An Evaluation of Implicit Time Integration Schemes for Discontinuous High Order Methods

Cheng Zhou¹ (✉) and Z. J. Wang²
University of Kansas, Lawrence, KS 66044

For many flow problems, implicit time integration schemes are known to achieve solution convergence in acceptable time, and significantly reduce the memory cost comparing with explicit schemes. Many implicit schemes have been developed recently for high-order discontinuous methods. However, it is unclear which is the most efficient, robust and general. We carry out a comparative study of several implicit time integration schemes with the Correction Procedure via Reconstruction (CPR) high-order discretization. These schemes include the Lower-Upper Symmetric Gauss-Seidel (LU-SGS), a matrix-free GMRES solver with LU-SGS as a preconditioner, and a GMRES linear equation solver with different preconditioners within PETSc (Portable, Extensible Toolkit for Scientific Computation). Through a series of numerical experiments, the performance of these implicit time integration schemes is evaluated for high-order methods.

I. Introduction

Improving computational efficiency is of great importance to the CFD community, and the use of implicit time integration schemes is one solution since they can advance the solution with significantly larger time steps comparing with the explicit schemes. On the other hand, due to high-order space discretizations, very large Jacobian matrices are produced, which make implicit schemes expensive in both memory and CPU time. Upon this trade-off between memory and efficiency, many implicit schemes have been developed, either to decrease memory requirement¹ or to accelerate convergence to steady state²³⁴⁵⁶. However, it is unclear which is the most efficient, robust and general. This paper performs a comparative study of several well-known or recently developed implicit time-integration schemes with the Correction Procedure via Reconstruction (CPR) high-order discretization for unstructured grids. The following implicit schemes are considered in the study, namely the Lower-Upper Symmetric Gauss-Seidel (LU-SGS), a matrix-free GMRES solver with LU-SGS as a preconditioner, and a GMRES linear equation solver with different preconditioners available in PETSc (Portable, Extensible Toolkit for Scientific Computation) library⁷, such as Jacobian, ILU and ILU(1) and line preconditioner. PETSc is used in solving the fully linearized equations with different preconditioners to compare with the other two schemes in terms of efficiency, memory cost and robustness.

The original LU-SGS approach was developed by Yoon and Jameson⁸, which employs a special first-order approximation in linearizing the left-hand side to reduce the block diagonal matrices to diagonal matrices. Through forward and backward sweeps in one Gauss-Seidel iteration, all of the off-diagonal matrices still contribute to the implicit operator. As a result, LU-SGS does not require any extra memory compared to explicit schemes, but can drive a steady solution to convergence much faster than explicit schemes. It was also found that the convergence rate could degrade considerably for viscous flow problems, especially after several orders of convergence⁹. To further improve the convergence rate, Chen and Wang⁶ and Jameson and Caughey⁷ developed a block (preconditioned) non-linear LUSGS (BLU-SGS) approach. For a wide variety of flow problems, BLU-SGS demonstrated low storage requirement (only the diagonal block matrix is stored), and faster convergence to steady state than the original LU-SGS. Details on a very compact form of the non-linear BLU-SGS used in this study can be found in Reference [8].

In the conventional GMRES approach, the full implicit matrix needs to be stored including the lower and upper matrices, which may require prohibitive storage in large 3D computations. This deficiency can be alleviated by various matrix-free implementations of GMRES.⁴⁶ Every successful GMRES iterative approach needs a good preconditioner. To achieve fast convergence, the preconditioner should be “close” to the implicit operator, but much easier to invert. Bassi and Rebay successfully developed a preconditioned GMRES approach for the DG method¹¹ to solve compressible flow. A matrix-free Krylov approach was developed by Rasetarinera and

¹ Graduate Student, Department of Aerospace Engineering, AIAA Member, email: chengzhou@ku.edu
² Spahr Professor and Chair, Department of Aerospace Engineering, AIAA Associate Fellow, email: zjw@ku.edu
Hussain[19] with an efficient LU-SGS preconditioner. This implementation demonstrates a considerable saving in storage compared to the standard GMRES.

In the present paper, according to a series of numerical experiments on the BLU-SGS approach and GMRES with BLU-SGS as the preconditioner (GMRES-BLUSGS), it is found that GMRES-BLUSGS, which only requires storing main block diagonal matrix, is a good compromise between efficiency and storage requirement. However, it failed to achieve convergence with the CPR high-order discretization for a flat plate boundary layer flow. To remedy this problem, we resort to the preconditioned GMRES linear solvers available in the PETSc library (PGMRES), which generally requires the derivative of the Jacobian matrix that can be approximated with first order finite differences. PETSc provides a robust and flexible suit of data-structure-neutral numerical routines for Newton-like methods, which enable various storage schemes and solvers through a single user interface[12][13][14][20].

The remainder of this paper is organized as follows. Section II describes the CPR high-order method to discretize the compressible Navier-Stokes equations, the BLU-SGS approach for time-integration, the matrix-free GMRES approach with BLU-SGS as a preconditioner, and the GMRES linear equation solver within PETSc paired with three different preconditioners (i.e. Jacobi (PGMRES-JACOBI), incomplete lower-upper decomposition(PGMRES-ILU), and line preconditioner (PGMRES-LINE)) in solving fully linearized equations.

Section III presents and compares the results for the following flows based on efficiency, memory requirement and robustness:

- A. Laminar boundary layer on a flat plate
- B. Subsonic, inviscid flow over a NACA 0012 airfoil
- C. Subsonic, viscous flow over a NACA 0012 airfoil

Through a series of numerical experiments, efficient, robust, and general implicit time integration schemes are suggested for high-order method in Section IV. Future studies will extend this work to additional supersonic, turbulent problems and 3D problems.

II. Numerical Methods

A. CPR Framework for Space Discretization

The CPR formulation was originally developed by Huynh in [21][22], and extended to simplex and hybrid elements by Wang & Gao in [12]. In the CPR method, the degrees-of-freedom (DOFs) are the state variables at a pre-defined nodal set named solution points (SPs), where the differential form of the governing equation is solved. In the present study, the solution points are chosen as the Gauss-Lobatto quadrature points. Details about CPR approach can be found in [13]. Here we directly apply the CPR formula to the Navier-Stokes equations, which can be written as

\[
\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) = \nabla \cdot \mathbf{F}(Q, \nabla Q), \tag{1}
\]

with proper initial and boundary conditions, where \( Q \) is the state vector, \( \mathbf{F} = (F, G) \) is the inviscid flux vector, \( F^v(Q, \nabla Q) \) the viscous flux vector.

First, following [23], we introduce a new variable \( \vec{R} \)

\[
\vec{R} = \nabla Q. \tag{2}
\]

Let \( \vec{R}_i \) be an approximation of \( \vec{R} \) on \( V_i \). Many studies have found that the obvious choice of \( \vec{R}_i = \nabla Q_i \) is not appropriate. Instead, the computation of \( \vec{R}_i \) needs to involve data from neighboring cells.

The CPR formulation applied to the above equations can be expressed as

\[
\frac{\partial Q_{ij}}{\partial t} + \prod_i (\nabla \cdot \mathbf{F}(Q_i)) - \prod_i (\nabla \cdot \mathbf{F}^v(Q_i, \vec{R}_i)) + \frac{1}{|V_i|} \sum_j \sum \alpha_{j,f,j} [\mathbf{F}^n f_j - [\mathbf{F}^{v,n}] f_j] S_j = 0, \tag{3}
\]

\[
\vec{R}_{ij} = (\nabla Q_i)_j + \frac{1}{|V_i|} \sum j \prod f \sum \alpha_{j,f,j} Q_{ij}^{com} - Q_{ij} f_j \vec{n}_f S_f, \tag{4}
\]

where \( \alpha_{j,f,j} \) are lifting constants independent of the solution, \( S_f \) is the face area, \( |V_i| \) is the volume of \( V_i \), and

\[
[F^n]^j = F^n_{ij} - F^n_{ij}(Q_i), \tag{3}
\]

\[
[F^{v,n}]_j = \mathbf{F}^v(Q_{ij}^{com}, \nabla Q_{ij}^{com}) \vec{n}_f - \mathbf{F}^v(Q_i, \vec{R}_i)| \vec{n}_f, \tag{4}
\]

\[
\mathbf{F}(Q_i, \vec{R}_i), \tag{4}
\]

\[
\mathbf{F}(Q_i, \vec{R}_i), \tag{4}
\]

\[
\mathbf{F}(Q_i, \vec{R}_i), \tag{4}
\]
with \( Q^c_{f,\text{com}} \) and \( \mathbf{\tilde{Q}} Q^c_{f,\text{com}} \) the common solution and gradient on interface \( f \) respectively, and \( Q_{c,f} \) is the solution with in cell \( i \) on FP \( l \) of face \( f \) or the trace of \( Q \) on \( f \). \( F^n \) is the normal flux on the interface. \( F^n_{\text{com}} \) denotes a common flux on the interface reconstructed by any Riemann solver.

\( \Pi \) and \( \Pi_e \) are the projection operator for the divergence of inviscid flux vector and viscous flux vector. Two efficient approaches on how to compute \( \Pi_f(\mathbf{\tilde{V}} \cdot \mathbf{F}(Q)) \) are developed in [15], namely, Lagrange polynomial approach (LP) and chain rule approach (CR). The computation of \( \Pi_f(\mathbf{\tilde{V}} \cdot \mathbf{F}(Q, \mathbf{\tilde{R}})) \) follows the LP approach.

Various schemes for viscous fluxes differ in how the common solution \( Q^c_{f,\text{com}} \) and the common gradient \( \mathbf{\tilde{Q}} Q^c_{f,\text{com}} \) are defined. In present study, we employ the BR2\(^{[14]} \) scheme to compute the common solution and gradient on interface. More other schemes can be found in [16] [17] [18] [19].

**B. Time Integration Schemes**

At each cell \( c \), applying the backward Euler differencing for the time derivative in Eq. (3,4) gives a nonlinear system of

\[
\frac{Q^{n+1}_c - Q^n_c}{\Delta t} = R_i(Q^{n+1}_c) \tag{7}
\]

Let \( \Delta Q_c = Q^{n+1}_c - Q^n_c \) and linearizing the residual using a Taylor expansion, we obtain

\[
R_i(Q^{n+1}_c) \approx R_i(Q^n_c) + \frac{\partial R}{\partial Q_c} \Delta Q_c + \sum_{nb_c} \frac{\partial R}{\partial Q_{ab}} \Delta Q_{ab}, \tag{8}
\]

where \( nb \) denotes all the neighboring cells contributing to the residual of cell \( c \). Substituting (8) into (7), the fully linearized equations for (7) can be written as

\[
\left( \frac{I}{\Delta t} - \frac{\partial R}{\partial Q_c} \right) \Delta Q^n_{ab} - \sum_{nb_c} \frac{\partial R}{\partial Q_{ab}} \Delta Q_{ab} = R_i(Q^n_c) \tag{9}
\]

**1. BLU-SGS Approach**

Since storing all the implicit Jacobian matrices in the algebraic system is expensive, we employ a LU-SGS scheme to solve (9), i.e., we use the most recent solution for the neighboring cells \( Q_{ab}^* \),

\[
\left( \frac{I}{\Delta t} - \frac{\partial R}{\partial Q_c} \right) \Delta Q^{(k+1)}_{ab} = R_i(Q^n_c) + \sum_{nb_c} \frac{\partial R}{\partial Q_{ab}} \Delta Q^*_{ab} \tag{10}
\]

The matrix

\[
D = \left( \frac{I}{\Delta t} - \frac{\partial R}{\partial Q_c} \right) \tag{11}
\]

is the element Jacobian matrix. Equation (10) is then solved with an exact LU decomposition solver. To avoid storing the Jacobian matrices for the neighboring cells \( \frac{\partial R}{\partial Q_{ab}} \), we further introduce inner iteration as follows,

Note that

\[
R_i(Q^n_c) + \sum_{nb_c} \frac{\partial R}{\partial Q_{ab}} \Delta Q^*_{ab} = R_i(Q^n_c, \{Q^*_{ab}\}) + \sum_{nb_c} \frac{\partial R}{\partial Q_{ab}} \Delta Q^*_{ab} \\
\approx R_i(Q^n_c, \{Q^*_{ab}\}) \approx R_i(Q^n_c, \{Q^*_{ab}\}) - \frac{\partial R}{\partial Q_c} \Delta Q^*_{ab} = R_i(Q^n_c) - \frac{\partial R}{\partial Q_c} \Delta Q^*_{ab} \tag{12}
\]

Let \( \Delta Q^{(k+1)} = \Delta Q^{(k+1)} - \Delta Q^* \), equation (10) and (12) together give a very compact form of the non-linear BLU-SGS

\[
\left( \frac{I}{\Delta t} - \frac{\partial R}{\partial Q_c} \right) \Delta Q^{(k+1)}_{ab} = R_i(Q^n_c) - \frac{\Delta Q^*}{\Delta t} \tag{13}
\]
Eq. (13) is then solved through numbers of symmetric forward and backward sweeps with a prescribed tolerance \( \varepsilon \) for convergence. Note that at each time step, if Eq. (13) is solved to machine zero, the unsteady residual \( R(Q) = \frac{\Delta Q}{\Delta t} \) is zero. The initial guess for \( Q^{n+1} \) then can be set to \( Q^n \). Consequently, the initial unsteady residual is the same as the steady residual at the last time step, i.e. \( R(Q^n) \). Then we can monitor the unsteady residual for convergence. This also indicates that, it is not necessary to solve the unsteady residual to machine zero for steady state problems. In fact, it can be more efficient to set criteria for maximum number of sweeps for (13), and move to the next time step.

It is very easy to implement this numerical approach, but the computational cost is expensive because all the changes of the degrees of freedom need to be considered. To further improve the computational efficiency, the element matrices \( D \) are frozen for many intervals of time steps in this study.

2. GMRES Approach with BLU-SGS Preconditioner

The Non-linear system for all DOFs can be written as \( \frac{Q^{n+1} - Q^n}{\Delta t} = R(Q^{n+1}) \), whose linearized version is

\[
A\Delta Q = R(Q^n),
\]

where \( \Delta Q = Q^{n+1} - Q^n \) and \( A = (I / \Delta t + \partial R / \partial Q) \).

The left-hand side (LHS) implicit operator is a very large sparse block matrix, of which the direct inversion is usually very expensive. Based on the fact that all the operations on matrix \( A \) in the GMRES is associated with the computation of matrix-vector products, which can be approximated using finite difference, the GMRES approach can be implemented without forming the Jacobian matrix explicitly or storing \( A \). Various matrix-free implementations of GMRES have been developed\(^9\). Any successful GMRES iterative approach with fast convergence needs a good pre-conditioner \( P \), which should be "close" to the implicit operator \( A \), but much easier to invert. An equivalent preconditioned form of (14) is

\[
P^{-1}A\Delta Q = P^{-1}R(Q^n),
\]

Note that the pre-conditioning matrix \( P \) must be formed and stored. The efficient LU-SGS preconditioner was demonstrated very good convergence properties for high-order DG schemes with polynomial degree to 3\(^9\). In this study, BLU-SGS approach is used as the preconditioner, which means \( P \) is solved following Eq.(13).

3. GMRES Solver within PETSc Coupled with Different Preconditioners

PETSc integrates series of components in a highly efficient way, including low-level distributed data structures for meshes, vectors, and matrices and high-level linear, nonlinear, and times stepping solvers. The algorithmic source code is written in high-level abstractions so that it can be easily understood and modified. This hierarchy makes code reuse possible and promotes code flexibility. The Newton-based methods within PETSc are written in a data-structure-neutral form that uses abstractions for vectors, matrices, and linear solvers \(^{20}\). In the present study, the PETSc library is applied as the linear equation solver with the CPR framework. The GMRES solver available in PETSc is compared with the existing GMRES approach with the BLU-SGS preconditioner. Only residual evaluations \( R(Q) \) and Jacobian at given state vectors \( dR/dQ \) in the user routines are called for the subroutines with PETSc library.

At first, numerical experiments indicate that for the laminar boundary layer on a flat plate, neither BLU-SGS nor GMRES-BLUSGS (storing main block diagonal matrix only) can achieve convergence. To remedy this problem, PETSc is directly applied to solve the non-linear system of fully linearized equations (14), which demonstrates a fast convergence. However, the computational cost of this approach is expensive because all the changes of the degrees of freedom need to be considered for every time step. To further improve the computational efficiency, the implicit operator matrix is frozen for several time steps in this study.

Thanks to the flexibility of PETSc, three different preconditioned GMRES within PETSc are compared with the approaches described in section II.B.1 and II.B.2 through solving Eq. (14), namely, Jacobi (PGMRES-JACOBI), incomplete lower-upper decomposition (PGMRES-ILU), and line preconditioner (PGMRES-LINE), which are shown to be the most appropriate preconditioners for different compressible flows. We remark that the line preconditioner is based on user routines that are called by PETSc subroutines rather than PETSc built-in preconditioners such as JACOBI and ILU. The lines in the present study are generated based solely on the mesh aspect-ratio (AR) to counter the stiffness associated with high AR elements. Fig. 1 and Fig. 2 illustrate the lines for several meshes, which are marked in green.
4. Memory requirement

In the Jacobian matrix, the number of entries for each cell is \( n_{\text{dof}} \times n_{\text{dof}} \), where \( n_{\text{dof}} \) is the number of degree of freedoms in one cell. For both BLU-SGS and GMRES- BLUSGS, only the main block diagonal matrices are stored.

\[
\text{Memory(BLU-SGS)} = n_{\text{Elems}} \times \text{Memory(cell)} \\
\text{Memory(cell)} = n_{\text{dof}} \times n_{\text{dof}}
\]

However, the GMRES solver needs extra memory to store the restart Krylov space vectors. In the present study, \( k = 30 \) for inviscid flow and \( k = 90 \) for viscous flow, which almost doubles the memory requirement (e.g. \( n_{\text{dof}} = 64 \) for p3 scheme on quadrilateral mesh).

\[
\text{Memory(GMRES - BLUSGS)} = n_{\text{Elems}} \times \text{Memory(cell)} + k \times n_{\text{Elems}} \times n_{\text{dof}} \\
\text{Memory(BLUSGS)} = 1 + \frac{k}{n_{\text{dof}}}
\]

The preconditioned GMRES approach within PETSc proposed here is solving the non-linear system of fully linearized equations (9), which means the sparse Jacobian matrix including all neighbors is stored completely. Thus, depending on the considered element type, we get the overall memory storage (assuming large total number of elements compared to the boundary elements) as

\[
\text{Memory(PGMRES)} = n_{\text{Elems}} \times \text{Memory(cell)} \times (1 + n_{\text{sides}}) + k \times n_{\text{Elems}} \times n_{\text{dof}}
\]

where \( n_{\text{sides}} \) is the number of sides for the element type (e.g. \( n_{\text{sides}} = 4 \) for the quadrilateral cell). It is clear that, solving the full Jacobian matrix drastically amplifies the memory cost (excluding the cost for the preconditioner).

\[
\frac{\text{Memory(PGMRES)}}{\text{Memory(BLUSGS)}} = 1 + n_{\text{sides}} + \frac{k}{n_{\text{dof}}}
\]

III. Numerical Results

This section presents high-order CPR solutions of steady problems in order to assess the efficiency and robustness of the proposed five time-integration schemes and preconditioners: BLU-SGS, GMRES-BLUSGS,
PGMRES-JACOBI, PGMRE-ILU and PGMRES-LINE. If not stated otherwise, the following results have been obtained when the steady residual is reduced by 10 orders of magnitude.

The main aim of the first case, the flow over a flat plate, is to demonstrate that solving main block diagonal matrix only is not sufficient for boundary layer dominated flow though the memory cost is much less, while the second and third test cases, the inviscid and viscous flow over NACA0012 airfoil, have been chosen to assess the proposed five time-integration schemes and preconditioners.

For all the computations here reported the initial flow field has been set to uniform freestream conditions. The computational efficiency is compared based on converged iterations as well as work units, which was defined in the 1st International Workshop on High-Order CFD Methods.[24] All the cases are from the Workshop.

The parameters of the restarted GMRES solver were set to 30 Krylov space vectors for inviscid flow and 90 for viscous flow, 200 maximum iterations and 10^{-5} relative convergence tolerance. The initial time step used is the largest time step that ensuring stability for each case. The growth of the time step is related to the relative residual according to the following relations:

\[ dt = \min\left(dt_0, f^\beta, dt_{\text{max}}\right) \]
\[ f = \frac{\|R(Q^\alpha)\|_2}{\|R(Q^\beta)\|_2} \]

where \( dt_0 \) is the initial time step, \( dt_{\text{max}} = 1.e20 \), and \( \beta = 1.05 \) For computation efficiency, the time step was increased to infinity (1.e20 in present study) after relative residual drops 3 orders magnitude.

A. Laminar boundary layer on a flat plate

The laminar boundary layer on a flat plate is simulated with freestream Mach number 0.5, angle of attack 0°, Reynolds number (based on the plate length) \( 1x10^6 \). This simulation is run on both a coarse mesh (h1) of 560 cells and a finer mesh (h2) of 2240 cells with either 3rd-order or 4th-order space discretization. Both meshes are standard meshed used in the 1st International Workshop on High-Order CFD Methods. Since this simulation can achieve full convergence (10 order residual reduction) only through solving the full linearized system, Fig. 3 displays the skin friction coefficient using PGMRES-LINE as the time-integration scheme. Comparing with the h1p3 result, simulations with higher order h1p4 and finer mesh h2p3 achieve a converged result.

![Figure 3. The skin friction coefficient of PGMRES-LINE for laminar flow on a flat plate](image)

Several other schemes are compared in Table 1 in terms of work units, iterations and initial time steps to achieve convergence. “Div” denotes diverged simulation. “C-3” means the residual can only converge to \( 1x10^{-3} \) and keeps oscillating at this level. “Inf” means infinity.

Table 1 shows that the simulation of the laminar boundary layer on a flat plat can converge to \( 1x10^{-10} \) only through solving the fully linearized equations. As for preconditioners, based on the 3rd-order results on different meshes Table 1(a) and Table 1(c), ILU(0) and line preconditioner gives the best performance for this case, while the JACOBI preconditioner cannot achieve convergence and ILU(1) takes twice as much the work units. Considering that LINE is a user-provided preconditioner and ILU is PETSc built-in preconditioner, the slightly more work units, namely 12% and 8%, can result from the optimized implementation of PETSc. Additionally, the result in Table 1(b), which is simulated by 4th-order scheme on coarse mesh h1, demonstrates that LINE is more robust than ILU for a stiffer high-order scheme as the flow is strongly coupled in the boundary layer direction.
Inviscid flow over the NACA0012 airfoil

Subsonic inviscid flow over the NACA0012 airfoil is simulated with freestream Mach number 0.5, and angle of attack $2^\circ$. This simulation is run on h2 mesh of 2240 cells with both 3rd-order and 4th-order space discretization. Fig. 4 displays the converged density field simulated with the 3rd-order scheme.

![Figure 4. The density field solutions of GMRES-BLUSGS for inviscid flow over NACA0012](image)

Several other schemes applied to this case are compared in Table 2 in terms of work units, iterations and initial time steps to achieve convergence. “Div” denotes diverged simulation.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>Inf</td>
<td>7</td>
<td>3.803</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>Inf</td>
<td>7</td>
<td>8.347</td>
<td>2.19</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>Inf</td>
<td>8</td>
<td>4.246</td>
<td>1.12</td>
</tr>
</tbody>
</table>

(a) 3rd-order scheme on 560 cell mesh

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>C-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>C-6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>DIV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>Inf</td>
<td>8</td>
<td>12.293</td>
<td></td>
</tr>
</tbody>
</table>

(b) 4th-order scheme on 560 cell mesh

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>C-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>Inf</td>
<td>7</td>
<td>23.704</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>Inf</td>
<td>7</td>
<td>35.657</td>
<td>1.50</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>Inf</td>
<td>8</td>
<td>25.742</td>
<td>1.08</td>
</tr>
</tbody>
</table>

(c) 3rd-order scheme on 2240 cell mesh

Table 1. Work units, iterations and initial time steps of different schemes for flow on a flat plate

B. Inviscid flow over the NACA0012 airfoil

Subsonic inviscid flow over the NACA0012 airfoil is simulated with freestream Mach number 0.5, and angle of attack $2^\circ$. This simulation is run on h2 mesh of 2240 cells with both 3rd-order and 4th-order space discretization. Fig. 4 displays the converged density field simulated with the 3rd-order scheme.
Table 2 shows that among the six schemes, BLUSGS and PGMRES-JACOBI is the worst scheme considering that they cannot even achieve convergence. While GMRES-BLUSGS takes more iterations to converge, it is still the most efficient scheme for this case in work units. GMRES solver within PETSc solving the full linearized system does not perform as well as the GMRES-BLUSGS, and the more the total degree of freedoms the worse it performs since more off-diagonal entries in the Jacobian matrix need to be stored and solved. Among the three different preconditioners coupled with PGMRES, ILU and LINE give the best performance and LINE is more robust for higher-order scheme, which agrees with the conclusion in section III.B. Table 2 also shows that ILU(1) only amplifies work units in the reason of keeping more entries during decomposition, while does not improve robustness in sense of the same or even more iterations than ILU.

### C. Viscous flow over the NACA0012 airfoil

Subsonic viscous flow over the NACA0012 airfoil is simulated with freestream Mach number 0.5, angle of attack 1° and Reynolds number Re=5000. This simulation is run on both a coarse mesh of 560 cells and a finer mesh of 2240 cells with 3rd-order space discretization. Fig. 5 displays the converged density and Mach contours on the finer mesh.

---

**Table 2. Work units, iterations and initial time steps of different schemes for inviscid flow over NACA0012 airfoil**

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>DIV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>2.5E-4</td>
<td>31</td>
<td>8.474</td>
<td>0.30</td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>0.02</td>
<td>19</td>
<td>27.918</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>0.02</td>
<td>24</td>
<td>94.521</td>
<td>3.43</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>0.02</td>
<td>22</td>
<td>34.926</td>
<td>1.28</td>
</tr>
</tbody>
</table>

(a) 3rd-order scheme

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>DIV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>2.5E-3</td>
<td>44</td>
<td>151.852</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>2.5E-3</td>
<td>44</td>
<td>603.029</td>
<td>3.97</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>2.5E-3</td>
<td>34</td>
<td>122.171</td>
<td>0.80</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>2.5E-3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) 4th-order scheme

Figure 5. The density field solutions of GMRES-BLUSGS for viscous flow over NACA0012

Several other schemes applied to this case are compared in Table 3 in terms of work units, iterations and initial time steps to achieve convergence.


<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>1.E-3</td>
<td>14</td>
<td>4.664</td>
<td>0.90</td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>5.</td>
<td>11</td>
<td>5.169</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>5.</td>
<td>19</td>
<td>19.261</td>
<td>3.72</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>5.</td>
<td>13</td>
<td>5.656</td>
<td>1.09</td>
</tr>
</tbody>
</table>

(a) Coarse mesh of 560 cells

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time step</th>
<th>Iterations</th>
<th>Work units</th>
<th>Comparison to ILU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUSGS</td>
<td>C-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMRES-BLUSGS</td>
<td>5.E-4</td>
<td>22</td>
<td>31.362</td>
<td>1.28</td>
</tr>
<tr>
<td>PGMRES-JACOBI</td>
<td>C-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGMRES-ILU</td>
<td>5.</td>
<td>11</td>
<td>24.585</td>
<td>1</td>
</tr>
<tr>
<td>PGMRES-ILU1</td>
<td>20.</td>
<td>14</td>
<td>67.821</td>
<td>2.78</td>
</tr>
<tr>
<td>PGMRES-LINE</td>
<td>10.</td>
<td>13</td>
<td>37.951</td>
<td>1.55</td>
</tr>
</tbody>
</table>

(b) Finer mesh of 2240 cells

Table 3. Work units, iterations and initial time steps of different schemes for viscous flow over NACA0012 airfoil

Table 3 shows that among these six schemes, BLUSGS and PGMRES-JACOBI are still the worst scheme considering they cannot even achieve convergence. PGMRES-ILU and PGMRES-LINE demonstrates to be the most robust schemes with larger initial time step, less iterations and competitive work units. Since GMRES-BLUSGS requires much less memory cost and comparable CPU time, it is recommended for memory-limited numerical simulations. Among the three different preconditioners coupled with PGGMRES, given ILU shows almost the same performance as line precondition, LINE is more stable in the sense that a larger initial time step can be used for finer mesh. Requiring almost three times work units, ILU(1) is not as robust as ILU(0) and line preconditioner, which agrees with III.A and III.B.

Comparing III.B and III.C., GMRES-BLUSGS does not give the impressive advantage for viscous flow as that for inviscid flow in terms of CPU time. This is because that the flux gradient in viscous flow requires more information from neighboring cells and thus the Jacobian matrix is not diagonal dominant as that of inviscid flow, solving the full linearized system is more reasonable and obviously accurate than only storing the main diagonal block matrix, which also explains the conclusion in III.A. that the simulation of laminar flow over a flat plate can achieve full convergence only through solving the full linearized system.

IV. Conclusions and Future Plans

A comparative study of several well-known or recently developed implicit time integration schemes with the CPR discretization is carried out in this study. The BLU-SGS approach, the matrix-free GMRES with BLU-SGS as a preconditioner, and the GMRES solver available in PETSc with three different preconditioners (JACOBI, ILU, LINE) are adopted in the development. The BLUSGS scheme and GMRES approach with BLU-SGS as precondition only store the main diagonal matrix to save memory cost. To achieve convergence for the laminar boundary layer on a flat plate, the fully linearized system is solved with the GMRES solver available in PETSc. The proposed time-integration schemes are tested for several cases described in Section I. For inviscid and viscous flow over NACA0012 airfoil, GMRES with BLU-SGS as the preconditioner is demonstrated the most efficient and economical. However, in sense of robustness, GMRES solver solving the fully linearized system is recommended for viscous flow. Considering the convergence for laminar flow over a flat plate, LINE preconditioned GMRES solver solving the fully linearized system is the most efficient, robust and general scheme for viscous flow. In order to generalize this conclusion, future studies will extend this work to additional supersonic, turbulent problems and 3D problems.

V. Acknowledgements

The study was supported by NASA under grant NNX12AK04A and partly by AFOSR under grant FA95501210286. The support is greatly appreciated. The authors wish to thank Dr. Meilin Yu and Mr. Lei Shi for their reviews and comments.
VI. Reference