48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2010, Orlando, Florida

AIAA 2010-116

A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations using the SBP-SAT approach

Michal Osusky*, Jason E. Hicken†, and David W. Zingg‡
Institute for Aerospace Studies, University of Toronto, Toronto, Ontario, M3H 5T6, Canada

This paper presents a three-dimensional Newton-Krylov flow solver for the Navier-Stokes equations which uses summation-by-parts (SBP) operators on multi-block structured grids. Simultaneous approximation terms (SAT's) are used to enforce the boundary conditions and the coupling of block interfaces. The discrete equations are solved iteratively with an inexact Newton method. The linear system of each Newton iteration is solved using a Krylov subspace iterative method with an approximate-Schur parallel preconditioner. The algorithm is validated against an established two-dimensional flow solver. Additionally, results are presented for laminar flow around the ONERA M6 wing, as well as low Reynolds number flow around a sphere. Using 384 processors, the solver is capable of obtaining the steady-state solution (reducing the flow residual by 12 orders of magnitude) on a 4.1 million node grid around the ONERA M6 wing in 4.2 minutes. Convergence to 3 significant figures in force coefficients is achieved in 83 seconds. Parallel scaling tests show that the algorithm scales well with the number of processors used. The results show that the SBP-SAT discretization, solved with the parallel Newton-Krylov-Schur algorithm, is an efficient option for three-dimensional Navier-Stokes solutions, with the SAT's providing several advantages in enforcing boundary conditions and block coupling.

I. Introduction

State-of-the-art flow solvers are capable of accurately simulating the flow around three-dimensional aircraft configurations. Some of these are coupled with optimization algorithms which allow for the aerodynamic shape optimization of aircraft geometries for specific objectives, but the size and complexity of the optimization problem in three dimensions presents a challenge from the perspective of computational cost. This paper presents the description of a flow solution algorithm that is well suited to functioning as the core of a three-dimensional aerodynamic shape optimizer.

For computations of high-Reynolds-number turbulent flows over complex geometries, the choice at present is between unstructured and multi-block structured meshes, with various hybrid approaches also in use. The present research is motivated by two advantages associated with structured meshes. They are inherently efficient, where efficiency is a measure of accuracy per unit cost. In particular, structured solvers are more efficient than unstructured solvers when higher-order spatial discretizations are used.[1,2] This paper presents progress toward an efficient parallel three-dimensional multi-block structured solver for turbulent flows over aerodynamic geometries. It extends previous work[2,3] on an efficient parallel Newton-Krylov flow solver for the Euler equations.

An important issue in the development of algorithms for structured multi-block meshes is the treatment of boundaries and block interfaces in a manner that is stable, accurate, and efficient. Multi-block meshes can be implemented using either patched or overlapping blocks. Simultaneous approximation terms (SAT's) can be used when treating block boundary nodes. They were originally developed to treat boundary conditions in an accurate and time-stable manner,[4] and later extended to deal with block interfaces.[5–7] Svärd *et al.*[8,9] and Nordström *et al.*[10] have shown the application of SAT's for the Navier-Stokes equations to

*PhD Candidate, AIAA Student Member
†Post-Doctoral Fellow, AIAA Member
‡Professor and Director, Tier 1 Canada Research Chair in Computational Fluid Dynamics, J. Armand Bombardier Foundation Chair in Aerospace Flight, Associate Fellow AIAA

unsteady problems, as well as some steady model problems. The SAT approach, which works on the basis of penalty application, has several advantages over more traditional approaches. It eliminates the need for mesh continuity across block interfaces, reduces the communication for parallel algorithms, and ensures linear time stability when coupled with summation-by-parts (SBP) operators. However, SAT's have received limited use in computational aerodynamics applications. They present a difficulty in that they can necessitate the use of small time steps with explicit solvers.[11] Hence, the combination of SAT's with a parallel Newton-Krylov solver has the potential to be an efficient approach. The objective of this paper is to demonstrate that such a combination can be an efficient algorithm for the solution of laminar flows over aerodynamic geometries. This is an important step toward the development of an efficient higher-order solver for turbulent flows.

The paper is divided into the following sections. Section II presents an overview of the governing equations, as well as the spatial discretization used, including the SAT's at block interfaces and boundaries. Section III provides details on the Newton-Krylov-Schur method used to solve the large nonlinear system resulting from the discretization of the Navier-Stokes equations. In Section IV, results will be presented comparing the flow solutions produced by the new algorithm against results obtained from an established flow solver, as well as experimental data. Conclusions will be presented in Section V.

II. Governing Equations and Discretization

A. The Navier-Stokes Equations

The work presented in this paper deals with the three-dimensional Navier-Stokes equations, given by:

$$\partial_t \mathbf{Q} + \partial_x \mathbf{E} + \partial_y \mathbf{F} + \partial_z \mathbf{G} = \frac{1}{Re}\Big(\partial_x \mathbf{E_v} + \partial_y \mathbf{F_v} + \partial_z \mathbf{G_v}\Big), \tag{1}$$

where $Re = \frac{\rho_\infty a_\infty l}{\mu_\infty}$,

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{bmatrix},$$

$$\mathbf{E_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \quad \mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix}.$$

Additionally, ρ is density, a is sound speed, e is energy, p is pressure, l is chord length, μ is viscosity, u, v, and w are the velocity components, and τ is the Newtonian stress tensor. The '∞' subscript denotes a free-stream value for the given quantity. The preceding variables have been made dimensionless by the use of the free-stream values of density and sound speed, as well as chord length. Viscosity is calculated using a dimensionless form of Sutherland's law:[12]

$$\mu = \frac{a^3(1 + S^*/T_\infty)}{a^2 + S^*/T_\infty}, \tag{2}$$

where $S^* = 198.6^\circ$R and $T_\infty = 460^\circ$R.

Applying the coordinate transformation $(x, y, z) \rightarrow (\xi, \eta, \zeta)$, which allows us to treat the governing equations on a uniform computational grid, the Navier-Stokes equations can be re-written as

$$\partial_t \hat{\mathbf{Q}} + \partial_\xi \hat{\mathbf{E}} + \partial_\eta \hat{\mathbf{F}} + \partial_\zeta \hat{\mathbf{G}} = \frac{1}{Re}\Big(\partial_\xi \hat{\mathbf{E}}_\mathbf{v} + \partial_\eta \hat{\mathbf{F}}_\mathbf{v} + \partial_\zeta \hat{\mathbf{G}}_\mathbf{v}\Big), \tag{3}$$

where

$$\hat{\mathbf{Q}} = J^{-1}\mathbf{Q},$$
$$\hat{\mathbf{E}} = J^{-1}\Big(\xi_x \mathbf{E} + \xi_y \mathbf{F} + \xi_z \mathbf{G}\Big), \quad \hat{\mathbf{F}} = J^{-1}\Big(\eta_x \mathbf{E} + \eta_y \mathbf{F} + \eta_z \mathbf{G}\Big), \quad \hat{\mathbf{G}} = J^{-1}\Big(\zeta_x \mathbf{E} + \zeta_y \mathbf{F} + \zeta_z \mathbf{G}\Big),$$
$$\hat{\mathbf{E}}_\mathbf{v} = J^{-1}\Big(\xi_x \mathbf{E_v} + \xi_y \mathbf{F_v} + \xi_z \mathbf{G_v}\Big), \quad \hat{\mathbf{F}}_\mathbf{v} = J^{-1}\Big(\eta_x \mathbf{E_v} + \eta_y \mathbf{F_v} + \eta_z \mathbf{G_v}\Big), \quad \hat{\mathbf{G}}_\mathbf{v} = J^{-1}\Big(\zeta_x \mathbf{E_v} + \zeta_y \mathbf{F_v} + \zeta_z \mathbf{G_v}\Big),$$

and J is the metric Jacobian that results from the coordinate transformation. The notation ξ_x, for example, is a shorthand form of $\partial_x \xi$. The viscous stresses also undergo the coordinate transformation and result in the following expressions:

$$
\begin{aligned}
\tau_{xx} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right) \\
\tau_{xy} &= (\mu+\mu_t)\left(\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta + \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta\right) \\
\tau_{xz} &= (\mu+\mu_t)\left(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta + \xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta\right) \\
\tau_{yx} &= \tau_{xy} \\
\tau_{yy} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right) \\
\tau_{yz} &= (\mu+\mu_t)\left(\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta + \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta\right) \\
\tau_{zx} &= \tau_{xz} \\
\tau_{zy} &= \tau_{yz} \\
\tau_{zz} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right) \\
E_{v,5} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_x \partial_\xi(a^2) + \eta_x \partial_\eta(a^2) + \zeta_x \partial_\zeta(a^2)] \\
F_{v,5} &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_y \partial_\xi(a^2) + \eta_y \partial_\eta(a^2) + \zeta_y \partial_\zeta(a^2)] \\
G_{v,5} &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_z \partial_\xi(a^2) + \eta_z \partial_\eta(a^2) + \zeta_z \partial_\zeta(a^2)]
\end{aligned}
$$

where μ_t is the kinetic eddy viscosity, Pr and Pr_t are the laminar and turbulent Prandtl numbers, taken as 0.72 and 0.90, respectively, and $\gamma = 1.4$ for air.

B. Spatial Discretization

The spatial discretization is obtained from the Navier-Stokes equations (3) by the use of SBP operators, while inter-block coupling and boundary conditions are enforced by the use of SAT's. The following will present a brief summary of the two concepts as they apply to the current algorithm. For more details, including the theory behind their development, please refer to references 8–10, 13, 14.

1. Summation-by-Parts Operators

A globally second-order accurate operator for a first derivative is given by

$$\mathcal{D}_1 = \mathcal{P}^{-1}\mathcal{Q}, \tag{4}$$

where

$$
\mathcal{P} = h\begin{bmatrix} \frac{1}{2} & 0 & & & & \\ 0 & 1 & & & & \\ & & \ddots & & & \\ & & & & 1 & 0 \\ & & & & 0 & \frac{1}{2} \end{bmatrix}, \quad
\mathcal{Q} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & \ddots & & & \\ & & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & & -\frac{1}{2} & \frac{1}{2} \end{bmatrix},
$$

and h takes on the value of the spatial difference in the pertinent coordinate direction, either $\Delta\xi$, $\Delta\eta$, or $\Delta\zeta$. In the context of the uniform computational grid, h has a value of 1 for all three coordinate directions.

Numerical dissipation is added to the system using the scalar dissipation model developed by Jameson *et al.*[15] and later refined by Pulliam.[16] The model consists of second- and fourth-difference dissipation operators, whose magnitudes are controlled by the κ_2 and κ_4 coefficients, respectively. For the results presented in this paper, κ_2 is set to 0, while κ_4 typically has a value of 0.02.

When one turns to the discretization of the viscous terms, two types of second derivatives are present, which will be referred to as cross- and double-derivatives, respectively. The cross-derivatives, which have the form

$$\partial_\xi\left(b\partial_\eta u\right), \tag{5}$$

where b is a variable coefficient, can be discretized by the use of (4), resulting in an operator of the form $\mathcal{D}_\xi b \mathcal{D}_\eta u$. As an example, this operator produces the following discretization at an interior point:

$$\partial_\xi\left(b\partial_\eta u\right)_{j,k,m} = \frac{1}{2}b_{j+1,k,m}\left(\frac{u_{j+1,k+1,m} - u_{j+1,k-1,m}}{2}\right) - \frac{1}{2}b_{j-1,k,m}\left(\frac{u_{j-1,k+1,m} - u_{j-1,k-1,m}}{2}\right), \tag{6}$$

where the subscripts (j,k,m) denote indices in the ξ, η, and ζ directions, respectively. At block boundaries, a first-order one-sided difference operator would result for either, or both, of the derivatives in (5).

The double-derivatives possess the following form:

$$\partial_\xi \left(b \partial_\xi u \right). \tag{7}$$

An SBP operator for such a term, denoted $\mathcal{D}_2$, was presented by Mattsson *et al.*:[17]

$$\mathcal{D}_2 = \mathcal{P}^{-1} \left(-\mathcal{D}^T \tilde{\mathcal{B}} \mathcal{D} + \mathcal{B}\mathcal{S} \right), \tag{8}$$

where

$$\mathcal{D} = \frac{1}{h} \begin{bmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & & & \\ & & & -1 & 1 & \\ & & & -1 & 1 \end{bmatrix}, \quad \tilde{\mathcal{B}} = \frac{h}{2} \begin{bmatrix} b_0 + b_1 & & & & & \\ & b_1 + b_2 & & & & \\ & & \ddots & & & \\ & & & b_{N-1} + b_N & & \\ & & & & 0 \end{bmatrix},$$

$$\mathcal{B} = \begin{bmatrix} -b_0 & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & b_N \end{bmatrix}, \quad \mathcal{S} = \frac{1}{h} \begin{bmatrix} -\frac{3}{2} & 2 & -\frac{1}{2} & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & \frac{1}{2} & -2 & \frac{3}{2} \end{bmatrix},$$

and N is the number of nodes in a given coordinate direction. For an internal node, this will result in the narrow stencil used by Pulliam[16] (with k and m subscripts suppressed):

$$\partial_\xi (b \partial_\xi u)_j = \frac{1}{2}(b_{j+1} + b_j)(u_{j+1} - u_j) - \frac{1}{2}(b_j + b_{j-1})(u_j - u_{j-1}). \tag{9}$$

At the high- and low-side boundaries, where one-sided differences are employed, the discretization takes on the form

$$\begin{aligned} \partial_\xi (b \partial_\xi u)_{j=j_{min}} = &\quad -b_j \left[2(u_{j+1} - u_j) - (u_{j+2} - u_{j+1}) \right] + b_{j+1}(u_{j+1} - u_j), \\ \partial_\xi (b \partial_\xi u)_{j=j_{max}} = &\quad b_j \left[2(u_j - u_{j-1}) - (u_{j-1} - u_{j-2}) \right] - b_{j-1}(u_j - u_{j-1}). \end{aligned} \tag{10}$$

2. Simultaneous Approximation Terms

Closely tied to the use of SBP operators for spatial discretization is the use of Simultaneous Approximation Terms (SAT's). SAT's are penalty terms that are added to the equations on block boundaries only, be they domain boundaries or interfaces to other blocks within the domain. Depending on their position, their function is either to enforce the necessary boundary conditions, or to preserve the solution coupling between blocks.

The purpose of this section is not to derive the forms of the various SAT's used, but rather to present the implementation used in the present algorithm. See references 3, 18 for an analysis and derivation of the SAT terms applied in the case of the Euler equations. A brief summary will be presented here. All SAT terms that follow are shown in the form in which they would be added to the right-hand side of (3). The precise form of the inviscid, or Euler, portion of the SAT's on the low side of a block is:

$$\frac{-1}{J} A_\xi^+ \left(\mathbf{Q} - \mathbf{Q}_{\text{external}} \right), \tag{11}$$

where

$$A_\xi^+ = \tfrac{A_\xi + |A_\xi|}{2}, \quad A_\xi = \tfrac{\partial \hat{\mathbf{E}}}{\partial \hat{\mathbf{Q}}},$$

and $\mathbf{Q}$ are the flow variables on the boundary node in the current block. $|A_\xi|$ denotes $X^{-1} |\Lambda| X$, where X is the right eigenmatrix of A_ξ, and Λ contains the eigenvalues along its diagonal. At a high-side boundary

A_ξ^- is used to capture the incoming characteristics and the sign of the penalty is reversed. When dealing with boundaries normal to the other two coordinate directions, A_ξ is replaced by either A_η or A_ζ.

$\mathbf{Q}_{\text{external}}$ takes on the "target" values to which the local values of $\mathbf{Q}$ are being forced. When dealing with a block interface, these are the flow variable values on a coincident node in a neighbouring block, or, when dealing with a far-field boundary, they can be the free-stream flow variable values. A number of different boundary conditions, such as a slip-wall or symmetry plane, can be enforced using this approach for the Euler equations. In each case, the SAT works on a principle very similar to characteristic boundary conditions.

The solution of the Navier-Stokes equations requires viscous SAT terms to be added in addition to the inviscid SAT terms presented in (11). The basis of the viscous SAT's is presented by Nordström *et al.*[10] and is summarized below, with special attention being paid to each type of block boundary.

The first type of viscous SAT is the far-field boundary SAT. In the ξ-direction, this term has the form

$$\frac{\sigma^V}{Re}\left(\hat{\mathbf{E}}_\mathbf{v} - \mathbf{g}_\mathbf{v}\right), \tag{12}$$

where $\hat{\mathbf{E}}_\mathbf{v}$ is the local viscous flux, and $\mathbf{g}_\mathbf{v}$ is the "target" value of the viscous flux. Since this boundary is supposed to force the solution towards free-stream conditions, $\mathbf{g}_\mathbf{v} = 0$. Additionally, $\sigma^V = 1$ at a low-side boundary, and -1 at a high-side boundary.

The no-slip wall boundary condition is enforced with the use of a different type of term, which is again added on top of the Euler SAT. The form of the viscous portion of the no-slip wall SAT, again for a boundary at the low or high side of a block in the ξ-direction, is

$$\frac{\sigma^W}{Re} I\left(\mathbf{Q} - \mathbf{Q}_\mathbf{w}\right), \tag{13}$$

where

$$\sigma^W \leq -\frac{\xi_x^2 + \xi_y^2 + \xi_z^2}{J}\frac{\mu}{2\rho}\max\left(\frac{\gamma}{\mathbf{Pr}}, \frac{5}{3}\right), \quad \mathbf{Q}_\mathbf{w} = \left[\rho_2, 0, 0, 0, e_2\right]^T.$$

σ^W is calculated based on local values, while $\mathbf{Q}_\mathbf{w}$ is constructed in order to enforce an adiabatic no-slip wall boundary condition. The three momentum components are forced toward zero, thus satisfying the no-slip condition, while ρ_2 and e_2 are calculated in such a way as to force zero temperature and pressure gradients normal to the wall. In the present scheme, ρ_2 is taken from one node above the solid surface, while e_2 is calculated based on the pressure of said node with zero velocity. Along with zero velocity, this enforces a zero density and pressure gradient, which in turn enforces the zero temperature gradient condition.

Block interfaces are treated in a similar way, but with unique viscous SAT terms. The form used is:

$$-\frac{\sigma^{iV}}{JRe}\left(\xi_x^2 B_{11} + \xi_y^2 B_{22} + \xi_z^2 B_{33}\right)\left(\mathbf{Q} - \mathbf{Q_2}\right), \tag{14}$$

where

$$\sigma^{iV} \leq 0.5$$

for stability and $\mathbf{Q_2}$ are the values of the flow variables on the coincident node in the adjoining block. The metric terms change to η or ζ depending on the coordinate direction normal to the interface. The B-matrices are related to the viscous Jacobian, and are derived based on Nordström *et al.*[10] Refer to the Appendix for their complete forms. Interface SAT's also make use of (12), where $\mathbf{g}_\mathbf{v}$ is replaced by $\hat{\mathbf{E}}_\mathbf{v2}$, the viscous flux on the coincident node in the adjoining block.

In order to reduce the size of the computational domain, symmetry boundaries can be imposed. SAT's are again used to impose this boundary condition by using (11) to impose a purely tangential flow ($\mathbf{Q}_{\text{external}}$ is constructed in such a way as to force the normal velocity component to zero). In addition, (13) is used to enforce a zero normal gradient in all conservative variables.

The following is used to enforce the inviscid SAT on an outflow boundary in a viscous flow:

$$\frac{\sigma^I}{J} A_\xi^-\left(\mathbf{Q}_{j_{max}} - \mathbf{Q}_{j_{max}-1}\right), \tag{15}$$

in which the boundary is assumed to be on the high side of a block in the ξ-direction. The modification is appropriate in dealing with the viscous wake region. The advantage of this approach is that it requires minimal modification to the existing Euler SAT term, which uses the free-stream flow conditions instead of $\mathbf{Q}_{j_{max}-1}$. An alternate approach to dealing with the outflow condition is presented by Svärd *et al.*[8]

III. Solution Methodology

Applying the SBP-SAT discretization described in the previous section to the steady Navier-Stokes equations results in a large system of nonlinear equations:

$$\mathcal{R}(\mathcal{Q}) = 0, \tag{16}$$

where $\mathcal{Q}$ represents the complete solution vector. When time-marched to steady state with the implicit Euler time-marching method, this system results in a large set of linear equations of the form:[19]

$$\left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)}\right)\Delta\mathcal{Q}^{(n)} = -\mathcal{R}^{(n)}, \tag{17}$$

where n is the outer iteration, Δt is the time step, $\mathcal{I}$ is an identity matrix, $\mathcal{R}^{(n)} = \mathcal{R}(\mathcal{Q}^{(n)})$, $\Delta\mathcal{Q}^{(n)} = \mathcal{Q}^{(n+1)} - \mathcal{Q}^{(n)}$, and

$$\mathcal{A}^{(n)} = \frac{\partial\mathcal{R}^{(n)}}{\partial\mathcal{Q}^{(n)}}$$

is the flow Jacobian.

In the infinite time step limit, the above describes Newton's method and will converge quadratically if a suitable initial iterate, $\mathcal{Q}^{(0)}$, is known. This must be sufficiently close to the solution of (16). Since it is unlikely that any initial guess made for a steady-state solution will satisfy this requirement, the present algorithm makes use of a start-up phase whose purpose it is to find a suitable initial iterate. The following sections describe each of the phases as they apply to the solution of the Navier-Stokes equations. It should be noted that both phases result in a large set of linear equations at each iteration, which are solved to a specified tolerance using a Krylov iterative solver such as GMRES.

A. Approximate-Newton Phase

The approximate-Newton method uses implicit Euler time-stepping in order to find a suitable initial iterate for Newton's method. Since we are not interested in a time-accurate solution, some useful modifications can be made. These include a first-order Jacobian matrix, a spatially varying time step, and the use of a lagged Jacobian update.

A first-order Jacobian matrix, $\mathcal{A}_1$, has been shown to be an effective replacement of the true Jacobian, $\mathcal{A}$, during the start-up phase.[20,21] Two approximations are made when creating the first-order approximation to the Jacobian. When dealing with the inviscid terms, the fourth-difference dissipation coefficient, κ_4, is combined with the second-difference dissipation coefficient, κ_2, to form a modified second-difference dissipation coefficient, $\tilde{\kappa}_2$, such that

$$\tilde{\kappa}_2 = \kappa_2 + \sigma\kappa_4,$$

where σ is a lumping factor. A value of $\sigma = 6$ has been shown to work well for the Navier-Stokes solutions with scalar dissipation.[22] The modified fourth-difference dissipation coefficient, $\tilde{\kappa}_4$, is set to zero. Applying this lumping approach reduces the accuracy of the Jacobian to first-order. However, this does not impact the order of accuracy of the final solution. Moreover, it also reduces the number of matrix entries of the inviscid terms.

The viscous terms, however, still possess a relatively large stencil. To mitigate this, the cross-derivative terms of (5) are dropped when constructing the first-order Jacobian, and only the double derivative terms (7) are linearized. This approach reduces the stencil of all interior nodes to nearest neighbours only, which is substantially smaller than that of the full Jacobian.

Most of the SAT's do not require modification when used as part of $\mathcal{A}_1$, as they only use information from one local node. The exception to this is the viscous flux SAT component found in the far-field and interface terms. What is done here, analogous to dropping the cross-derivative terms in the internal scheme, is that the tangential components of the viscous stresses are not linearized.

The use of the implicit Euler method necessitates the specification of a time step, whose inverse is added to the diagonal elements of $\mathcal{A}_1$. A spatially varying time step has been shown to improve the convergence rates of Newton-Krylov algorithms, leading to the use of the following value:

$$\Delta t^{(n)}_{j,k,m} = \frac{J_{j,k,m}\Delta t^{(n)}_{\text{ref}}}{1 + \sqrt{J_{j,k,m}}}, \tag{18}$$

where (*j,k,m*) denotes the node to which this time step is being applied. Since the solver uses the unscaled flow variables $\mathbf{Q}$, instead of the transformed variables $\hat{\mathbf{Q}}$, the J term that results from the coordinate transformation is lumped into the numerator of (18). The reference time step is

$$\Delta t_{\text{ref}}^{(n)} = a(b)^n,$$

where typical values used are $a = 0.1$ and $b \in [1.1, 1.6]$.

The first-order Jacobian is factored using block incomplete lower-upper factorization (BILU) with fill level p in order to construct the preconditioner used throughout the solution process. This is a computationally expensive task, especially in the approximate-Newton phase, which requires many outer iterations. Previous work[3, 23] has shown that lagging the update of the preconditioner (freezing it for a number of iterations) during the start-up phase can positively impact the efficiency of the flow solver.

An approximate-Schur[24] parallel preconditioner is used in the current algorithm. An additive-Schwarz[25] approach will be evaluated in the future, which will help determine which one is better suited to the class of problems being tackled.

An important part of using a start-up phase is knowing when a suitable iterate has been found to initiate the inexact-Newton phase. For this purpose, the relative drop in the residual is used:

$$R_d^{(n)} \equiv \frac{||\mathcal{R}^{(n)}||_2}{||\mathcal{R}^{(0)}||_2}. \tag{19}$$

Once this value reaches 0.01, i.e. the residual has dropped by 2 orders of magnitude in the approximate-Newton phase, the algorithm is switched to the inexact-Newton method.

B. Inexact-Newton Phase

The inexact-Newton phase uses a different scheme for the reference time step. This scheme is designed to ramp the time step toward infinity more rapidly than in the approximate-Newton phase, eliminating the inverse time term from the diagonal of the left-hand-side of the discretized Navier-Stokes equations. The present work involves the use of a scheme developed by Mulder and van Leer,[26] by which a new reference time step is calculated and used in (18):

$$\Delta t_{\text{ref}}^{(n)} = \max\left[\alpha\left(R_d^{(n)}\right)^{-\beta}, \Delta t_{\text{ref}}^{(n-1)}\right],$$

where $\beta \in [1.8, 2.0]$ and α is calculated as

$$\alpha = a(b)^{n_{\text{Newt}}}\left(R_d^{(n_{\text{Newt}})}\right)^{\beta},$$

and n_{Newt} is the first iteration of the inexact-Newton phase.

In contrast with the approximate-Newton method, this method uses the exact, second-order accurate Jacobian. However, since we use a Krylov subspace method, we do not need to form the Jacobian matrix, $\mathcal{A}$, explicitly. Instead, only Jacobian-vector products are required, which can be approximated using a first-order forward difference:

$$\mathcal{A}^{(n)}\mathbf{v} \approx \frac{\mathcal{R}(\mathcal{Q}^{(n)} + \epsilon\mathbf{v}) - \mathcal{R}(\mathcal{Q}^{(n)})}{\epsilon}.$$

The parameter ϵ is determined from

$$\epsilon = \sqrt{\frac{N_u\delta}{\mathbf{v}^T\mathbf{v}}},$$

where N_u is the number of unknowns and $\delta = 10^{-13}$. The approximate Jacobian, $\mathcal{A}_1$, is still used for preconditioning the system.

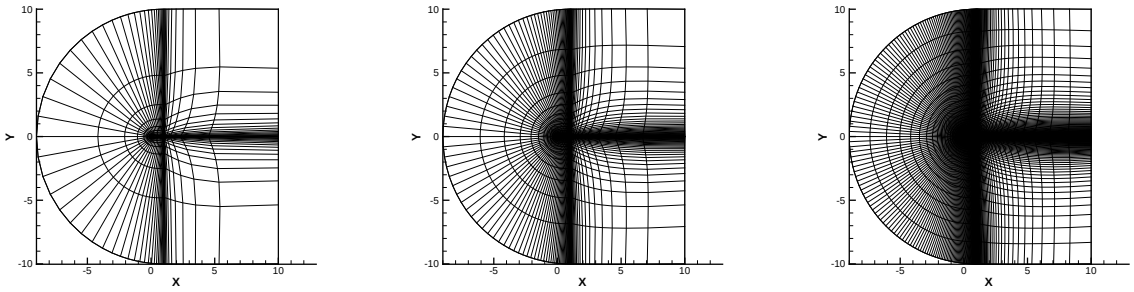
Finally, it should be emphasized that neither the approximate-Newton nor the inexact-Newton phases solve their respective linear systems exactly. Instead, the following inequality is used to govern how far the system is solved:

$$||\mathcal{R}^{(n)} + \mathcal{A}^{(n)}\Delta\mathcal{Q}^{(n)}||_2 \leq \eta_n||\mathcal{R}^{(n)}||_2,$$

where the forcing parameter η_n is specified. If it is too small, the linear system will be over-solved and will take too much time, but if it is too large, non-linear convergence will suffer. For the present work, a value of 0.1 is used for the approximate-Newton phase, while 0.01 is used for the inexact-Newton phase.

Table 1: 2D grid parameters

Grid	Grid Size	Nodes off wall	Nodes in wake	Off-wall spacing
coarse	71x17	17	11	5.4e-4
medium	141x33	33	21	2.3e-4
fine	281x65	65	41	1.0e-4



(a) coarse (b) medium (c) fine

Figure 1: Grids used in convergence study

IV. Results

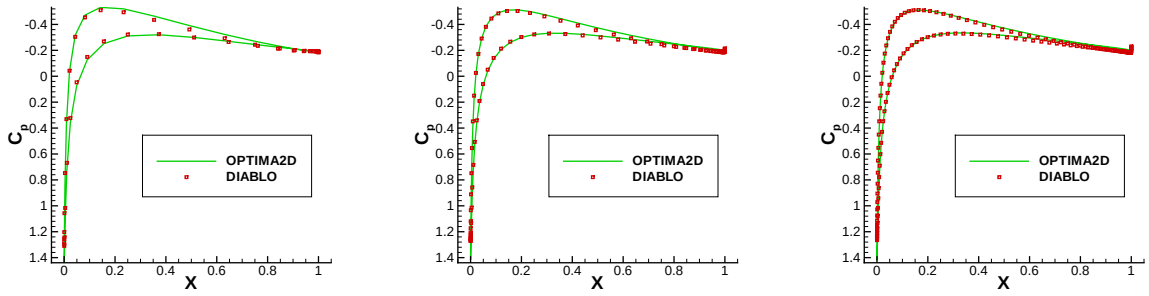
The results presented in this section highlight the the use of the SBP-SAT approach in solving the Navier-Stokes equations for laminar flow over a NACA 0012 airfoil. A series of refined grids was created that is used with the three-dimensional solver developed here, called DIABLO, as well as an extensively verified and validated two-dimensional finite-difference Newton-Krylov algorithm, OPTIMA2D,[27] that does not use SAT's. The spatial discretization in OPTIMA2D is essentially the same as that in the well-known ARC2D.[16] The purpose of this comparison is to highlight the effects of using the SBP-SAT approach and the behaviour of the flow solution on a series of refined grids. In addition, the effect of introducing additional interfaces into the computational domain will be examined. A series of three-dimensional flow solutions around an ONERA M6 wing are also presented, highlighting the use of the solver on an H-H topology multi-block grid. The parallel scaling of the algorithm is examined by computing the solution around an ONERA M6 wing with up to 384 processors. Finally, flow solutions around a sphere at low Reynolds numbers are computed, and the drag values are compared to published experimental results.

A. NACA 0012 Grid Convergence Study

1. Grids

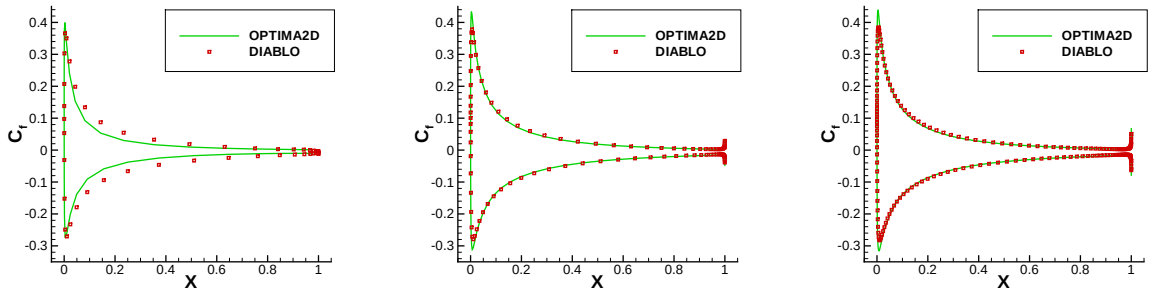
The finest grid used in this study was created using an elliptical grid generation program around the NACA 0012 airfoil geometry. The two coarser grids were created by removing every second grid node in both coordinate directions. The grid topology is a single-block C-mesh, with a single interface present at the wake-cut. Table 1 provides a summary of the grid characteristics for the three meshes used in this study, while Figure 1 shows the grids themselves.

To highlight the use of SAT interfaces in a multi-block setting, a series of grids with additional interfaces at the trailing edge of the airfoil were created, each having 3 blocks and 3 interfaces linking those blocks together. The added interfaces extend vertically up and down from the trailing edge of the airfoil.



(a) coarse mesh (b) medium mesh (c) fine mesh

Figure 2: C_p contours around NACA 0012 airfoil



(a) coarse mesh (b) medium mesh (c) fine mesh

Figure 3: C_f contours around NACA 0012 airfoil (plotted negative on the lower surface for clarity)

2. Flow Solutions

The flow parameters for the grid convergence study are

$$Ma = 0.63, \quad Re = 600, \quad \alpha = 2.0^{\circ},$$

where α is the angle of attack.

Converged steady-state flow solutions were obtained for all grids by reducing the initial flow residual by 12 orders of magnitude. As the contours of the flow solutions themselves are not appreciably different between the two solvers, the focus here is on the coefficients of pressure and skin friction, which determine the lift and drag. The coefficient of pressure, C_p, plots can be seen in Figure 2, while Figure 3 presents the coefficient of skin friction, C_f, plots for all three grid levels. As demonstrated by the C_p plots, the solutions produced by DIABLO match very closely with those produced by OPTIMA2D on all three grid levels. The plots of C_f show that, as grid refinement decreases, the skin friction predicted by OPTIMA2D decreases, while that predicted by DIABLO increases. The agreement is excellent on the finest grid.

SAT's do not enforce boundary and interface conditions exactly; instead, they force the solution at the appropriate node toward the desired value. In the case of the airfoil surface, the solution is forced towards a no-slip wall condition, but this is only achieved in the limit as h goes to 0. The surface velocity is not exactly zero, as it is in OPTIMA2D. Figure 4 presents the largest surface velocity error as it varies with the number of nodes in the ξ-direction for each grid. The slope of this plot provides the convergence rate for the method. Hence, the velocity is converging at a second-order rate, as expected.

The same behavior was observed when the solution was computed on the grids that had the additional interfaces inserted at the trailing edge of the airfoil, so no C_p and C_f plots are presented for those cases. Instead, a qualitative comparison of the density contours at the trailing edge is presented in Figure 5,

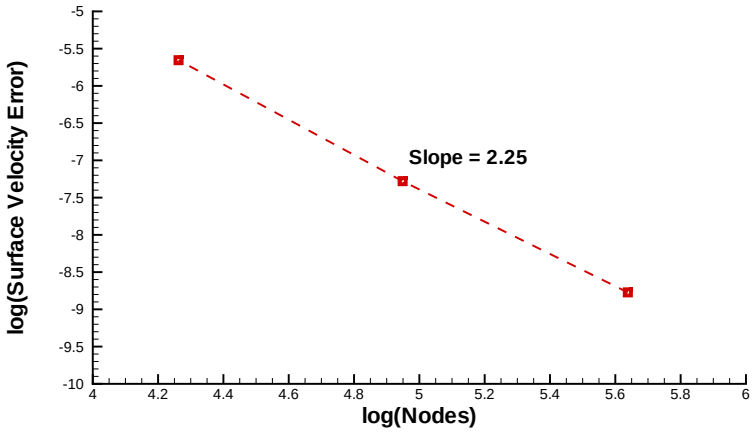
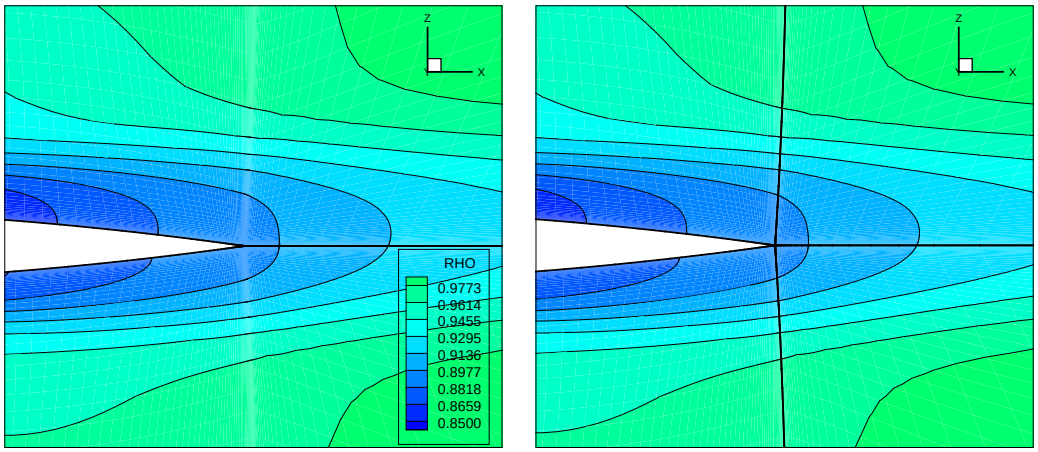


Figure 4: DIABLO maximum surface velocity error vs. node number



(a) Single-block grid

(b) 3-block grid

Figure 5: Density contours at the trailing edge of NACA 0012 airfoil (thicker lines represent block interfaces)

highlighting that the presence of the interfaces has a negligible impact on the converged solution. Both solutions shown were computed on the finest grids.

Finally, Table 2 compares the lift and drag values produced by OPTIMA2D and DIABLO. The values once again demonstrate the effect that the solid-surface SAT's have on the solution. While the finest grids, where the SAT's are most successful in enforcing the no-slip condition, produce close correspondence between OPTIMA2D and DIABLO, this is not the case for the coarser grids. In addition, the introduction of the extra interfaces has little effect on the lift and drag values produced by DIABLO. The values of C_d computed using DIABLO on the coarse mesh are closer to the fine mesh results than those computed with OPTIMA2D.

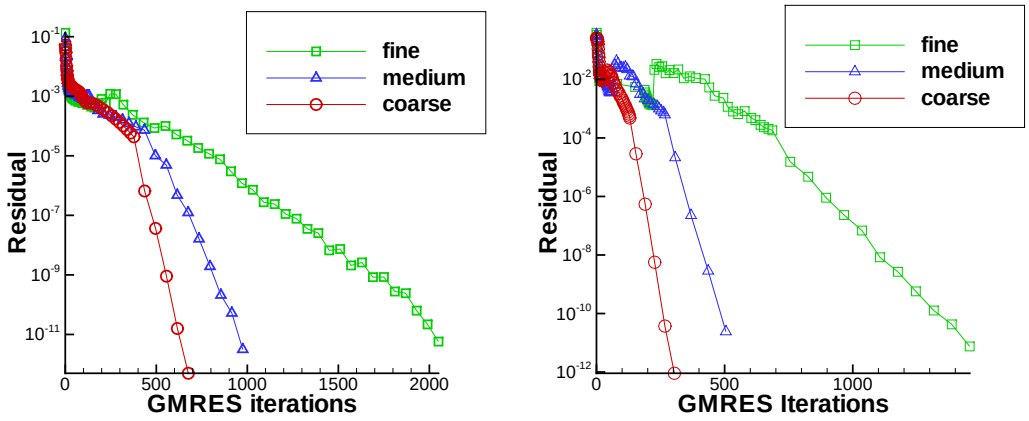
Figure 6a shows the convergence histories for all three grid levels in the 1-block case. The plot shows the number of linear (GMRES) iterations in both solution phases. The symbols along the lines show the outer iterations.

B. ONERA M6 Wing Flow Solutions

To demonstrate the use of DIABLO on a three-dimensional multi-block grid, a grid refinement study was carried out on a series of 48-block grids around an ONERA M6 wing. The grids consist of an H-H topology

Table 2: Lift and Drag coefficients for C-mesh (p and f denote pressure and friction contributions, respectively; t denotes total)

Grid		OPTIMA2D			DIABLO (1 blk)			DIABLO (3 blk)		
		C_l	C_d	C_l/C_d	C_l	C_d	C_l/C_d	C_l	C_d	C_l/C_d
coarse	p		0.0613			0.0556			0.0555	
	f		0.0677			0.0980			0.0979	
	t	**0.1294**	**0.1290**	**1.0031**	**0.1029**	**0.1536**	**0.6699**	**0.1030**	**0.1534**	**0.6714**
medium	p		0.0624			0.0557			0.0557	
	f		0.1050			0.1061			0.1060	
	t	**0.1063**	**0.1675**	**0.6349**	**0.0994**	**0.1617**	**0.6147**	**0.0996**	**0.1617**	**0.6160**
fine	p		0.0626			0.0559			0.0559	
	f		0.1090			0.1095			0.1095	
	t	**0.1029**	**0.1717**	**0.5997**	**0.1042**	**0.1653**	**0.6303**	**0.1043**	**0.1653**	**0.6310**



(a) 1-block NACA 0012 cases

(b) 48-block ONERA M6 cases

Figure 6: DIABLO convergence histories

mesh (see Figure 7). Flow parameters for this case are

$$Ma = 0.63, \quad Re = 600, \quad \alpha = 2.0^{\circ}.$$

All solutions were calculated using 48 processors, with each block of the computational domain assigned to one processor. All blocks have the same size, so the problem is load balanced.

A three-dimensional grid presents a number of challenges that are not present in a two-dimensional C-mesh. Most notably, the wing tip, where several blocks come together, provides a good test of the SAT's ability to produce smooth solutions across block interfaces. Several different types of SAT's are applied at the tip, accounting for the surface of the wing, as well as the interfaces present at this location. Figure 8 presents the Mach number contours at the wing tip (viewed from above). The solution is continuous across the interfaces, and the no-slip condition is enforced (to order h^2) on the solid surface boundaries of all blocks, where the Mach number is reduced to zero.

Figure 6b presents the convergence history of the flow solutions around the ONERA M6 wing on all three grid levels. Table 3 presents a summary of some grid parameters, as well as the final CPU time required to reduce the residual 12 orders of magnitude and the final lift and drag coefficient values obtained on all grid levels. On the finest mesh, which has roughly 4 million nodes, the solver converges the values of C_L and C_D to 3 significant figures in 26 minutes using 48 processors.

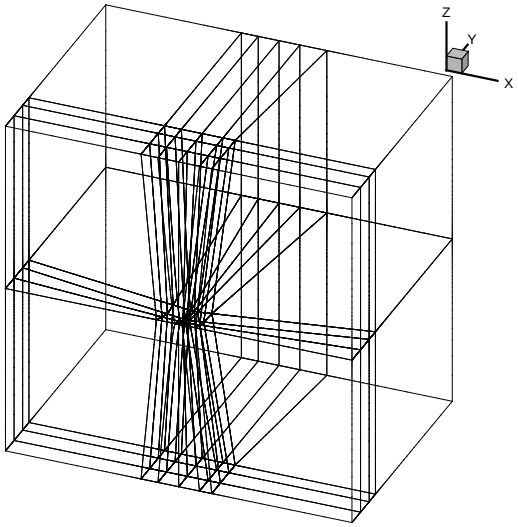


Figure 7: ONERA M6 grid

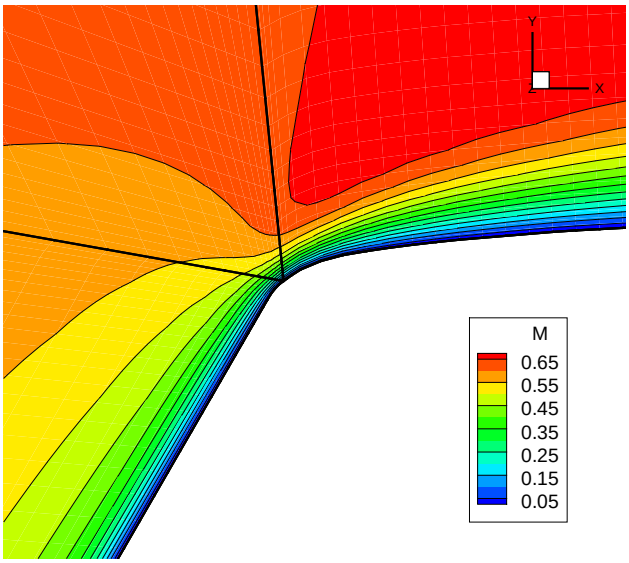
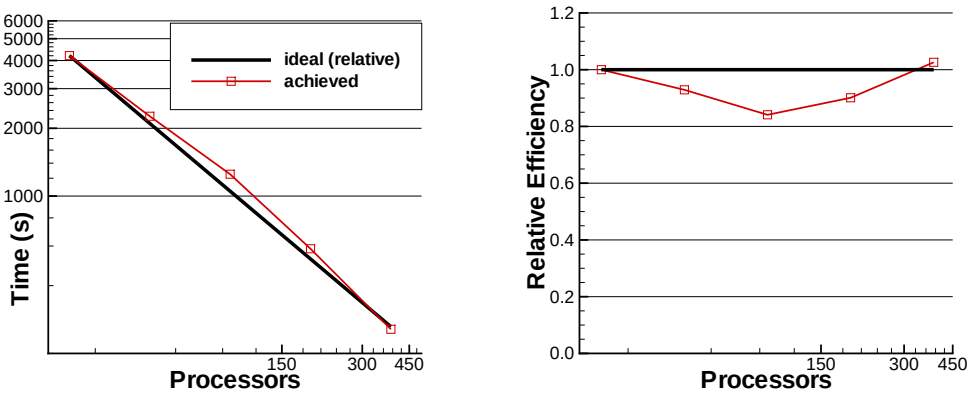


Figure 8: ONERA M6 wing-tip contour

Table 3: Summary of grid characteristics and results for 48-block ONERA M6 wing flow solutions

Grid	Block Dimensions	Grid Size (nodes)	Solution Time (min)	Lift C_L	Drag C_D	Lift/Drag C_L/C_D
coarse	10x14x11	73,920	0.78	0.0984	0.1019	0.9657
medium	19x27x21	517,104	5.87	0.0920	0.1588	0.5793
fine	37x53x41	3,859,248	92.5	0.0878	0.1679	0.5229



(a) CPU time

(b) Relative efficiency

Figure 9: Scaling performance results. The ideal time and efficiency values are based on the 24 processor case.

C. Parallel Scaling

In order to demonstrate the parallel performance of the current algorithm, a 384-block H-H topology grid around an ONERA M6 wing was created, with (19x27x21) nodes per block, and an off-wall spacing of 10^{-4} chord units. The total grid size is approximately 4.1 million nodes. All runs were performed on the general-purpose cluster of SciNet, which uses Nehalem processors interconnected with non-blocking 4x-DDR InfiniBand.

The results presented in Figure 9 show that the algorithm scales well with the number of processors. Comparison to ideal scaling based on the 24 processor case shows that the algorithm's performance does not deteriorate as more processors are used to compute a flow solution. It should be noted that the preconditioner effectively changes as the number of processors increases due to the distribution of blocks among the processors. This results in a slightly different solution algorithm for each number of processors, which explains the super-linear speedup observed in the figure. Using 384 processors, the solver is capable of obtaining the steady-state solution (reducing the flow residual by 12 orders of magnitude) in 4.2 minutes, while convergence to 3 significant figures in force coefficients is achieved in 83 seconds.

D. Flow Around a Sphere at Low Reynolds Numbers

The final set of results demonstrate the calculation of laminar flow around a sphere at low Reynolds numbers. This case permits a direct comparison to experimental drag values and highlights the ability of the solver to predict relatively complex three-dimensional flows which include flow separation. The flow parameters for the cases were

$$Ma = 0.25, \quad Re = 20 \text{ to } 200, \quad \alpha = 0.0^{\text{o}}.$$

A 5-block mesh is used with a 10^{-4} sphere diameter units off-wall spacing, 9,000 surface nodes, and 360,000 total nodes.

The results show excellent agreement with the experimental drag data of Roos and Willmarth,[28] as can be seen in Figure 10. Moreover, Figure 11 demonstrates that the solver correctly captures the flow contours of a separated, recirculating flow at a Reynolds number of 118, as compared to the experimental results of Taneda.[29]

V. Conclusions

The SBP-SAT approach was shown to provide accurate laminar flow solutions to the Navier-Stokes equations when used within a Newton-Krylov-Schur algorithm. Validation against an established two-dimensional

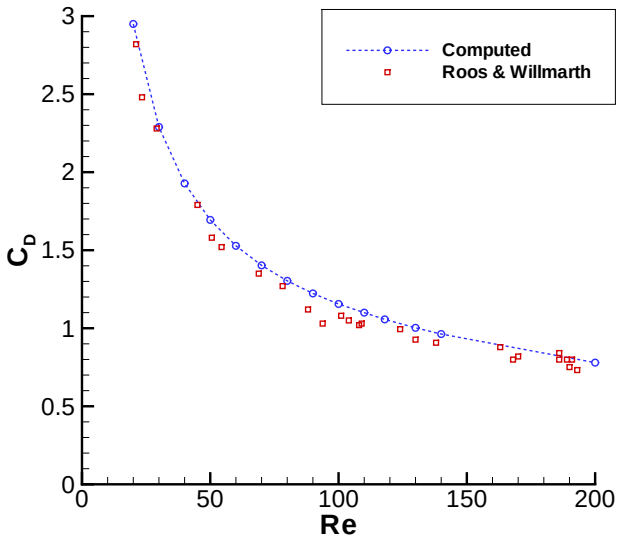
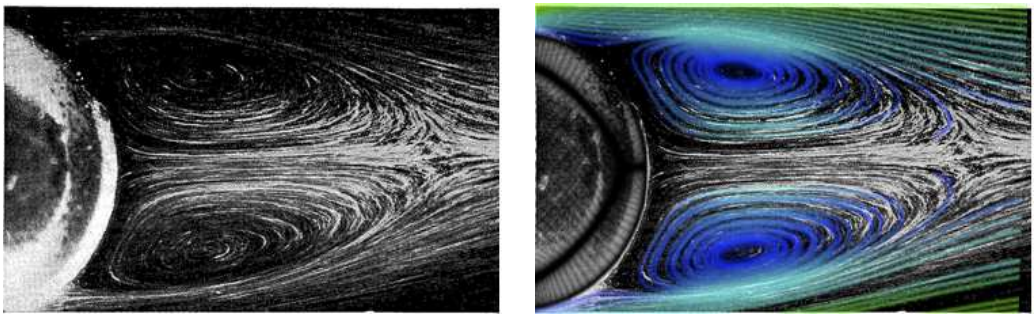


Figure 10: Drag on a sphere at low Reynolds numbers



(a) Experimental result[29]

(b) Superimposed numerical result

Figure 11: Recirculating flow behind sphere at $Re = 118$

solver showed that the accuracy of the SAT-based approach is comparable to the traditional approach. The non-zero surface velocities which result from the SAT-based enforcement of boundary conditions converge at a second-order rate, as expected. The inclusion of interfaces in the grid structure does not introduce appreciable changes to the converged steady-state solution, and the SAT's produce smooth solutions across block interfaces, even in areas where many interfaces meet.

The solution of a three-dimensional flow around an ONERA M6 wing demonstrates that the current algorithm can be readily applied to a mesh consisting of many blocks. The SBP-SAT approach can be applied to a computational domain with an arbitrary number of blocks, which, when solved on a large number of processors, can greatly reduce the time needed to obtain accurate solutions. The algorithm has very good parallel efficiency in the range of processors considered (24 to 384).

Finally, computations of the flow around a sphere for a range of low Reynolds numbers highlight the capabilities of the solver in not only correctly predicting the drag, but also in capturing complex flow features, such as flow separation and recirculation.

Acknowledgments

The authors gratefully acknowledge financial assistance from the Natural Sciences and Engineering Research Council (NSERC), the Canada Research Chairs program, and the University of Toronto.

Appendix

The following are the complete forms of the conservative formulation of the B_{ii} matrices, where $i = 1, 2, 3$.
B_{11}:

$$B_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{4}{3}\frac{\mu}{\rho}u & \frac{4}{3}\frac{\mu}{\rho} & 0 & 0 & 0 \\ -\frac{\mu}{\rho}v & 0 & \frac{\mu}{\rho} & 0 & 0 \\ -\frac{\mu}{\rho}w & 0 & 0 & \frac{\mu}{\rho} & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & \frac{\gamma}{\mathrm{Pr}}\frac{\mu}{\rho} \end{bmatrix}$$

where:

$$\begin{aligned} b_{51} &= \frac{\mu}{\rho}\left[-\frac{\gamma}{\mathrm{Pr}}\frac{p}{(\gamma-1)\rho} - \left(\tfrac{4}{3}u^2 + v^2 + w^2\right) + \tfrac{1}{2}\tfrac{\gamma}{\mathrm{Pr}}\left(u^2+v^2+w^2\right)\right] \\ b_{52} &= \left(\tfrac{4}{3} - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}u \\ b_{53} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}v \\ b_{54} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}w \end{aligned}$$

B_{22}:

$$B_{22} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{\mu}{\rho}u & \frac{\mu}{\rho} & 0 & 0 & 0 \\ -\frac{4}{3}\frac{\mu}{\rho}v & 0 & \frac{4}{3}\frac{\mu}{\rho} & 0 & 0 \\ -\frac{\mu}{\rho}w & 0 & 0 & \frac{\mu}{\rho} & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & \frac{\gamma}{\mathrm{Pr}}\frac{\mu}{\rho} \end{bmatrix}$$

where:

$$\begin{aligned} b_{51} &= \frac{\mu}{\rho}\left[-\frac{\gamma}{\mathrm{Pr}}\frac{p}{(\gamma-1)\rho} - \left(u^2 + \tfrac{4}{3}v^2 + w^2\right) + \tfrac{1}{2}\tfrac{\gamma}{\mathrm{Pr}}\left(u^2+v^2+w^2\right)\right] \\ b_{52} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}u \\ b_{53} &= \left(\tfrac{4}{3} - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}v \\ b_{54} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}w \end{aligned}$$

B_{33}:

$$B_{33} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{\mu}{\rho}u & \frac{\mu}{\rho} & 0 & 0 & 0 \\ -\frac{\mu}{\rho}v & 0 & \frac{\mu}{\rho} & 0 & 0 \\ -\frac{4}{3}\frac{\mu}{\rho}w & 0 & 0 & \frac{4}{3}\frac{\mu}{\rho} & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & \frac{\gamma}{\mathrm{Pr}}\frac{\mu}{\rho} \end{bmatrix}$$

where:

$$\begin{aligned} b_{51} &= \frac{\mu}{\rho}\left[-\frac{\gamma}{\mathrm{Pr}}\frac{p}{(\gamma-1)\rho} - \left(u^2 + v^2 + \tfrac{4}{3}w^2\right) + \tfrac{1}{2}\tfrac{\gamma}{\mathrm{Pr}}\left(u^2+v^2+w^2\right)\right] \\ b_{52} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}u \\ b_{53} &= \left(1 - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}v \\ b_{54} &= \left(\tfrac{4}{3} - \tfrac{\gamma}{\mathrm{Pr}}\right)\frac{\mu}{\rho}w \end{aligned}$$

References

[1]De Rango, S. and Zingg, D. W., "Higher-order spatial discretization for turbulent aerodynamic computations," *AIAA Journal*, Vol. 39, No. 7, July 2001, pp. 1296–1304.

[2]Dias, S. C. and Zingg, D. W., "A high-order parallel Newton-Krylov flow solver for the Euler equations," *19th AIAA Computational Fluid Dynamics Conference*, No. AIAA–2009–3657, San Antonio, Texas, United States, June 2009.

[3]Hicken, J. E. and Zingg, D. W., "A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms," *AIAA Journal*, Vol. 46, No. 11, Nov. 2008, pp. 2773–2786.

[4]Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236.

[5]Hesthaven, J. S., "A stable penalty method for the compressible Navier-Stokes equations: III. Multidimensional domain decomposition schemes," *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 1998, pp. 62–93.

[6]Carpenter, M. H., Nordström, J., and Gottlieb, D., "A stable and conservative interface treatment of arbitrary spatial accuracy," *Journal of Computational Physics*, Vol. 148, No. 2, 1999, pp. 341–365.

[7]Nordström, J. and Carpenter, M. H., "High-order finite difference methods, multidimensional linear problems, and curvilinear coordinates," *Journal of Computational Physics*, Vol. 173, No. 1, 2001, pp. 149–174.

[8]Svärd, M., Carpenter, M. H., and Nordström, J., "A stable high-order finite difference scheme for the compressible Navier-Stokes equations, far-field boundary conditions," *Journal of Computational Physics*, Vol. 225, No. 1, July 2007, pp. 1020–1038.

[9]Svärd, M. and Nordström, J., "A stable high-order finite difference scheme for the compressible Navier-Stokes equations, no-slip wall boundary conditions," *Journal of Computational Physics*, Vol. 227, No. 10, May 2008, pp. 4805–4824.

[10]Nordström, J., Gong, J., van der Weide, E., and Svärd, M., "A stable and conservative high order multi-block method for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 24, 2009, pp. 9020–9035.

[11]Nordström, J. and Carpenter, M. H., "Boundary and interface conditions for high-order finite-difference methods applied to the Euler and Navier-Stokes equations," *Journal of Computational Physics*, Vol. 148, No. 2, 1999, pp. 621–645.

[12]White, F. M., *Viscous Fluid Flow*, McGraw–Hill Book Company, New York, 1974.

[13]Kreiss, H.-O. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations*, edited by C. de Boor, Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[14]Strand, B., "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47–67.

[15]Jameson, A., Schmidt, W., and Turkel, E., "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes," *14th Fluid and Plasma Dynamics Conference*, Palo Alto, CA, 1981, AIAA Paper 81–1259.

[16]Pulliam, T. H., "Efficient solution methods for the Navier-Stokes equations," Tech. rep., Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Rhode-Saint-Genèse, Belgium, Jan. 1986.

[17]Mattsson, K., Svärd, M., and Shoeybi, M., "Stable and accurate schemes for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 227, No. 4, 2008, pp. 2293–2316.

[18]Svärd, M., Mattsson, K., and Nordström, J., "Steady-state computations using Summation-by-Parts operators," *Journal of Scientific Computing*, Vol. 24, No. 1, July 2005, pp. 79–95.

[19]Lomax, H., Pulliam, T. H., and Zingg, D. W., *Fundamentals of Computational Fluid Dynamics*, Springer–Verlag, Berlin, Germany, 2001.

[20]Nichols, J. and Zingg, D. W., "A three-dimensional multi-block Newton-Krylov flow solver for the Euler equations," *17th AIAA Computational Fluid Dynamics Conference*, No. AIAA–2005–5230, Toronto, Canada, June 2005.

[21]Pueyo, A. and Zingg, D. W., "Efficient Newton-Krylov solver for aerodynamic computations," *AIAA Journal*, Vol. 36, No. 11, Nov. 1998, pp. 1991–1997.

[22]Kam, D. C. W., *A three-dimensional Newton-Krylov Navier-Stokes flow solver using a one-equation turbulence model*, Master's thesis, University of Toronto, Toronto, Ontario, Canada, 2007.

[23]Kim, D. B. and Orkwis, P. D., "Jacobian update strategies for quadratic and near-quadratic convergence of Newton and Newton-like implicit schemes," *31st AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA–93–0878, Reno, Nevada, 1993.

[24]Saad, Y. and Sosonkina, M., "Distributed Schur complement techniques for general sparse linear systems," *SIAM Journal of Scientific Computing*, Vol. 21, No. 4, 1999, pp. 1337–1357.

[25]Keyes, D. E., "Aerodynamic applications of Newton-Krylov-Schwarz solvers," *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, Springer, New York, 1995, pp. 1–20.

[26]Mulder, W. A. and van Leer, B., "Experiments with implicit upwind methods for the Euler equations," *Journal of Computational Physics*, Vol. 59, No. 2, 1985, pp. 232–246.

[27]Zingg, D. W., De Rango, S., Nemec, M., and Pulliam, T. H., "Comparison of several spatial discretizations for the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 160, No. 2, May 2000, pp. 683–704.

[28]Roos, F. W. and Willmarth, W. W., "Some experimental results on sphere and disk drag," *AIAA Journal*, Vol. 9, No. 2, Feb. 1971, pp. 285–291.

[29]Taneda, S., "Experimental investigation of the wake behind a sphere at low Reynolds numbers," *Journal of the Physical Society of Japan*, Vol. 11, No. 10, Oct. 1956, pp. 1104–1108.