure 2: ATBN at $\varepsilon_1 = 0.1$

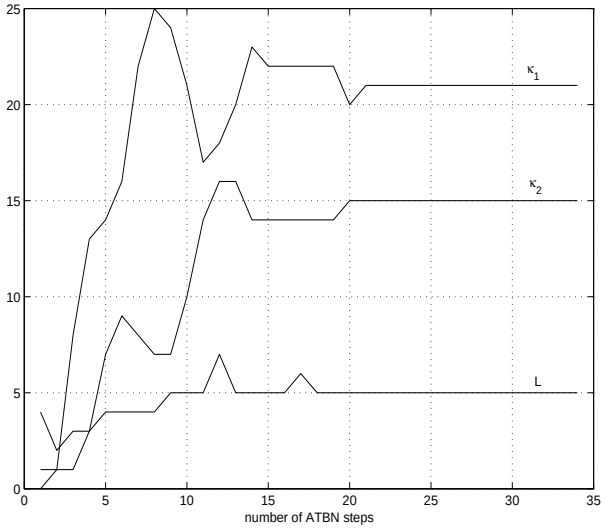

Figure 3: κ_1, κ_2 and ℓ for $\varepsilon_1 = 0.1$

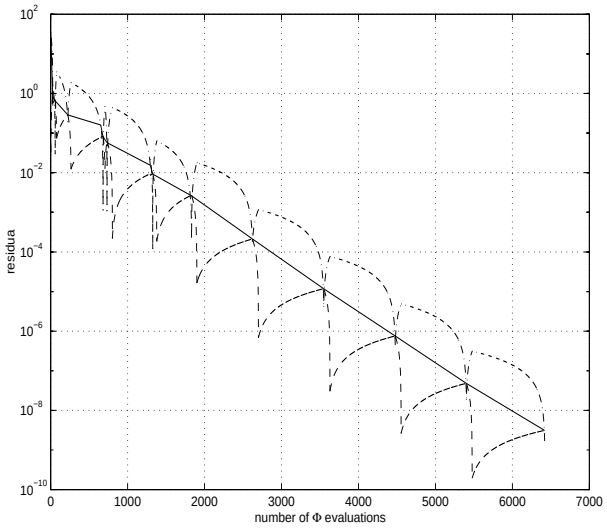

Figure 4: ATBN at $\varepsilon_1 = 0.01$

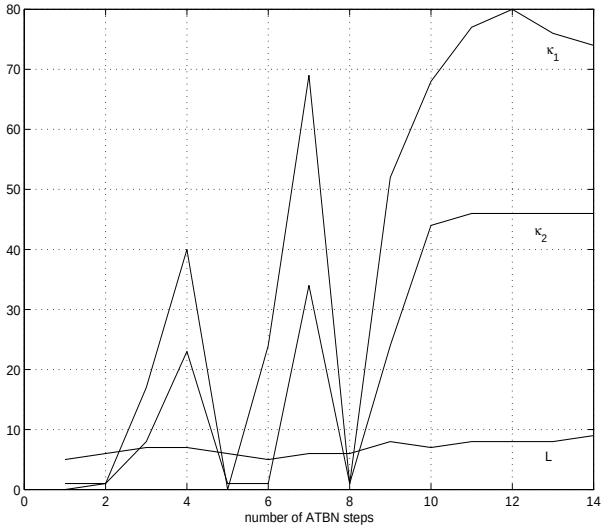

Figure 5: κ_1, κ_2 and ℓ for $\varepsilon_1 = 0.01$

The residua plots are to be read as follows:

- The horizontal axes represent the numbers of Φ evaluations used. The vertical axes represent the residual norms.

- The dashed lines indicate the values of $\|f\|$, the dash-dotted ones the values of $\|g\|$. They interpolate linearly between the values at three critical phases of the composite ATBN step: a) before the f step, b) before the g step, c) after a virtual *undamped* g step. It will become clear why this makes sense.

- The solid lines connect the values of $\max(\|f\|, \|g\|)$ before and after the actual composite ATBN step. They can be used to measure the success of the step: the steeper the slope, the better the step.

Figures 2 and 3 demonstrate the overall behaviour of the method for $\varepsilon_1 = 10^{-1}$. The steps are relatively small; after a startup phase the method stabilizes itself at $\kappa_1 = 21 \pm 1$ and $\kappa_2 = 15 \pm 1$. As we assumed, the number ℓ of BiCGStab iteration steps remains almost constant after the startup phase.

Figures 4 and 5 show how the control mechanism adapts to a reduction of ε_1 : Since the quality of the g step will improve and since the costs of solving (22) will increase, the mechanism decides to improve the f step and the quality of the tangential directions as well. The overall costs of the step actually hardly increase: The solid lines in figures 4 and 2 both meet the level 10^{-8} at approximately 6000 Φ iterations. The control mechanism will always provide some sort of compensation for a maladjusted ε_1 but there can be no guarantee that it will always be as successful as in this example; it is of course preferable not to choose ε_1 unnecessarily small in the first place.

It is perhaps worth noting that the slightly odd oscillations of the κ_i between very small and relatively large values during the startup phase is due to the fact that the estimated effective reduction associated with (κ_1, κ_2) tends to have two separate local minima. Due to changes in the estimates of λ, μ and q and gaps between $\|f_n\|$ and $\|g_n\|$ it is not clear from the outset which one will dominate in the long run. See figures 6 and 7 for the model estimates of the effective reduction associated with different combinations of κ_i for steps 7 and 8 in figure 5, respectively.

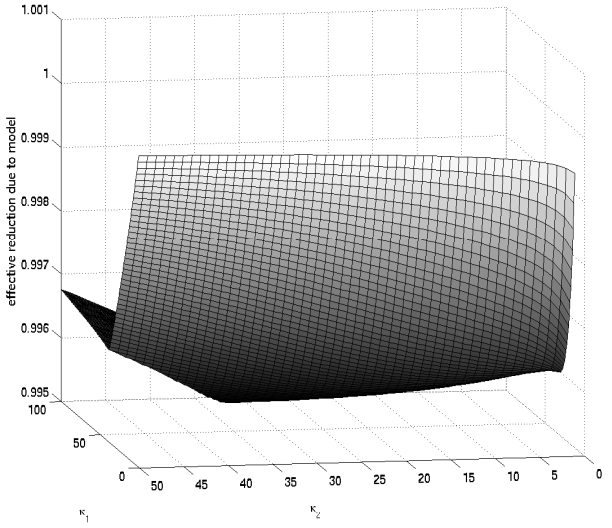

Figure 6: model of step # 7, $\varepsilon_1 = 0.01$

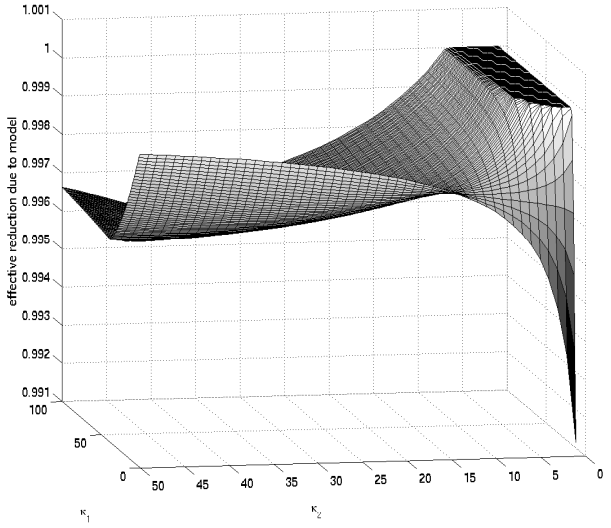

Figure 7: model of step # 8, $\varepsilon_1 = 0.01$

Finally, let us take a closer look at one isolated step of the method: Figures 8 and 9 demonstrate the influence of the κ_i on the performance of the step. The value of ε_1 was 10^{-2}, and the model had suggested to use $\kappa_1 = 66$ and $\kappa_2 = 47$. For figure 8 we accepted κ_2 and varied κ_1, for figure 9 vice versa. We always used the optimal β associated with the actual κ_i values.

First of all, both figures clearly reflect the fact that the f step is typically much more expensive than the g step. This should be no surprise because the g step always involves computing tangential corrections. Since the quality of these corrections is governed by the value of κ_2, the influence of this parameter on the computational costs of the ATBN step is much stronger than the influence of κ_1. Nevertheless it is still not wise to choose κ_1 too large: The amplification of $\|f\|$ during the g step depends crucially on $\|g\|$, and thus a disproportionately small value of $\|f^+\|$ will only be maintained if $\tilde{C}$ is very accurate. The loss of efficiency caused by an oversized κ_1 will not be too grave though, owing to the fact that the f step will remain relatively cheap (cf. figure 8, $\kappa_1 = 150$). Conversely, a small κ_1 can be very annoying because it will prevent exploiting the qualities of κ_2 and ε_1, which were typically rather expensive to achieve (cf. figure 8, $\kappa_1 = 30$).

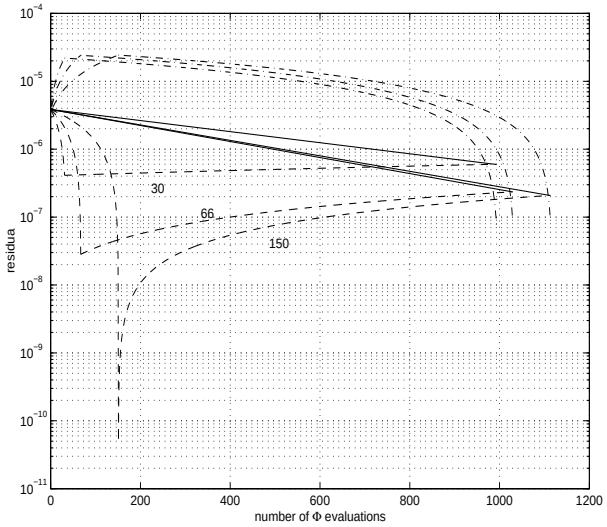

Figure 8: varying κ_1 in an ATBN step

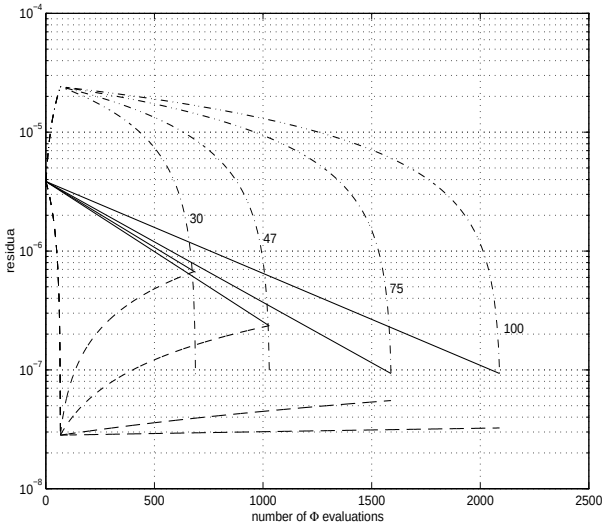

Figure 9: varying κ_2 in an ATBN step

To a certain extent, figure 9 mirrors figure 8 in that increasing κ_1 corresponds to decreasing κ_2 and vice versa. If κ_2 is too small, the g step will partly spoil the success of the f step (cf. figure 9, $\kappa_2 = 30$), if κ_2 is too large, the quality of the resulting tangential corrections can not be fully exploited. Note that figure 9 shows ε_1 to become the main impediment for large κ_2 : In the cases of $\kappa_2 = 75$ and $\kappa_2 = 100$ the ATBN step clearly is hampered by the limited ability of the g step to reduce $\|g^+\|$, that is by the value of ε_1 .

Acknowledgements. The author thanks Wolfgang Mackens for his fruitful suggestions and his helpful criticism.

References

[1] Aris, R.: *The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts, Vol.1: The Theory of the Steady State*, Clarendon Press: Oxford (1975).

[2] Artlich, S.: *Zweidimensionale Simulation der Kohleverbrennung in Druckwirbelschichtfeuerungen*, Dissertation, TU Hamburg-Harburg 1996, VDI-Verlag, Reihe 6, Nr. 346: Düsseldorf (1996).

[3] Artlich, S.: Combustion of Coal in Pressurized Fluidized Bed Reactors, in *Scientific Computing in Chemical Engineering*, F. Keil, W. Mackens, H. Voß, Werther, J. (eds), Springer Verlag: Heidelberg (1996).

[4] Artlich, S. and Mackens, W.: Newton-Coupling of Fixed Point Iterations, in *Numerical Treatment of Coupled Systems*, W. Hackbusch and G. Wittum (eds), Vieweg-Verlag: Braunschweig, Wiesbaden (1995), 1–10.

[5] AspenTech: *SPEEDUP User Manual*, Aspen Technology Inc., Cambridge, MA (1990).

[6] Blomgren, P. and Chan, T.F.: *Modular Solvers for Constrained Image Restoration Problems*, U.C.L.A. Computational and Applied Mathematics Report 97-52, University of California, Los Angeles (1997).

[7] Chan, T.F.: An Approximate Newton Method for Coupled Nonlinear Systems, *SIAM J. Numer. Anal.* **22** (1985), 904–913.

[8] Chan, T.F.: An Efficient Modular Algorithm for Coupled Nonlinear Systems, in *Numerical Analysis. Proceedings of the Fourth IIMAS Workshop held at Guanajuato, Mexico, July 23–27, 1984*, J.P. Hennart (ed), *Springer Lect. Notes in Math.* **1230**, Springer-Verlag: Berlin (1986), 73–85.

[9] Dennis, J.E. and Schnabel, R.B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* Prentice-Hall, Inc.: Englewood Cliffs (1983).

[10] Freund, R.W.: A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems, *SIAM J. Sci. Comput.* **14** (1993), 470–482.

[11] Greenbaum, A.: Any Nonincreasing Convergence Curve is Possible for GMRES, *SIAM J. Matrix Anal. Appl.* **17** (1996), 465–469.

[12] Greenbaum, A.: Estimating the Attainable Accuracy of Recursively Computed Residual Methods, *SIAM J. Matrix Anal. Appl.* **18** (1997), 535–551.

[13] Gutknecht, M. H.: *Lanczos-Type Solvers for Nonsymmetric Linear Systems of Equations*, Technical Report TR-97-04, CSCS/SCSC, ETH Zürich (1997).

[14] Hoyer, W. and Schmidt, J. W.: Newton-Type Decomposition Methods for Equations Arising in Network Analysis, *ZAMM* **64** (1984), 397–405.

[15] Hoyer, W., Schmidt, J. W. and Shabani, N.: Superlinearly Convergent Decomposition Methods for Block-Tridiagonal Nonlinear Systems of Equations, *Numer. Funct. Anal. and Optimiz.* **10** (1989), 961–975.

[16] Lanzkron, P. J., Rose, D. J. and Wilkes, J. T.: An Analysis of Approximate Nonlinear Elimination, *SIAM J. Sci. Comput.* **17** (1996), 538–559.

[17] Mackens, W.: *Some Notes on Block-Gauss-Seidel Newton Iterations for the Solution of Sparse Nonlinear Systems*, Bericht Nr. 37 des Instituts für Geometrie und Praktische Mathematik der RWTH Aachen (1986).

[18] Mackens, W.: *Quadratic Convergence of the Recursive Block-Gauss-Seidel-Newton Iteration*, Bericht Nr. 44 des Instituts für Geometrie und Praktische Mathematik der RWTH Aachen (1987).

[19] Nachtigal, N. M., Reddy, S.C. and Trefethen, L.N.: How Fast are Nonsymmetric Matrix Iterations? , *SIAM J. Matrix Anal. Appl.* **13** (1992), 778–795.

[20] Saad, Y. and Schultz, M. H.: GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Statist. Comput.* **7** (1986), 856–869.

[21] Schmidt, J. W., Hoyer, W. and Hauffe, C.: Consistent Approximation in Newton-Type Decomposition Methods, *Numer. Math.* **47** (1985), 413–425.

[22] Sleijpen, G. L. G. and Fokkema, D. R.: BiCGstab(ℓ) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum, *Electronic Trans. Numer. Anal.* **1** (1993), 11–32.

[23] Sonneveld, P.: CGS, a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems, *SIAM J. Sci. Statist. Comput.* **10** (1989), 36–52.

[24] van der Vorst, H. A.: Bi-CGstab: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. Statist. Comput.* **13** (1992), 631–644.

[25] Zhang, X., Byrd, R.H. and Schnabel, R.B.: Parallel Methods for Nonlinear Block Bordered Systems of Equations, *SIAM J. Stat. Comput.* **13** (1992), 841–859.