

MELVIN, R. G.; YOUNG, D. P.; KEYES, D. E.; ASHCRAFT, C. C.; BIETERMAN, M. B.;
HILMES, C. L.; HUFFMAN, W. P.; JOHNSON, F.T.

A Two-level Acceleration Scheme Applied to Full Potential Flow Computations

The TRANAIR aerodynamics code is used as a testbed for developing advanced iterative methods that lend themselves to parallel or distributed computing. In the case of aerodynamics design optimization, a significant part of the computational cost is in the solution of large sparse nonsymmetric linear systems. We have implemented a two-level method for solving such linear systems and present preliminary computational results for problems in two space dimensions. We observe a time-space trade-off mediated by parameters that govern the amount of fill-in in incomplete factorizations on each level. In both subsonic and transonic flow regimes, parameter choices exist that lead to substantial improvements in runtime for comparable memory requirements, relative to a single grid method.

1. Introduction

In this paper, a two-dimensional version of the full potential flow solver TRANAIR [1] is used as a testbed to explore the usefulness of multilevel preconditioners. The computational fluid dynamics (CFD) code is a Newton method based solver, in which the linearized flow equations are solved iteratively using the generalized minimum residual (GMRES) method. A sequence of locally refined solution adaptive grids is employed so that good initial iterates are available on each successively finer grid. An approximate optimization problem is defined on each successive grid. This optimization problem is defined using a regularized model of the flow in the reduced space formed by the design unknowns [2]. The sensitivities of the flow variables to the design parameters are determined by solving a linear system involving the linearized flow problem and right-hand sides that represent the change in transpiration boundary conditions. In many cases, these sensitivity calculations consume the vast majority of the computational resources in the overall optimization. Below, we describe the theory and preliminary computational results of implementing a two-level preconditioner for solving these linear systems.

2. Linear Systems

In the context of a TRANAIR aerodynamic analysis, a series of linear systems is solved when applying Newton's method to the nonlinear full potential equation coupled to an integral boundary layer. In the context of a TRANAIR design and optimization, many such systems must be solved using the same linear operator and differing right-hand sides. As discussed in more detail in [1], the linear systems can be written in the form

$$L \begin{pmatrix} \tilde{T}^{-1}Q \\ \Phi \end{pmatrix} = f, \quad (1)$$

where L represents the linearization of the full potential operator, \tilde{T} is the discrete Prandtl-Glauert operator, Φ is a vector of potentials and doublet parameters and Q is a vector of sources on the global grid. The preconditioned system of equations can be written as:

$$TN^{-1}(f - LT^{-1}\mathcal{X}) = 0 \quad \text{where } \mathcal{X} = \begin{pmatrix} Q \\ \Phi \end{pmatrix}, \quad \text{and } T = \begin{pmatrix} \tilde{T} & 0 \\ 0 & I \end{pmatrix}. \quad (2)$$

Here, N^{-1} is an incomplete factorization operator in which the fill-in is controlled by a drop tolerance. (This formulation ignores cutting off the source distribution in the far field.) The convergence rate of this method depends on the drop tolerance used in the sparse solver. The approximate factorization preconditioner is applied to a reduced set of the solution unknowns, which includes all potentials in an adaptively refined subregion of the global grid and at grid nodes close to configuration surfaces. Equation (2) is actually a formal simplification of the equation for the preconditioned residual evaluation. In fact, the source unknowns Q are further partitioned into sources inside and outside the reduced set, and the preconditioners N^{-1} and T^{-1} are defined differently for the different sets of unknowns. The details of this treatment can be found in [1]. We mention this here merely to explain that the presence of sources in the formulation implies a lower limit greater than one for the number of iterations needed to

solve the linear system (2) even if N^{-1} is computed exactly, that is, with zero drop tolerance.

Suppose we are solving a linear system $Ax = f$ and have two preconditioners B_1^{-1} and B_2^{-1} . One can apply these preconditioners in sequence or “multiplicatively”, first computing $r = f - Ax$, $\delta\hat{x} = B_1^{-1}r$ and $\hat{x} = x + \delta\hat{x}$, and then $\hat{r} = f - A\hat{x}$, $\delta x = B_2^{-1}\hat{r}$ and $x_n = \hat{x} + \delta x$. Now, combining preconditioners gives:

$$\delta x = x_n - x = [B_1^{-1} + B_2^{-1} - B_2^{-1}AB_1^{-1}](f - Ax). \quad (3)$$

3. Two-level Method in TRANAIR

We now introduce a coarse grid into the iterative method in TRANAIR and use the multiplicative method discussed above. In order to make the distinction between the two grids clear we use the subscript c to denote coarse grid and the subscript f for fine grid. So, for example, the coarse grid linear operator is L_c and the fine grid operator is L_f . Let P be the interpolation operator that extends a coarse grid solution onto the fine grid. R is the corresponding restriction operator. We also assume the availability of a preconditioner, N_c^{-1} , on the coarse grid. If we let $B_1^{-1} = PN_c^{-1}R$ and $B_2^{-1} = N_f^{-1}$ in equation (3), we obtain (also applying the T and T^{-1} as in equation (2))

$$T[N_f^{-1} + PN_c^{-1}R - N_f^{-1}LPN_c^{-1}R](LT^{-1}\mathcal{X} - f) = 0 \quad (4)$$

The outer application of TN_f^{-1} is the same as in the standard TRANAIR preconditioning given by equation (2). Equation (4) is a compact symbolic representation of the two-level method. We describe here some of the specific choices taken for the implementation used for the numerical experiments described in the following section.

For this initial implementation, the two-level method is only applied to the design sensitivity calculations on the last grid of the sequence of grids for an inviscid design computation. Like the fine grid, the coarse grid data structure is a non-uniformly refined quadtree. The coarse grid is taken to be the penultimate grid in the sequence of grids. In order to give the method a chance to use a genuinely coarser grid, the adaptive gridding strategy applied to create the last grid ensures an artificially large growth factor in grid size. The prolongation operator P is a matrix form of the existing linear interpolation operator used to generate an initial guess on a grid from the solution on the previous grid. The restriction operator R is a scaled version of P^T , i.e., a full weighting. The coarse grid preconditioner N_c^{-1} is an incomplete factorization of the coarse grid matrix L_c . The approximate factorization may be different from that used for the sensitivities on the previous grid because the coarse grid factorization may be computed with a different choice of drop tolerance. The degree of approximation in the coarse and fine grid preconditioners, N_c^{-1} and N_f^{-1} , is controlled by using coarse and fine grid drop tolerances, τ_c and τ_f , respectively. In order to facilitate code implementation, no local grid derefinement is permitted in the adaptive step.

The major expense per iteration of the current iterative method in TRANAIR (equation (2)) is one application of L and one application of N_f^{-1} . For the iteration given by equation (4), this expense is two applications of L and one each of N_f^{-1} , P , R and N_c^{-1} . We expect the cost of applying P and R to be some small fraction of the cost of applying L .

4. Results

In this section we describe the computational results of applying the two-level method described above to some sample aerodynamic design/optimization test problems in two dimensions.

First we will discuss results for a subsonic flow problem. In this case, the full potential equation is an elliptic partial differential equation. While the equation is nonlinear, it is only mildly so for a low free stream Mach number. As a transonic aerodynamic design test problem, we considered the problem of redesigning an airfoil to ‘match’ (in a least squares sense) a given pressure distribution. The starting geometry is the NACA 0012 airfoil section. Geometry perturbations were parameterized by 10 Tchebyshev polynomial mode shapes. A total of 10 variables were used to parameterize the changes so design sensitivities required the solution of linear systems for 10 different right hand sides. The flow conditions on the freestream Mach number M_∞ and the angle of attack α were $M_\infty = 0.4$ and $\alpha = 1^\circ$. The design objective function was to match the pressure distribution taken from an airfoil derived from the airfoil sections of the ONERA M6 wing analyzed at these flow conditions. Figure 1 shows the target pressure coefficient (C_P) distribution, the solutions on the original and designed airfoils and also the airfoil geometries.

The solution process used a sequence of eight adaptively refined grids where the last two grids (what we call the coarse and fine grids in the two-level method) have 17694 grid boxes and 65511 grid boxes, respectively. Figure 1 shows part of grid 6 (4770 grid boxes) close to the airfoil surface.

This design was repeated with several different values for the drop tolerance(s) in the iterative method. First in the original one-grid method the drop tolerance τ_f was varied. As the drop tolerance was lowered the size

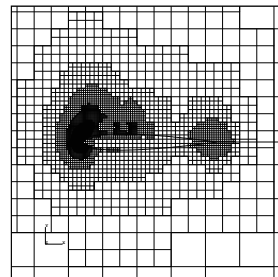


Figure 2: Transonic Flow Test Case

of the decomposed matrix increased and the number of iterations for convergence decreased. For the sensitivity calculations, the relation between the preconditioned relative residual and the unpreconditioned relative residual was monitored and the problem was considered converged when the unpreconditioned residual was reduced by one and a half digits. The effect of inexactness on gradient calculations was investigated in a previous paper [2]. For this particular problem CPU cost for the standard one-grid method was minimized with a small drop tolerance, approximately 10^{-6} , but this comes with a significant storage penalty. At this point it is important to note that for large 3D problems (the eventual target problems) the choice of drop tolerance is limited by the available memory. If the choice of a smaller drop tolerance to improve convergence leads to a decomposition spilling to disk the additional I/O wait time for disk access will almost certainly outweigh the iteration count reduction. Table 1 shows the results of varying the drop tolerance(s) for the one-grid (baseline) method and two-level method. All the times shown in

τ_f	Baseline (1 grid)			Two-level method								
				10^{-3}			10^{-4}			10^{-5}		
	Its. Ave.	CPU (sec)	Elts. (K)	Its. Ave.	CPU (sec)	Elts. (K)	Its. Ave.	CPU (sec)	Elts. (K)	Its. Ave.	CPU (sec)	Elts. (K)
3.1×10^{-1}				31.6	3019	925	12.5	1309	1372	8.9	1003	1614
1.0×10^{-1}				21.8	2452	1186	9.3	1089	1633	6.1	801	1875
1.0×10^{-2}	85.5*	5359	1051	18.4	2180	1910	8.5	1232	2357	5.6	928	2599
1.0×10^{-3}	27.9	2764	2779	15.7	2508	3638	8.3	1576	4085	5.6	1220	4327
1.0×10^{-4}	12.9	1993	4558									
1.0×10^{-5}	6.6	1692	5948									
1.0×10^{-8}	4.9	1957	6923									
0.0	4.9	4297	11632									

* - Not all linear systems fully converged in 100 iterations.

Table 1: Subsonic Flow Test Case: Timings and Memory Use.

Table 1 are on an SGI IRIS 4D35, with a 36MHz R3000 processor and 128MB of memory. The time listed is the total CPU time for the linear system iterations plus the total time for building the decomposition(s). The column 'Its. Ave.' lists the average number of iterations for the ten linear systems. An indication of the memory used is shown in the columns headed 'Elts.' For the one-grid method this is just the number of elements in the decomposition ($|N_f^{-1}|$). For the two-level method this is the sum of the number of elements in the fine grid decomposition, the number of elements in the coarse grid decomposition, and the number of elements in the prolongation operator ($|N_f^{-1}| + |N_c^{-1}| + |P|$). Looking at the results in Table 1, we can see that a significant gain can be made with appropriate choices of drop tolerances used for the two-level method. If we compare the results from the one-grid method with $\tau_f = 10^{-3}$ to those from the two-level method with $\tau_f = 10^{-2}$ and $\tau_c = 10^{-5}$, the storage is slightly reduced for the two-level method while decreasing the total CPU time by about a factor of three. If sufficient storage is available to allow the one-grid method with $\tau_f = 10^{-5}$ then the best comparable two-level results would still be able to gain a factor of two speed-up. For realistic 3D configurations, where memory usage is critical, $\tau_f = 10^{-3}$ may be more representative of the best use of overall computer resources and is in fact the default in the TRANAIR code.

We next consider another pressure-matching aerodynamic design but in which the underlying flow is transonic. The starting geometry was again the NACA 0012 airfoil section. For the transonic flow problem, the flow conditions

on the freestream Mach number M_∞ and the angle of attack α were $M_\infty = 0.75$ and $\alpha = 1^\circ$. Again, the design objective function was to match a target pressure distribution at these flow conditions. In this case both the original airfoil and the target airfoil have significant shock waves in the solution field. Figure 2 shows the target pressure coefficient (C_P) distribution, the solutions on the original and designed airfoils and also the airfoil geometries. The grid sizes are similar to those for the subsonic test problem. Figure 2 also shows part of grid 6 (4779 grid boxes) close to the airfoil surface.

Table 2 shows the results in the same format as for the subsonic test case.

τ_f	Baseline (1 grid)			Two-level method					
				τ_c					
				10^{-4}			10^{-5}		
	Its.	CPU	Elts.	Its.	CPU	Elts.	Its.	CPU	Elts.
	Ave.	(sec)	(K)	Ave.	(sec)	(K)	Ave.	(sec)	(K)
3.1×10^{-3}	96.6*	7841	2188	23.7	3808	3601	18.8	3151	3948
1.0×10^{-3}	47.0	4809	3021	11.8	2153	4434	7.6	1609	4781
3.1×10^{-4}	22.7	2887	3967	10.6	2275	5381	6.9	1745	5727
1.0×10^{-4}	15.2	2473	4961						
1.0×10^{-5}	7.9	2147	6658						
1.0×10^{-8}	5.0	2448	8362						
0.0	5.0	6232	14076						

* - Not all linear systems fully converged in 100 iterations.

Table 2: Transonic Flow Test Case: Timings and Memory Use.

Comparison with the subsonic flow test case results shows this to be a harder problem. The results do not show two-level improvements as large as for the subsonic problem. If, however, we compare the results from the one-grid method with $\tau_f = 10^{-4}$ to those from the two-level method with $\tau_f = 10^{-3}$ and $\tau_c = 10^{-5}$, the storage is slightly reduced for the two-level method while the total CPU time is reduced from 2473 seconds to 1609 seconds.

In the method currently implemented, the coarse grid preconditioner N_c^{-1} is based on a saved linearized coarse grid operator L_c . This operator is saved before the design update on the coarse grid. An improved coarse grid operator could be obtained by not using the saved operator for L_c but constructing an operator by computing the coarse grid Jacobian matrix evaluated about the coarse grid restriction of the current fine grid solution. For the subsonic problem the distinction between these two approaches was not important. We are also hopeful that a more favorable choice of coarse grid will decrease the cost while not sacrificing its efficacy at speeding up convergence. This could be accomplished by allowing the coarse grid to be several levels coarser than the fine grid.

5. Conclusions and Future Directions

Initial testing of a two-level method shows significant reduction in CPU time when compared to the existing one-grid method, in particular for a subsonic flow problem for which a factor of three reduction was observed. We expect that a coarse grid preconditioner based on a linearization about the current fine grid solution will yield a more effective method for transonic flow. In future work we expect to explore other choices for the coarse grid, computing the coarse grid operator about the current fine grid solution, and extending the method to three dimensions.

6. References

- 1 YOUNG, D. P., MELVIN, R. G., BIETERMAN, M. B., JOHNSON, F. T. SAMANT, S. S., BUSSOLETTI, J. E.: A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics; *J. Comp. Phys.*, **92** (1991), 1–66.
- 2 YOUNG, D. P., HUFFMAN, W. P., MELVIN, R. G., BIETERMAN, M. B., HILMES, C. L., JOHNSON, F. T.; Inexactness and Global Convergence in Design Optimization; AIAA Paper 94-4386, 1994.

Addresses: R. G. MELVIN, D. P. YOUNG, C. C. ASHCRAFT, M. B. BIETERMAN, C. L. HILMES, W. P. HUFFMAN, F. T. JOHNSON, The Boeing Company, M/S: 7L-21, PO Box 3707, Seattle WA 98124, USA. D. E. KEYES, Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162 and ICASE, MS 132C, NASA LaRC, Hampton, VA 23681-0001, USA.