Coupling Iterative Subsystem Solvers

Wolfgang Mackens, Jürgen Menck, and Heinrich Voss

Technische Universität Hamburg, Arbeitsbereich Mathematik, Kasernenstraße 12, D-21073 Hamburg,{mackens, menck, voss}@tu-harburg.de, http://www.tu-harburg.de/mat/

Abstract. We sketch an algorithmic framework to integrate (iterative) solvers of subsystems of a large nonlinear system into a joint iterative solution process.

1 Introduction

Complex technical systems are often assembled from already existing components. The idea suggests itself to similarly synthesize numerical simulators of compound systems from available simulators of subsystems. This facilitates utilising knowledge about the subsystems and it spares the designer expensive revalidations of the subsystems simulators.

Chemical Engineering seems to be the first scientific field where this integrational idea has been translated into action. There are quite some flow sheeting programms which couple numerical models of unit operations to simulate whole production plants: ASPEN+, SPEEDUP, gPROMS and PRO/II are just four of the better known systems.

Block-oriented simulators (such as ASPEN+ and PRO/II) reduce the interactions of the unit operations to relatively few input/output variables. This is accomplished either by simplifying the relations between the coupling variables or by implementing more complicated ones through the solution of internal subproblems. In the first case an overall middling modeling quality may result, in the second case the hidden subsystems may lead to large runtimes.

Equation oriented simulators (like SPEEDUP and gPROMS) avoid these problems by collecting all involved equations and applying subtle Newton type iterations to the joint system. If applicable this approach will certainly exhibit near optimal computing expenses. However, the equations of a simulator may well not be accessible, or it may not be desirable to collect them.

In this paper we discuss how to solve the system using only the existing subsystem simulators. We assume that these are iterative in their inner variables, the results of single iteration steps are available and the subsystems are complemented by a set of coupling equations. Newton's iteration for the full system serves as the guiding principle.

2 Limitations of Jacobi and Gauss-Seidel couplings

Even in medium size problems it is common to investigate substructures individually first with the required data from the other parts assumed known. As a small case study we consider the numerical computation of the stationary distributions of mass (y) and heat (θ) inside a porous catalyst particle. The time depending 1D modeling equations as taken from [8] read

$$\frac{\partial \theta}{\partial t} - \frac{\partial^2 \theta}{\partial x^2} = \delta y \exp\left(\frac{\theta}{1+\theta/\gamma}\right), \tag{1a}$$

$$\mathrm{Lw}\frac{\partial y}{\partial t} - \frac{\partial^2 y}{\partial x^2} = -\frac{\delta}{\gamma\beta} y \exp\left(\frac{\theta}{1+\theta/\gamma}\right), \tag{1b}$$

($\mathrm{Lw} = 5.5, \gamma = 20, \beta = 0.2, \delta \in (0,2)$) with boundary conditions

$$y(-1) = y(1) = 1, \qquad \theta(-1) = \theta(1) = 0. \tag{2}$$

Discretization with finite differences on a uniform grid of step size h (we always chose $h = 1/20$ in our examples) of the stationary equations yields

$$A_h \theta_h = \delta F(y_h, \theta_h), \tag{3a}$$

$$A_h y_h = \delta G(y_h, \theta_h), \tag{3b}$$

where the matrix A_h is a discrete second derivative operator and F and G are discretizations of the right hand sides of (1) including the boundary values (2). (See [1] e.g., for a thorough explanation of the discretization process).

A popular way to attack (3a) (with y_h assumed given) is Picard's iteration

$$\theta_h^{n+1} := \delta A_h^{-1} F(y_h, \theta_h^n) =: U[y_h, \delta](\theta^n). \tag{4}$$

For small δ and y this converges to a positive solution $\theta(\delta, y_h)$. For equation (3b) one could of course use Picard's iteration, too:

$$y_h^{n+1} := \delta A_h^{-1} G(y_h^n, \theta_h) =: V[\theta_h, \delta](y^n). \tag{5}$$

But since equation (3b) is linear with respect to y a direct solution might appear more appropriate.

To solve both equations simultaneously, engineers tend to favour a nonlinear Jacobi- or Gauss-Seidel approach:

$$\theta_h^{n+1} = U[y_h^n, \delta]^{\kappa_1}(\theta_h^n), \tag{6a}$$

$$y_h^{n+1} = V[\theta_h^{n(+1)}, \delta]^{\kappa_2}(y_h^n) \tag{6b}$$

(We use $\kappa_1 = \kappa_2 = 1$ in our examples). It is an even more common habit to solve the equations exactly in turns:

$$\text{solve } A_h \theta_h^{n+1} = \delta F(y_h^n, \theta_h^{n+1}) \text{ for } \theta_h^{n+1}, \tag{7a}$$

$$\text{solve } A_h y_h^{n+1} = \delta G(y_h^{n+1}, \theta_h^{n+1}) \text{ for } y_h^{n+1}. \tag{7b}$$

Of course it is also possible to mix both variants by combining (6a) with (7b) or (6b) with (7a).

Many users believe that these hybrid methods will always converge, at least if the steps are suitably damped. However, this belief is erroneous: Figure 1 shows the contraction rates of the optimally damped iterations (6a,6b), (7a,7b) and (6a,7b) as functions of the parameter δ.

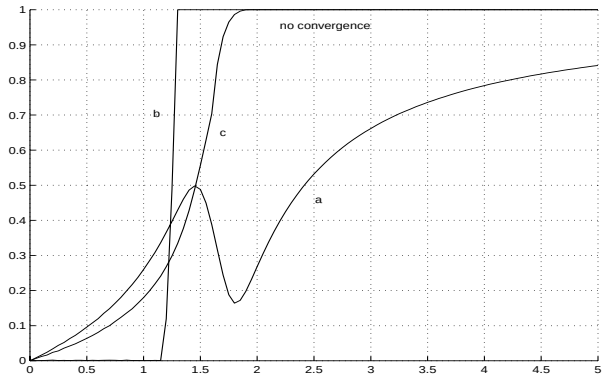

Fig. 1. Contractions for damped a =(6a,6b), b =(7a,7b), c =(6a,7b)

The figure clearly shows that both (7a,7b) and (6a,7b) fail to converge for large values of δ. Ironically, the cheapest of the methods, (6a,6b), remains convergent in this situation, which proves that perfecting the individual solution steps can even have a damaging effect on the performance of a hybrid method.

We might add that the breakdown of convergence just observed is not even triggered by dynamical instabilities of the underlying time dependent equations (1a,1b,2). There *is* a connection as long as time stepping methods for this system are used. This, however, will not be desirable due to the poor performance of these methods.

3 Newton type coupling

3.1 Problem specification

We assume that $k \in \mathbb{N}$ subsystems are given by k iterative solvers $x_i^{n+1} := \Phi_i(x_i, y)$ in their internal variables $x_i \in \mathbb{R}^{k_i}$ and a common set $y \in \mathbb{R}^m$ of "coupling variables". These systems are accompanied by an additional coupling equation $g(x_1, \ldots, x_k, y) = 0$, with $g : \mathbb{R}^K \times \mathbb{R}^m \to \mathbb{R}^m$, $K := \sum_{i=1}^k k_i$, such that the total system of equations to be solved,

$$0 = f_i(x_i, y) := x_i - \Phi_i(x_i, y), \ i = 1, \ldots, k, \tag{8}$$

$$0 = g(x_1, \ldots, x_k, y), \tag{9}$$

is square. We assume that the system is sufficiently smooth. Normally, continuous derivatives up to order two will do.

The appearence of several f_i and Φ_i models system inherent parallelism. Since parallelism is not our central subject here (cf. [2], however) we merge the iterations into a joint iteration $x^{n+1} := \Phi(x^n, y)$ with $x = (x_1, \ldots, x_k)$ and $\Phi = (\Phi_1, \ldots, \Phi_k)$. With $f(x, y) := x - \Phi(x, y)$ we shall henceforth consider the system

$$\begin{aligned} f(x, y) &= 0, \\ g(x, y) &= 0. \end{aligned} \tag{10}$$

Note that a direct solver for the i-th system $x_i := \Phi_i(y)$ is nothing but a very fast iterative solver. Hence direct solvers can be incorporated in the setting, too.

3.2 The tangential block Newton Iteration TBN

The algorithmic structure to be described now is in principle known and has been discovered and rediscovered several times (cf. [4] for references). Our starting point is the desire to have some Newton-flavoured iteration for the blocked system (10). An obvious idea would be to use the Gauss-Seidel-Newton iteration ([6]), see figure 2. It is clear from this figure (think of the linear case) that convergence will depend very much on the geometry of the solution manifolds of f and g.

Under natural smoothness and regularity assumptions Block-Gauss-Seidel-Newton does converge if $f_y(x^*, y^*) = 0$. This condition means that the solution manifold of $f(x, y) = 0$ near (x^*, y^*) is parallel to the direction of the second partial iteration step. This geometrical view leads to an extension of the method to the general case: We restrict g to the tangential space of the solution manifold of $f(x, y) = f(x^+, y^n)$ at (x^+, y^n), see fig. 3. Through implicit differentiation of $f(x(y), y)) = f(x^+, y^n)$ with respect to y one finds that this space is spanned by $T = \begin{pmatrix} -f_x^{-1} f_y \\ I_m \end{pmatrix} =: \begin{pmatrix} -\boldsymbol{C} \\ I_m \end{pmatrix} \in \mathbb{R}^{(K+m,m)}$. We can thus compute Δy from the Newton step ansatz

$$0 = g(x^+ - \boldsymbol{C}\Delta y, y^n + \Delta y) \approx g(x^+, y^n) + \underbrace{(-g_x \boldsymbol{C} + g_y)}_{=:\boldsymbol{S}} \Delta y.$$

Thus we end up with the

Tangential-Block-Newton Iteration:

$$\begin{aligned} &\textbf{(A)} \quad x^+ := x^n - f_x^{-1} f(x^n, y^n), \\ &\textbf{(B}_1\textbf{)} \quad \boldsymbol{C} := f_x^{-1} f_y, \\ &\textbf{(B}_2\textbf{)} \quad \boldsymbol{S} := g_y - g_x \boldsymbol{C}, \\ &\textbf{(B}_3\textbf{)} \quad \boldsymbol{S}\Delta y = -g(x^+, y^n), \\ &\textbf{(E)} \quad \begin{pmatrix} x^{n+1} \\ y^{n+1} \end{pmatrix} = \begin{pmatrix} x^n \\ y^n \end{pmatrix} + \begin{pmatrix} -\boldsymbol{C} \\ \boldsymbol{I}_m \end{pmatrix} \Delta y. \end{aligned}$$

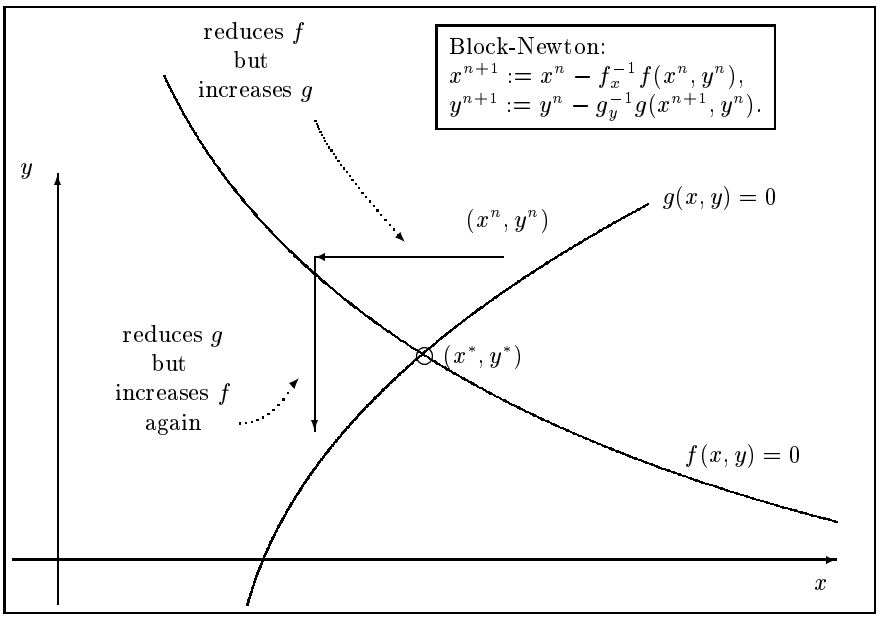

Fig. 2. Simple Gauss-Seidel Block-Newton Iteration

By interpreting TBN as a perturbation of the usual Newton step, one can prove [4] that *under standard assumptions TBN is locally quadratically convergent.*

3.3 Fixed point realization of TBN

The obvious problem in applying TBN to our setting is the need to access the matrices f_x^{-1}, C, and S. An idea for appropriate substitutes is rather simple in principle, however.

First observe that the purpose of f_x^{-1} in step (A) of TBN is to help approximate the solution of $f(x, y_n) = 0$ starting from (x_n, y_n). This task can of course be taken over by the available fixed point iteration.

One would hence replace the step (A) by the iterative variant $x^+ := \Phi^{\kappa_1}(x_n, y_n)$. Here, $\kappa_1 \geq 1$ is a local iteration count that can be used to control the quality of the approximation. To come up with an approximation for C, multiply $d = f_x^{-1} q$ by $f_x = I - \Phi_x$ giving the fixed point equation $d = \Phi_x(x^n, y^n)d + q$. Approximate $\Phi_x(x^n, y^n)d$ with a suitable differencing step size h by

$$\Phi_x(x^n, y^n)d \approx h^{-1} \left(\Phi(x^n + hd, y^n) - \Phi(x^n, y^n)\right).$$

Finally use this to replace Φ_x in the fixed point equation:

$$d = h^{-1} \left(\Phi(x^n + hd, y^n) - \Phi(x^n, y^n)\right) + q. \tag{11}$$

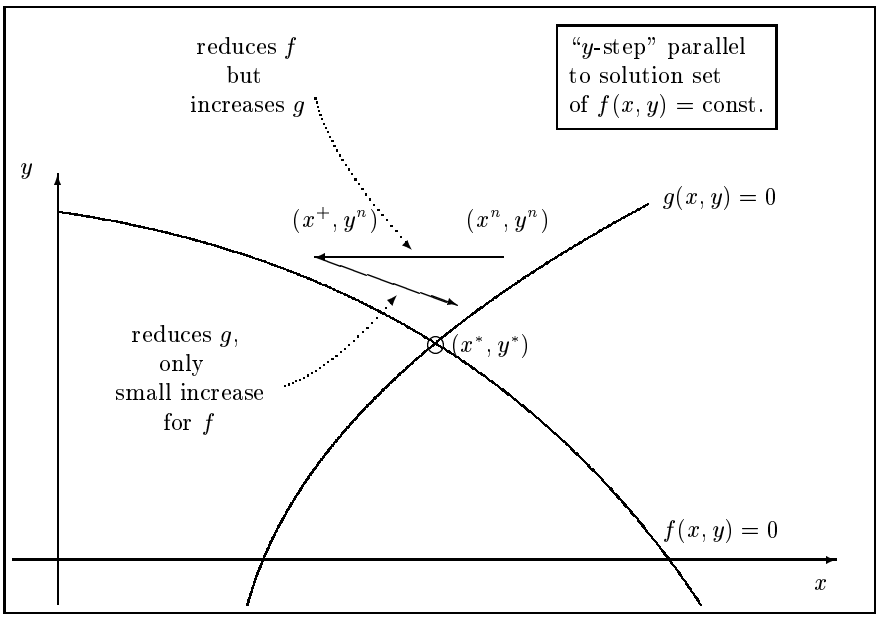

Fig. 3. Tangential Block-Newton Iteration

With the iterative approach we are now in the position to approximate "components" of C, i.e. vectors Cp. Thus the whole TBN-Iteration can be performed approximately by using only the given f solver.

The template of figure 4 gives an algorithmic framework into which all the variants to be dealt with will fit.

Approximate TBN Template:

(A') $x^+ := x^n + \alpha(\Phi^{\kappa_1}(x^n, y^n) - x^n)$
(B') Determine an approximate Δy
from an approximate treatment of
$\boldsymbol{C} := f_x^{-1} f_y,$
$\boldsymbol{S} := g_y - g_x \boldsymbol{C},$
$\boldsymbol{S}\Delta y = -g(x^+, y^n),$

(C_1') $y^{n+1} := y^n + \beta \Delta y,$
(C_2') Adapt x^+ by approximate execution of
$x^{n+1} := x^+ - \beta \boldsymbol{C} \Delta y.$

(L) Adjust damping parameters α, β adequately.

Fig. 4. Approximate Tangential Block Newton: Template

3.4 Subprocedures of ATBN

The following procedures can be useful in turning the ATBN template into a fleshed-out algorithm:

Directional differencing for C Use the iteration formula (11) from the previous section to compute matrix-vector products involving C.

Explicit calculation of C If the problem is not too large one might wish to assemble matrix C explicitly. This can be achieved by computing the columns of C as Ce^i via directional differencing.

Directional differencing for S Note that matrix-vector products Sp can be interpreted as directional derivatives of g into the direction $(-(Cp)^T, p^T)^T$. Thus Sp can be computed according to

$$Sp \approx \tilde{S}p := h^{-1}\left(g(x - h\tilde{C}p, y + hp) - g(x, y)\right), \tag{12}$$

where $\tilde{C}p$ may already be an approximation of Cp in itself.

Explicit calculation of S If m (and not necessarily K) is small, one might wish to assemble matrix S. This can be achieved by computing the columns of S as Se^i via directional differencing.

Quasi-Newton approach to Δy-computation The idea of using a Quasi-Newton approach for the computation of Δy has already been persued in [3] and proven to work quite well if the starting approximations are sufficiently good. The combination y step / S update reads (for the undamped case)

$$\begin{aligned} S\Delta y &= -g(x^+, y^n); \\ y^{n+1} &:= y^n + \Delta y; \\ x^{n+1} &:= x^+ - \tilde{C}\Delta y; \\ S^+ &:= S + \frac{g(x^{n+1}, y^{n+1})(\Delta y)^T}{(\Delta y)^T \Delta a}. \end{aligned}$$

Here $\tilde{C}$ stands for an exact or approximate application of the C operation.

Matrix-free solution of Schur complement equation If m is large, it may be a good idea to compute Δy by solving $S\Delta y = -g^+$ via a transpose free generalized conjugate gradients iteration for nonsymmetric problems, such as Bi-CGStab or GMRES (cf. [5] for a short overview of suitable methods). The only access to S will then be through matrix vector products Sq which can be computed through directional differencing.

x^+-adaption to $C\Delta y$-update of y After having updated $y^n \to y^n + \beta\Delta y$ one has to adapt x^+ to give $x^{n+1} = x^+ - \beta C\Delta y$. There are again several possibilities to perform (an approximation to) this task, two of them being directional differencing and explicit multiplication by the matrix C.

4 A short example

The iterations (4) and (5) are adapted to the framework of Section 3.1 by letting $x := \theta_h, y := y_h, \Phi(x,y) := U[y_h,\delta](\theta_h)$ and $g(x,y) := y_h - V[\theta_h,\delta](y_h)$. Clearly, the iterative potential of (5) is not brought to bear. At the price of doubling the problems size this can be enabled through

$$\begin{array}{ll} x_1 := \theta_h, x_2 := y_h, & y_1, y_2 = \text{ auxiliary variables,} \\ \Phi_1(x_1,x_2,y_1,y_2) := U[y_2,\delta](x_1), & \Phi_2(x_1,x_2,y_1,y_2) := V[y_1,\delta](x_2), \\ g_1(x_1,x_2,y_1,y_2) := y_1 - x_1, & g_2(x_1,x_2,y_1,y_2) := y_2 - x_2. \end{array}$$

Numerical results can be found in [4].

5 Final remarks

A practical implementation of ATBN-iteration involves further ingredients: Breakdown of contractions of the Φ_i-iterations can be cured by a stabilization as described in [2]. Global convergence is obtained by subspace search methods [7]. Good local convergence is accomplished by the methods of [5].

References

1. Erich Bohl: *Finite Modelle gewöhnlicher Randwertaufgaben.* Teubner Studienbücher Mathematik, Stuttgart 1981
2. Uwe Kleis: Parallel Solution of Diffusion Reaction problems with a stabilization method. pp. ??? – ??? in this Volume of this Proceedings
3. Wolfgang Mackens: Quadratic convergence of the Recursive Block-Gauss-Seidel-Newton iteration, Bericht Nr. 44 des Instituts für Geometrie und Praktische Mathematik der RWTH Aachen, Januar 1987
4. Wolfgang Mackens, Jürgen Menck, and Heinrich Voss: Numerical System Synthesis: Concepts for Coupling Subsystem Solvers, Report ??, Section of Mathematics, Technical University Hamburg-Harburg, 1998
5. Jürgen Menck: Work control for Newton type coupling, pp. ??? – ??? in this Volume of this Proceedings.
6. J. M. Ortega and W. C. Rheinboldt: *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, Boston 1970
7. Richard Rascher-Friesenhausen: Subspace search methods for large scale nonlinear optimizations. pp. 183-189 in Keil, Mackens, Voß, Werther (eds.): Scientific Computing in Chemical Engineering, Springer 1996
8. D. Roose and V. Hlavacek: Numerical Computation of Hopf bifurcation points for parabolic diffusion-reaction differential equations, SIAM J. Appl. Math. 5 (1983) 1075 - 1085