Implementation of a Parallel Framework for Aerodynamic Design Optimization on Unstructured Meshes

E.J. Nielsen\textsuperscript{a*}, W.K. Anderson\textsuperscript{a†}, and D.K. Kaushik\textsuperscript{b‡}

\textsuperscript{a}NASA Langley Research Center, MS 128, Hampton, VA 23681-2199

\textsuperscript{b}Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162

1. ABSTRACT

A parallel framework for performing aerodynamic design optimizations on unstructured meshes is described. The approach utilizes a discrete adjoint formulation which has previously been implemented in a sequential environment and is based on the three-dimensional Reynolds-averaged Navier-Stokes equations coupled with a one-equation turbulence model. Here, only the inviscid terms are treated in order to develop a basic foundation for a multiprocessor design methodology. A parallel version of the adjoint solver is developed using a library of MPI-based linear and nonlinear solvers known as PETSc, while a shared-memory approach is taken for the mesh movement and gradient evaluation codes. Parallel efficiencies are demonstrated and the linearization of the residual is shown to remain valid.

2. INTRODUCTION

As computational fluid dynamics codes have steadily evolved into everyday analysis tools, a large focus has recently been placed on integrating them into a design optimization environment. It is hoped that this effort will bring about an ability to rapidly improve existing configurations as well as aid the designer in developing new concepts.

Much of the recent work done in the area of CFD-based design optimization has focused on adjoint methods. This approach has been found to be efficient in aerodynamic problems for cases where the number of design variables is typically large and the number of cost functions and/or flow field constraints is usually small. The adjoint formulation allows rapid computation of sensitivity information using the solution of a linear system of equations, whose size is independent of the number of design variables. Recent examples of this approach can be found in [1-9].

In [8] and [9], a methodology for efficiently computing accurate aerodynamic sensitivity information on unstructured grids is described. A discrete adjoint formulation has been

\textsuperscript{*}Resident Research Associate, National Research Council.
\textsuperscript{†}Senior Research Scientist, Computational Modeling and Simulation Branch, Aerodynamics, Aerothermodynamics, and Acoustics Competency.
\textsuperscript{‡}Graduate Student. Also, Mathematics and Computer Science Division, Argonne National Laboratory.
employed and the exact linearization of the residual has been explicitly demonstrated. Although
the differentiation of the flow solvers has been shown to be highly accurate, a major deficiency
uncovered by the work is the excessive CPU time required to determine adjoint solutions as well
as other associated tasks such as mesh movement. This drawback has hindered computations on
realistically-sized meshes to this point.

In an effort to mitigate this expense, the present work is aimed at the parallelization of the var-
ious steps of the design process. A previously-developed multiprocessor version of the flow
solver is employed, so that the current focus includes modification of the adjoint solver, as well
as appropriate treatment of the mesh movement and gradient evaluation codes. The goal of the
study is to demonstrate acceptable scalability as the number of processors is increased while
achieving results identical to that of the sequential codes. A discussion of the domain decompo-
sition procedure is presented. Speedup figures are established and consistent derivatives are
shown.

3. GOVERNING EQUATIONS

3.1. Flow Equations

The governing flow equations are the compressible Reynolds-averaged Navier-Stokes
equations\(^{10}\) coupled with the one-equation turbulence model of Spalart and Allmaras.\(^{11}\) The
present flow solver implementation is known as FUN3D and is available in both compressible
and incompressible formulations.\(^{12,13}\) The solvers utilize an implicit upwind scheme on
unstructured meshes. The solution is advanced in time using a backward-Euler time-stepping
scheme, where the linear system formed at each time step is solved using a point-iterative algo-
right, with options also available for using preconditioned GMRES.\(^{14}\) The turbulence model is
integrated all the way to the wall without the use of wall functions. This solver has been chosen
for its accuracy and robustness in computing turbulent flows over complex configurations.\(^{9,15}\)
Although originally a sequential code, a parallel version has recently been constructed for invis-
cid flow using MPI and PETSc\(^{16}\) as described in [17]. This implementation utilizes a matrix-
free, preconditioned GMRES algorithm to solve the linear system.

3.2. Adjoint and Gradient Equations

The discrete adjoint equation is a linear system similar in form to that of the flow equations. A
pseudo-time term is added which allows a solution to be obtained in a time-marching fashion
using GMRES, much like that used in solving the flow equations. In the current work, all linear-
izations are performed by hand, and details of the solution procedure can be found in [9]. Once
the solution for the costate variables has been determined, the vector of sensitivity derivatives
can be evaluated as a single matrix-vector product.

4. DOMAIN DECOMPOSITION METHODOLOGY

In the current work, the mesh partitioner MeTiS\(^{18}\) is used to divide the original mesh into sub-
domains suitable for a parallel environment. Given the connectivities associated with each node
in the mesh and the number of partitions desired, MeTiS returns an array that designates a partition
number for each node in the mesh. The user is then responsible for extracting the data struc-
tures required by the specific application.

Due to the gradient terms used in the reconstruction procedure, achieving second-order accu-
racy in the flow solver requires information from the neighbors of each mesh point as well as their neighbors. In the present implementation, the gradients of the dependent variables are computed on each mesh partition, then the results are scattered onto neighboring partitions. This approach dictates that a single level of “ghost” nodes be stored on each processor. These ghost nodes that are connected to mesh points on the current partition are referred to as “level-1” nodes. Similarly, the neighbors of level-1 nodes that do not lie on the current partition are designated “level-2” nodes. This terminology is illustrated graphically in Figure 1.

The adjoint solver requires similar information; however, unlike the flow solver, residual contributions must be written into off-processor memory locations associated with level-2 mesh points. This implies that a second level of ghost information must be retained along partition boundaries. The gather and scatter operations associated with these off-processor computations for the flow and adjoint solvers are handled seamlessly using the PETSc toolkit described in a subsequent section.

Software has been developed to extract the required information from a pre-existing mesh based on the partitioning array provided by MeTiS. This domain decomposition operation is done prior to performing any computations. The user is also able to read in existing subdomains and their corresponding solution files and repartition as necessary. This capability is useful in the event that additional processors become available or processors currently being employed must be surrendered to other users. In addition, software has been developed that reassembles partition information into global files and aids in post-processing the solutions.

5. PARALLELIZATION

Adapting the adjoint solver to the parallel environment has been performed using the MPI message passing standard. In order to expedite code development, the Portable, Extensible Toolkit for Scientific Computation (PETSc)\(^\text{16}\) has been employed, using an approach similar to that taken in [17]. PETSc is a library of MPI-based routines that enables the user to develop parallel tools without an extensive background in the field. The software includes a number of built-in linear and nonlinear solvers as well as a wide range of preconditioning options.

To parallelize the mesh movement and gradient evaluation codes, a shared-memory approach
6. SPEEDUP RESULTS

6.1. Adjoint Solver

For this preliminary work, the speedup obtained by parallelizing the adjoint solver is demonstrated using an SGI Origin 2000 system. Here, an inviscid test case is run on an ONERA M6 wing. The mesh for this test consists of 357,900 nodes. The surface mesh contains 39,588 nodes, and is shown in Figure 2. The freestream Mach number is 0.5 and the angle of attack is 2°. The flow solver is converged to machine accuracy prior to solving the adjoint system. The adjoint solution consists of seven outer iterations, each composed of a GMRES cycle utilizing 50 search directions and no restarts. Figure 3 shows the speedup obtained using an increasing number of processors, and it can be seen that the solver demonstrates a nearly linear speedup for this test case.

6.2. Mesh Movement

As the design progresses, the volume mesh must be adapted to conform to the evolving surface geometry. Currently, this is done using a spring analogy as outlined in [8] and [9]. This procedure is also used to generate mesh sensitivity terms required for evaluating the gradient of the cost function. The implementation of the spring approach requires a number of sweeps through the mesh in order to modify the node coordinates throughout the entire field. Furthermore, in the case of evaluating mesh sensitivities, this process must be repeated for each design variable. For
large meshes, this process can be a costly operation. Therefore, the method has been extended to run across multiple processors using a shared-memory approach as outlined earlier.

Figure 4 shows the speedup obtained by running the mesh movement procedure on the 357,900-node ONERA M6 mesh using a varying number of processors. It can be seen from the figure that the code exhibits a superlinear behavior. This is believed to be due to improved cache efficiency as the size of the subdomains is reduced.

6.3. Gradient Evaluation

Once the flow and adjoint solutions have been computed, the desired vector of design sensitivities can be evaluated as a single matrix-vector product. For shape optimization, this procedure requires the linearization of the residual with respect to the mesh coordinates at every point in the field. Again, for large meshes, this computation is quite expensive. For this reason, a shared-memory approach has been used to evaluate these terms in a parallel fashion.

The previously described ONERA M6 mesh is used to demonstrate the speedup of this implementation, and results obtained for the computation of a single sensitivity derivative are shown in Figure 5. It can be seen that the procedure is generally 90-95% scalable for the case examined. Due to the large amount of memory required for the gradient computation, superlinear speedup such as that shown in Figure 4 is not obtained for this case.

7. CONSISTENCY OF LINEARIZATION

The accuracy of the linearizations used in the sequential adjoint solver has previously been demonstrated in [8] and [9]. In these references, sensitivity derivatives obtained using the adjoint solver were shown to be in excellent agreement with results computed using finite differences. To confirm that these linearizations remain consistent through the port to the parallel environment, sensitivity derivatives are shown in Table 1 for several design variables depicted in Figure 6, where the geometric parameterization scheme has been described in [8] and [19].
Here, the cost function is a linear combination of lift and drag. Results are shown for both the sequential and multiprocessor versions of the codes, using the flow conditions stated in the previous discussion. For the parallel results, eight processors are utilized. It can be seen that the derivatives are in excellent agreement.

8. SUMMARY

A methodology for performing inviscid aerodynamic design optimizations on unstructured meshes has been described. The approach utilizes the PETSc toolkit for the flow and adjoint solvers, in addition to a shared-memory approach for the mesh movement and gradient evaluation codes. Speedup results have been demonstrated for a large test case, and the linearizations have been shown to remain valid.

Table 1
Sensitivity derivatives computed using the sequential and parallel versions of the adjoint solver.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Sequential</th>
<th>Parallel (8 CPU’s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camber #7</td>
<td>-0.241691</td>
<td>-0.241691</td>
</tr>
<tr>
<td>Thickness #5</td>
<td>-0.0204348</td>
<td>-0.0204348</td>
</tr>
<tr>
<td>Twist #2</td>
<td>0.0129824</td>
<td>0.0129824</td>
</tr>
<tr>
<td>Shear #4</td>
<td>0.0223495</td>
<td>0.0223495</td>
</tr>
</tbody>
</table>

Here, the cost function is a linear combination of lift and drag. Results are shown for both the sequential and multiprocessor versions of the codes, using the flow conditions stated in the previous discussion. For the parallel results, eight processors are utilized. It can be seen that the derivatives are in excellent agreement.
9. ACKNOWLEDGMENTS

The authors would like to thank David Keyes for his valuable help and suggestions on the PETSc implementations of the flow and adjoint solvers.

10. REFERENCES