

AIAA 98-4712

**Acceleration of Multidisciplinary
Analysis Solvers by Inexact
Subsystem Simulations**

Matthias Heinkenschloss

Mary Beth Hribar

Michael Kokkolaras

Rice University

Houston, Texas 77005-1892

**7th AIAA/USAF/NASA/ISSMO Symposium on
Multidisciplinary Analysis and Optimization
September 2-4, 1998/ St. Louis, MO**

Acceleration of Multidisciplinary Analysis Solvers by Inexact Subsystem Simulations

Matthias Heinkenschloss*

Mary Beth Hribar†

Michael Kokkolaras‡

Rice University

Houston, Texas 77005-1892

This paper is concerned with the numerical solution of systems of blocked nonlinear equations arising in the solution of Multidisciplinary Analysis (MDA) problems. We consider the case where individual discipline solvers/simulators are given and are iterative methods. Thus, an MDA solver consists of an outer iteration for the solution of the system of blocked nonlinear equations and of inner iterations in the discipline simulators. We will show how the control of the truncation of the inner iterations can be effectively used to accelerate the overall iteration. The key is the interpretation of the outer iteration with inexact inner iteration as an iteration of a related system with same solution as the MDA problem.

Introduction

Multidisciplinary Analysis (MDA) problems link mathematical models from more than one discipline. We consider MDA problems where each discipline is described by a system of nonlinear equations, and where the output of each discipline is computed by solving this system iteratively. We call the computation required for each discipline a subsystem simulation. In this paper we consider the case where these subsystem simulations already exist and we are interested in numerical methods for linking these subsystems.

The overall MDA problem is then formulated as a large scale system of blocked nonlinear equations, where each block corresponds to the output of one discipline. A solver for the MDA problem will involve outer iterations of the method for the overall system of blocked equations, and inner iterations performed within the subsystem simulations. There will be inexactness in the subsystem simulations from the truncation errors of iterative methods, causing inexactness in the residual and derivative computations of the outer iteration. In general, inexact residual evaluations are a curse for nonlinear system solvers. In this paper, however, we will show that the type of inexactness in the inner iterations described previously

can be used to significantly improve the performance of the MDA solver.

In this paper, we will concentrate on the case where Newton's method is used in the inner iterations. We consider applying standard methods for nonlinear systems of equations to this case and show how to exploit the inexactness in the inner iteration. The theoretical results will then be illustrated using a numerical example of an MDA problem.

Problem Formulation

We denote the output of discipline i by $x_i \in \mathbb{R}^{n_i}$, $i = 1, \dots, m$. Given the outputs x_1, \dots, x_{i-1} , x_{i+1}, \dots, x_m of the other disciplines, x_i is computed as the solution of the implicit equation

$$G_i(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_m) = 0. \quad (1)$$

Often (1) represents a discretized partial differential equation that has to be solved for x_i . Under the assumptions of the Implicit Function Theorem, (5) defines a function

$$x_i = S_i(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m). \quad (2)$$

The function S_i represents the solver or simulator of the i th discipline. The linked subsystems can be now be described as follows: Find an $x = (x_1, x_2, \dots, x_m) \in \mathbb{R}^n$ such that

$$F(x) = x - S(x) = 0, \quad (3)$$

where

$$x - S(x) = \begin{pmatrix} x_1 - S_1(x_2, \dots, x_m) \\ x_2 - S_2(x_1, x_3, \dots, x_m) \\ \vdots \\ x_m - S_m(x_1, \dots, x_{m-1}) \end{pmatrix} \quad (4)$$

*Associate Professor, Department of Computational and Applied Mathematics

†Research Scientist, Department of Computational and Applied Mathematics

‡Research Scientist, Department of Mechanical Engineering and Materials Science

Copyright © 1998 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

and $n_1 + \dots + n_m = n$. We define the MDA problem to have the form of the system stated in (3).

Obviously, the system (3) is related to

$$G(x) = \begin{pmatrix} G_1(x_1, x_2, \dots, x_m) \\ G_2(x_1, x_2, \dots, x_m) \\ \vdots \\ G_m(x_1, x_2, \dots, x_m) \end{pmatrix} = 0. \quad (5)$$

In fact, a solution of (3) is also a solution of (5). The other direction may not necessarily be true. The solution method for (1) usually returns a specific solution, but not the entire set of solutions. Thus, (5) may have multiple solutions and not all of these may solve (3). It is also easy to construct examples where (5) has a solution but (3) does not. However, this relation between (3) and (5) is still important.

The residual evaluations in (5) are usually cheap compared to the residual evaluations in (3). In many practical applications, however, the exact form of (5) is not known. This is typically the case when the individual disciplines are first simulated independently and coupled later. In this case the individual disciplines (2) are solved by a black-box and are coupled via (3).

The individual discipline simulations are often done iteratively and therefore the subsystem simulations (2) are inexact due to truncation error. Thus, instead of the function S_i we can only access a function

$$S_i^{(\nu_i)}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m). \quad (6)$$

Here (ν_i) indicates the inexactness. It will be specified later. Consequently, if a nonlinear equation solver is applied to (3), it can not access the exact residual $F(x)$, but only

$$\begin{aligned} F^{(\nu)}(x) &= x - S^{(\nu)}(x) \\ &= \begin{pmatrix} x_1 - S_1^{(\nu_1)}(x_2, \dots, x_m) \\ x_2 - S_2^{(\nu_2)}(x_1, x_3, \dots, x_m) \\ \vdots \\ x_m - S_m^{(\nu_m)}(x_1, \dots, x_{m-1}) \end{pmatrix}. \end{aligned} \quad (7)$$

It is important to be aware of methods used to compute the individual disciplines S_i and not to treat them as merely a black box. In this paper we are concerned with MDA problems with this blocked and inexact form in which the individual disciplines S_i are computed by applying Newton's method to solve (1) for x_i .

Adapting Methods for Coupled Systems of Nonlinear Equations

Given a system of nonlinear equations, there are many standard methods for solving this problem. In this paper, we will consider Gauss-Seidel-type

methods,¹ Newton type methods^{2,3} and Broyden's method.^{2,3} They will be discussed in the context of the MDA problem (3). However, instead of solving the system defined in (3), we solve the inexact system defined in (7) where Newton methods (8) are used to solve the discipline simulations. We compute $S_i^{(\nu_i)}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m)$ by a Newton method as follows:

$$\left. \begin{aligned} &\text{Given } z_i^0. \\ &\text{For } l = 0, 1, 2, \dots \\ & \quad z_i^{l+1} = z_i^l - \left(\frac{\partial}{\partial x_i} G_i(x_1, \dots, z_i^l, \dots, x_m) \right)^{-1} \\ & \quad \quad \quad \times G_i(x_1, \dots, z_i^l, \dots, x_m). \\ &\text{When a stopping criteria is satisfied STOP} \\ &\text{Set } S_i^{(\nu_i)}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m) = z_i^{l+1}. \end{aligned} \right\} \quad (8)$$

At the termination of this Newton iteration, there is some level of inexactness in the solution. We will show that a suitable and practical control of this kind of inexactness may be used to significantly speed up the overall solution process. Our presentation in this section will be expository and it will be limited to rather simplified scenarios. The technical, but important details needed to make these ideas robustly applicable to a wider range of problems and the technical convergence proofs will be presented in a forthcoming paper.

Gauss-Seidel-Type Methods

Of the nonlinear equation solvers for (3) the sequential substitution is the easiest to implement. It does not require calculations of derivatives of S_i , it preserves the structure of the problem, and if discipline simulations are available, it is easy to implement. Given $x^k = (x_1^k, \dots, x_m^k)$ one step of the sequential substitution for (3) computes

$$x_i^{k+1} = S_i(x_1^{k+1}, \dots, x_{i-1}^{k+1}, x_{i+1}^k, \dots, x_m^k), \quad (9)$$

for $i = 1, \dots, m$. This iteration is the nonlinear Gauss-Seidel method applied to the system (5). See the book.¹ The observation that the iterative method for (3) can in fact be interpreted as an iterative method for (5) offers interesting possibilities to derive more efficient solution methods.

Suppose that the i th discipline is evaluated by the Newton method (8) with starting value $z_i^0 = x_i^k$. We assume that in this context ν_i represents the number of Newton iterations performed, i.e., in (8) $l+1 = \nu_i$. The exact solution of (1) corresponds to $\nu_i = \infty$ and in this case $S_i = S_i^{(\infty)}$. In this scenario the nonlinear Gauss-Seidel method (9) for (5) with S_i replaced by $S_i^{(\infty)}$ is the Gauss-Seidel-Newton method applied to the system (5). It is known that the sufficient conditions to guarantee local convergence of the nonlinear Gauss-Seidel method for (5) are also sufficient to guarantee local convergence of the Gauss-Seidel Newton method.

Moreover, it is known that the local convergence rate is independent of the number of inner Newton iterations $\nu_i \geq 1$. See page 327 of the book.¹ Thus, from a local convergence point of view, the application of more than one inner Newton iteration (8) is wasteful. It does not result in a faster outer iteration, but only increases the cost of an outer iteration.

The key to the acceleration of the sequential substitution method applied to (3) as shown before is the interpretation of this method and its inexact form as a solution method for the system (5). This way the inexactness due to truncation of the inner iteration is not viewed as a perturbation of (2), (3), but imbedded into a corresponding nested outer-inner iteration for (5). Hence, this type of inexactness does not lead to a decay in the local convergence rate, but it leads to significant saving in computing time per iteration due to inexactness. In case of the Gauss-Seidel-type method it is also practical and easy to implement, since one only needs to specify the number of Newton iterations (namely $\nu_i = 1$) the black-box simulator has to perform. Knowledge of G_i or the details of the inner Newton solver (8) are not required.

Newton-Type Methods

A Newton type method has much better local convergence properties than Gauss-Seidel methods. Sufficient conditions for local convergence of the Gauss-Seidel methods are more restrictive than those for Newton's method. Moreover, the local convergence rate of the nonlinear Gauss-Seidel method and the Gauss-Seidel-Newton method is r-linear, while Newton's method converges q-quadratically. See.¹⁻³

For the specific case of the inexactness in the inner solver due to terminating the inner Newton solution after one iteration, we can show that the outer Newton is still q-quadratically convergent. The k th iteration of Newton's method for (3) is

$$F'(x^k)s^k = -F(x^k); \quad x^{k+1} = x^k + s^k. \quad (10)$$

For the following discussion, it will be helpful to introduce the notation

$$\tilde{x}_i = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m), \quad (11)$$

$$x_{i,S} = (x_1, \dots, x_{i-1}, S_i(\tilde{x}_i), x_{i+1}, \dots, x_m). \quad (12)$$

Under the assumptions of the Implicit Function Theorem,

$$\frac{\partial}{\partial x_j} S_i(\tilde{x}_i) = \left(\frac{\partial}{\partial x_i} G_i(x_{i,S}) \right)^{-1} \frac{\partial}{\partial x_j} G_i(x_{i,S}). \quad (13)$$

Hence,

$$F'(x) = \begin{pmatrix} \vdots & & & \\ \dots & \left(\frac{\partial}{\partial x_i} G_i(x_{i,S}) \right)^{-1} \frac{\partial}{\partial x_j} G_i(x_{i,S}) & \dots & \\ \vdots & & & \end{pmatrix} \quad (14)$$

Now suppose that the discipline solver S_i is evaluated by applying the Newton method (8) to (1). Hence, in (10) $F(x_k)$ is replaced by $F^{(\nu)}(x_k)$. If we perform only one Newton iteration (8) with starting value $z_i^0 = x_i^k$, then

$$S_i^{(1)}(\tilde{x}_i^k) = x_i^k - \left(\frac{\partial}{\partial x_i} G_i(x^k) \right)^{-1} G_i(x^k).$$

Hence, $F^{(1)}(x^k)$ defined in (7) has the form

$$F^{(1)}(x^k) = \begin{pmatrix} \left(\frac{\partial}{\partial x_1} G_1(x^k) \right)^{-1} G_1(x^k) \\ \vdots \\ \left(\frac{\partial}{\partial x_m} G_m(x^k) \right)^{-1} G_m(x^k) \end{pmatrix}. \quad (15)$$

In (14) we replace $x_{i,S}^k$ by x^k , i.e., we replace $S(\tilde{x}_i^k)$ in the argument (12) by $S^{(0)}(\tilde{x}_i^k) = x_i^k$. If we insert this expression for $F'(x^k)$ and (15) for $F(x^k)$ into the Newton iteration (10), then we obtain the iteration

$$\begin{pmatrix} \vdots & & & \\ \dots & \left(\frac{\partial}{\partial x_i} G_i(x^k) \right)^{-1} \frac{\partial}{\partial x_j} G_i(x^k) & \dots & \\ \vdots & & & \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_m \end{pmatrix} = - \begin{pmatrix} \left(\frac{\partial}{\partial x_1} G_1(x^k) \right)^{-1} G_1(x^k) \\ \vdots \\ \left(\frac{\partial}{\partial x_m} G_m(x^k) \right)^{-1} G_m(x^k) \end{pmatrix}. \quad (16)$$

This is equivalent to the Newton iteration

$$G'(x^k)s = -G(x^k) \quad (17)$$

for (5).

As in the case of the Gauss-Seidel method, interpretation of the Newton iteration (10) with inexactness can be interpreted as an iteration for (5). In this case, the inexactness in the inner iteration is one step of Newton's method for (1) and the resulting outer iteration is Newton's method for (5). Under appropriate conditions both methods, Newton's method for (3) and Newton's method for (5) converge locally q-quadratic. However, from a local convergence point of view, nothing can be gained by evaluating S_i exactly by executing the inner Newton method (8) to very high accuracy. Only one step of the inner Newton method is sufficient to guarantee local q-quadratic convergence. More inner iterations do not result in a faster outer convergence, but only increase the cost per iteration.

In contrast to the Gauss-Seidel method, Newton's method (17) for (5) is more difficult to implement. To evaluate the right hand side one only needs to execute the inner Newton iteration (8) once with starting value $z_i^0 = x_i^k$. This does not require knowledge of the black box that implements (8). However, one also needs the

partial derivatives (13) evaluated at $x = x^k$. If these are approximated by finite differences, then great care must be taken in the proper choice of the finite difference step size. This must be adjusted to the error in the function evaluation.^{2,3} In our context, it must be adjusted to the residual size $\|G_i(x^k)\|$ in the inner Newton iteration (8). See also the discussion in the numerical examples sections.

Broyden–Type Methods

If the derivatives in (14) or on the left hand side of (16) are too difficult or expensive to compute, either directly or by finite differences, then Broyden’s method is an attractive alternative to Newton’s method. Here the Jacobian $F'(x^k)$ is replaced by the Broyden matrix B_k , which is updated using in each iteration using a rank-one update. See.^{2,3} The k th iteration of Broyden’s method for (3) is

$$\begin{aligned} B_k s^k &= -F(x^k), \\ x^{k+1} &= x^k + s^k, \\ B_{k+1} &= B_k - \frac{(F(x^{k+1}) - F(x^k) - B_k s^k)(s^k)^T}{(s^k)^T s^k}. \end{aligned} \quad (18)$$

In addition to a starting value for x , we also have to provide a starting matrix B_0 . Since the Jacobian $F'(x)$ is of the form $I - S'(x)$ we will use $B_0 = I$. This choice meets the requirements of the convergence theory, if $S'(x^*)$ is small relative to I .

As before, we consider the case in which the simulations S_i are not performed exactly, but by a Newton iteration (8). Suppose that only one inner Newton iteration (8) is performed. Thus, $F(x)$ in (18) is replaced by

$$F^{(1)}(x) = (\text{diag}(G'(x)))^{-1} G(x), \quad (19)$$

where

$$\text{diag}(G'(x)) = \begin{pmatrix} \frac{\partial}{\partial x_1} G_1(x) & & \\ & \ddots & \\ & & \frac{\partial}{\partial x_m} G_m(x) \end{pmatrix},$$

cf. (7). We see that Broyden’s method (18) with this inexact S_i evaluation is equivalent to Broyden’s method applied to the nonlinear system

$$F^{(1)}(x) = 0.$$

If the partial derivatives of $\frac{\partial}{\partial x_i} G_i(x)$ exist, then $F^{(1)}(x)$ is differentiable. At a solution x_* of (3),

$$\frac{d}{dx} F^{(1)}(x_*) = \text{diag}(G'(x_*))^{-1} \frac{d}{dx} F^{(1)}(x_*),$$

where $\text{diag}(G'(x_*))$ denotes the block diagonal matrix in (19). Under the additional differentiability assumption of G_i , Broyden’s method for $F^{(1)}(x) = 0$ converges locally q–superlinear, i.e., with the same rate

as Broyden’s method for (3). As in the previous case, nothing can be gained from performing more than one inner iteration. The local convergence rate of the outer Broyden method will not be effected, but each Broyden iteration will be more expensive if more than one inner newton iteration per discipline is performed.

Numerical Examples

Buoyancy–Driven Cavity Flow

We now demonstrate the benefit of the inexact inner iteration by means of an example. The buoyancy-driven flow in a rectangular cavity couples fluid flow equations, given in the stream function vorticity formulation, with an equation for the temperature. It is described by the following partial differential equations.

$$\begin{aligned} \Delta^2 \psi - Ra \frac{\partial \theta}{\partial x} \\ - \frac{1}{Pr} \frac{\partial \psi}{\partial z} \frac{\partial}{\partial x} \Delta \psi + \frac{1}{Pr} \frac{\partial \psi}{\partial x} \frac{\partial}{\partial z} \Delta \psi &= 0, \end{aligned} \quad (20)$$

$$-\Delta \theta + \frac{\partial \psi}{\partial z} \frac{\partial \theta}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \theta}{\partial z} = 0 \quad (21)$$

in $\Omega = (0, 1) \times (0, 1)$ with boundary conditions

$$\begin{aligned} \psi = 0, \quad \frac{\partial \psi}{\partial n} \psi = 0 \text{ on } \partial \Omega, \\ \theta(0, z) = u(z), \quad \theta(1, z) = 0, \\ \frac{\partial \theta}{\partial n} \theta(0, z) = \frac{\partial \theta}{\partial n} \theta(1, z) = 0, \quad 0 \leq z \leq 1, \end{aligned} \quad (22)$$

where θ is the temperature, ψ is the stream function, u is a given function, e.g., $u \equiv 1$, Ra is the Rayleigh number, and Pr is the Prandtl number.

We view the flow equation (20) with corresponding boundary conditions as subsystem 1 and the temperature equation (21) with corresponding boundary conditions as subsystem 2. In particular

$$\begin{aligned} G_1(\psi, \theta) &= \Delta^2 \psi - Ra \frac{\partial \theta}{\partial x} - \frac{1}{Pr} \frac{\partial \psi}{\partial z} \frac{\partial}{\partial x} \Delta \psi \\ &\quad + \frac{1}{Pr} \frac{\partial \psi}{\partial x} \frac{\partial}{\partial z} \Delta \psi, \end{aligned} \quad (23)$$

$$G_2(\psi, \theta) = -\Delta \theta + \frac{\partial \psi}{\partial z} \frac{\partial \theta}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \theta}{\partial z}, \quad (24)$$

each equipped with the corresponding boundary conditions (22). If G_1 is solved for the stream function ψ , it defines $S_1(\theta)$ and if G_2 is solved for the temperature θ , it defines $S_2(\psi)$. The system (20)–(22) can be written as

$$\begin{aligned} \psi &= S_1(\theta), \\ \theta &= S_2(\psi). \end{aligned} \quad (25)$$

For a numerical solution, we apply a finite difference discretization following the descriptions in the paper by Schreiber and Keller.⁴

The systems (23) and (24) are solved for ψ and θ respectively by an inexact Newton-GMRES method, computing the directional derivatives by finite differences. See, e.g. §6.2.1 in the book.³ For each system

G_i , $i = 1, 2$, at each iteration of the inexact Newton method, the absolute stop tolerance for the GMRES solver is the minimum of $\eta \|G_i\|^2$ and η . In all tests, we set $\eta = 0.8$. The inner Newton-GMRES routine is terminated when the residual of the system is less than a prescribed tolerance or a maximum number of iterations has been reached. Notice that since (24) is linear in the temperature θ , the Newton-GMRES algorithm applied to (24) is equivalent to the GMRES method with restart.

We use three methods for the outer iteration: Newton-GMRES method, Broyden's method and the Gauss-Seidel method. The Newton-GMRES method uses finite differences to estimate the derivatives, taking care to calculate the stepsize according to the level of inexactness in the inner Newton iteration. The method contains an implementation of the GMRES algorithm specified in⁵ and the Newton algorithm given in.³ The Broyden's method used is the limited memory algorithm given in.³ The Gauss-Seidel method is given in.¹ We consider only computing only one or two iterations of the inner Newton-GMRES solver. Thus, we define the residual of the overall problem to be:

$$F^{(\nu)}(\psi, \theta) = \begin{pmatrix} \psi - S_1^{(\nu_1)}(\theta) \\ \theta - S_2^{(\nu_2)}(\psi) \end{pmatrix} \quad (26)$$

where ν_1 and ν_2 are either 1 or 2. By comparing these results, we see that by using only one iteration of the inner solver, we can maintain the same local convergence results. Thus, we reduce the overall time of the method.

In all test we use $Ra = 10^4$, $Pr = 1$ and $N = 30$, where N denotes the number of subintervals on the x - and the y axis in the discretization. We terminate the overall algorithm when $\|F^{(\nu)}\| < 10^{-6}$. The tests were executed on a Sun UltraSPARC 1.

In figures 1 - 3, the norm of residual, as defined in (26), at each iteration of the outer method is graphed. Two plots appear on each graph: one where one step of the inner solver for G_1 and G_2 is used and one where two steps of the inner solver is used. In figure 1, the Newton-GMRES method is the method for the outer iteration. We see that using two steps requires less outer iterations. However, the local convergence properties of using one step and two steps are the same as indicated by the slope of the graph at the last few iterations. In figure 2, we see the results for using Broyden's method as the outer iteration. Once again, using two steps requires fewer outer iterations, but both have the same local convergence properties. In figure 3, the results for using the Gauss-Seidel method as the outer method are given. Once again, using one step of the inner solver requires more iterations, but has the same local convergence properties as using two steps.

Table 1 shows the number of evaluations of the systems (23) and (24) for each of the methods. When two

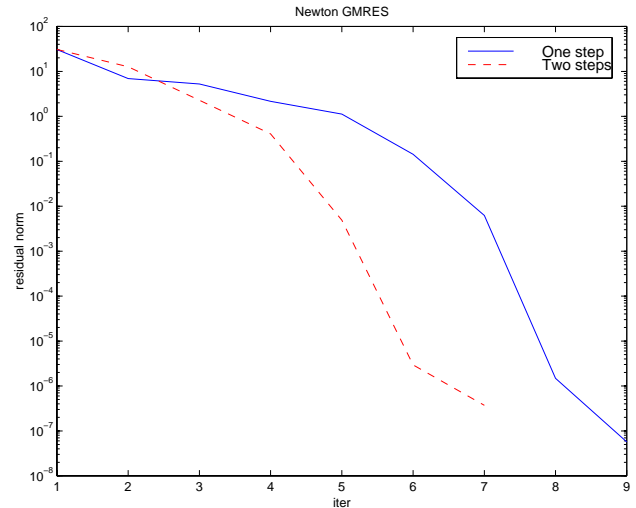


Fig. 1 Residual norms using Newton-GMRES and one or two steps of the inner discipline solver.

steps of the inner solver are used instead of one step, the number of evaluations of each system increases for each method. The number of evaluations of the systems (23) and (24) reflects the overall expense of the solver for the coupled system. The total CPU time in seconds for each outer method, in each case of one and steps of the inner solver are shown in Table 2.

Method	G_1 evals	G_2 evals
Newton-GMRES one step	317	803
Newton-GMRES two steps	666	1248
Broyden one step	218	368
Broyden two steps	353	694
Gauss-Seidel one step	2460	4655
Gauss-Seidel two steps	4549	7372

Table 1 The number of residual evaluations of G_1 and G_2 required by each method.

Method	Time (sec)
Newton-GMRES one step	27
Newton-GMRES two steps	50
Broyden one step	15
Broyden two steps	27
Gauss-Seidel one step	177
Gauss-Seidel two steps	308

Table 2 The total CPU time, in seconds, for each method.

The Viscous-Inviscid Interaction Problem

The second example is a viscous-inviscid interaction (VII) problem in which the Euler equation is coupled with a boundary layer equation to model the flow of air around an airfoil. This example does not exhibit the same traits as the convection example and we cannot apply the theory we have developed. However, by including this example, we can motivate the need for

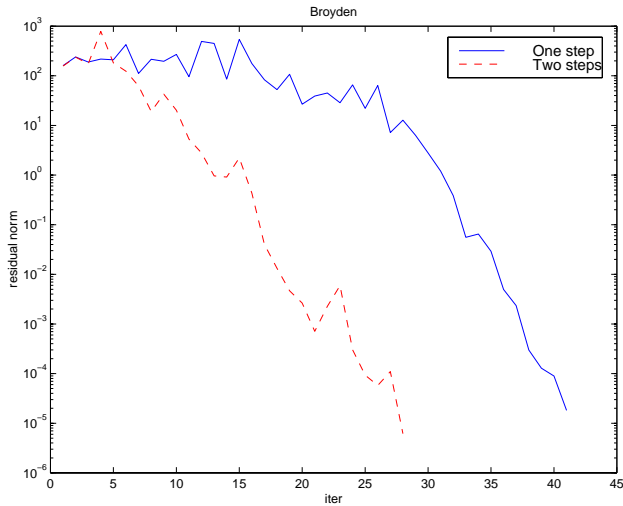


Fig. 2 Residual norms using Broyden and one or two steps of the inner discipline solver.

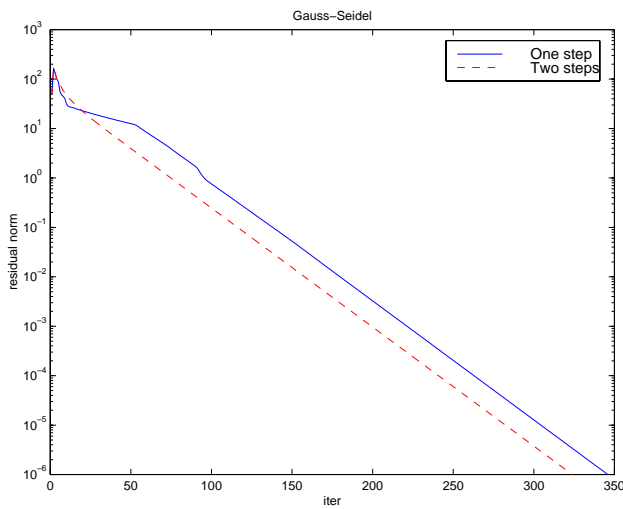


Fig. 3 Residual norms using Gauss-Seidel and one or two steps of the inner discipline solver.

further research.

The VII problems considered here is the same one solved in⁶ using a least squares approach for matching and in^{6,7} using sequential substitution with relaxation. We present here a simplified version of this VII, to keep the description brief. Neglecting the curvature of the airfoil, assuming that its surface corresponds to $y = 0$, we can model the flow by Euler equations in the region at some distance $\delta^*(x)$ away from the airfoil:

$$\partial_x \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ u(\rho e + p) \end{pmatrix} + \partial_y \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ v(\rho e + p) \end{pmatrix} = 0, \quad (27)$$

where u and v denote horizontal and vertical velocity respectively, $p = p(\rho, T)$ denotes pressure, ρ denotes density, T is temperature, and e denotes the specific total energy. In addition to (27), we must specify appropriate boundary conditions at the far field and at $(x, \delta^*(x))$. Near the airfoil, viscosity cannot be ne-

glected and we use boundary layer equations to model the flow. For simplicity, we consider laminar flow, which is modeled by the Prandtl boundary layer equations:

$$\partial_x \begin{pmatrix} \rho u \\ p + \rho u^2 \\ u(\rho e + p) \end{pmatrix} + \partial_y \begin{pmatrix} \rho v \\ \rho uv - \mu \partial_y u \\ v(\rho e + p) - \mu u \partial_y u - k \partial_y T \end{pmatrix} = 0, \quad (28)$$

where μ is the dynamic viscosity, k is the thermal conductivity and $\partial_y p = 0$ and $u = v = 0$ on the surface of the wing ($y = 0$) and $u(x, \infty) = U_e(x)$. Additional boundary conditions may be specified at the leading and trailing edges of the airfoil. From $\partial_y p = 0$ it follows that p (and ρ) is a function of x and, thus, can be computed from the pressure p_e (and ρ_e) at the boundary layer edge. From the solution u of (28) one can compute the displacement thickness δ^* .

The viscous-inviscid-interaction can now roughly be formulated as follows: Given a (an approximation to the) displacement thickness δ^* , one can compute the solution of the Euler equation on a region around the effective airfoil (this is the actual airfoil plus displacement thickness δ^* , i.e., the surface of the effective airfoil is given by $(x, \delta^*(x))$). From the Euler solution one computes the tangential velocity U_e , the pressure p_e , and the density ρ_e at the surface of the effective airfoil. This leads to a function

$$\delta^* \rightarrow (U_e(\delta^*), p_e(\delta^*), \rho_e(\delta^*)). \quad (29)$$

On the other hand, given (approximations of the) U_e , p_e , and ρ_e , one can solve the boundary layer equations. From its solution one extracts the displacement thickness δ^* . This defines a function

$$(U_e, p_e, \rho_e) \rightarrow \delta^*(U_e, p_e, \rho_e), \quad (30)$$

where

$$\delta^*(U_e, p_e, \rho_e) = \int_0^\infty \left(1 - \frac{\rho u}{\rho_e U_e} \right) dy. \quad (31)$$

To couple both equations we need to find $\bar{\delta}^*$ and $\bar{U}_e, \bar{p}_e, \bar{\rho}_e$ such that

$$\begin{aligned} & F((\bar{U}_e, \bar{p}_e, \bar{\rho}_e), \bar{\delta}^*) \\ &= \begin{pmatrix} (\bar{U}_e, \bar{p}_e, \bar{\rho}_e) - (U_e(\bar{\delta}^*), p_e(\bar{\delta}^*), \rho_e(\bar{\delta}^*)) \\ \bar{\delta}^* - \delta^*(\bar{U}_e, \bar{p}_e, \bar{\rho}_e) \end{pmatrix} \\ &= 0. \end{aligned} \quad (32)$$

For the numerical solution, we need to discretize the problem. Thus, the discretized triple (U_e, p_e, ρ_e) corresponds to x_1 in the abstract formulation and the discretized δ^* corresponds to x_2 . The discretized versions of the functions (30) and (29) correspond to S_1 and S_2 , respectively. The functions (29) and (30) are implicitly defined using discretizations of the partial differential equations (27), (28). Abstractly, this can be written

as $G_1((\bar{U}_e, \bar{p}_e, \bar{\rho}_e), \bar{\delta}^*) = 0$ and $G_2((\bar{U}_e, \bar{p}_e, \bar{\rho}_e), \bar{\delta}^*) = 0$. In our numerical experiments, we use existing codes for the subsystem simulations. These are the same as those applied in.^{6,7} The Euler equation solver used to evaluate the function S_1 in (30) is GAUSS2, developed by P. M. Hartwich.^{8,9} The boundary layer equation solver was developed by A. Meade.⁷ Neither of the two solvers uses Newton's method as the inner solver. Thus our discussions in the previous section does not apply. Moreover, details of the discretization (e.g., precise step size selections) applied in the discipline solvers are not known. Therefore, the exact formulations for G_1 and G_2 are not known. Thus, we cannot report the same results as we did for the convection problem. Instead, we can merely state that we can improve upon the basic Gauss-Seidel iteration for this problem.

For the NACA 0012 airfoil, Broyden's method required 511490 seconds and the Gauss-Seidel method required 2881447 seconds to run on a Sun SparcStation Ultra 1.

Extensions of our theory to cover this case, which is representative for many MDA problems, is part of current research.

Conclusions and Outlook

We have studied the numerical solution of systems of blocked nonlinear equations (3) arising in the solution of MDA problems. The evaluation of the residual $F(x)$ involves subsystem simulations, which are performed by iterative methods. The MDA solvers consists of an outer iteration for the solution of the system of blocked nonlinear equations and of inner iterations in the discipline simulators. If the inner iteration is a Newton iteration, then we have shown that the truncation of the inner iterations can be effectively used to accelerate the overall iteration. The key is the interpretation of the outer iteration with inexact inner iteration as an iteration of a related system (5) with same solution as the MDA problem. Further research is needed in the globalization of this inexact iteration and to include different inner iterations, such as pseudo-time marching methods.

Acknowledgements

The authors would like to thank Prof. Andrew Meade, Department of Mechanical Engineering and Material Science, Rice University, for discussions on the VII problem and for making the discipline solvers for the VII problem available to us.

References

¹Ortega, J. M. and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

²Dennis, Jr., J. E. and Schnabel, R. B., *Numerical Methods for Nonlinear Equations and Unconstrained Optimization*, SIAM, Philadelphia, 1996.

³Kelley, C. T., *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

⁴Schreiber, R. and Keller, H. B., "Driven Cavity Flows by Efficient Numerical Techniques," *Journal of Computational Physics*, Vol. 49, 1983, pp. 310–333.

⁵Saad, Y. and Schultz, M. H., "GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comp.*, Vol. 7, 1986, pp. 856–869.

⁶Meade, A. J. and Kokkolaras, M., "Enhancement of a Viscous-Inviscid-Interaction Airfoil Analysis Code Using the Parallel Direct Search Algorithm," *Math. Comput. Modelling*, Vol. 25, 1997, pp. 85–96.

⁷Meade, A. J., "The semi-discrete Galerkin finite element modelling of compressible viscous flow past an airfoil," Tech. rep., Department of Mechanical Engineering, Rice University, 1992.

⁸Hartwich, P. M., "Fresh Look at Shock Fitting," *AIAA Paper 90-0108*, 1990.

⁹Hartwich, P. M., "Split Coefficient Matrix (SCM) Method with Floating Shock Fitting," *Proceedings of the 12th International Conference on Numerical Methods in Fluid Dynamics*, Lecture Notes in Physics, Springer Verlag, New York, 1991.