KRYLOV METHODS FOR COMPRESSIBLE FLOWS

M. D. Tidriri [*]
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center, Hampton, VA 23681-0001

Abstract

In this paper we investigate the application of Krylov methods to compressible flows, and the effect of implicit boundary conditions on the implicit solution of nonlinear problems. Two defect-correction procedures, namely, Approximate Factorization (AF) for structured grids, and ILU/GMRES for general grids are considered. Also, considered here, is Newton-Krylov matrix-free methods that we combine with the use of mixed discretization schemes in the implicitly defined Jacobian and its preconditioner. Numerical experiments that show the performance of our approaches are then presented.

[*]This work was supported by the National Aeronautics and Space Administration under NASA contract NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681-0001.

1 Introduction

The implicit discretization of the compressible flows leads to a large sparse linear system which needs to be solved at each time step. In the derivation of this system one often uses a defect-correction procedure, in which the left-hand side of the system is discretized using a lower order approximation than for the right-hand side. This is due to storage considerations and computational complexity, and also to the fact that the resulting lower order matrix is better conditioned than the higher order matrix. The resulting methods are only moderately implicit. In the case of structured, body-fitted grids, the linear system can easily be solved using approximate factorization (AF), which is among the most widely used methods for such grids. For unstructured grids, such techniques are no longer valid, and the system is solved using direct or iterative methods. Because of the prohibitive computational costs and large memory requirements for the solution of compressible flows, iterative methods are preferred. In these defect-correction methods, which are of practice in most CFD computer codes, the mismatch in the right and left hand side operators together with explicit treatment of the boundary conditions, lead to severely limited CFL number, which results in a slow convergence to steady state aerodynamic solutions. Many authors have tried to replace explicit boundary conditions with implicit ones (see for instance [19],[15], and [8]). They showed that high CFL number can be used, however no clear advantages in terms of CPU time as compared to explicit boundary conditions have been drawn.

We investigate here defect-correction procedures based on Krylov methods; more particularly we study the ILU/GMRES methods together with implicit treatment of the boundary conditions. We show in particular that, in the context of Krylov methods improvements in terms of convergence rate can be achieved through the use of implicit boundary conditions as compared to explicit ones. However, the attractive Newton's method convergence property cannot be approached because of the mismatch of the right and left hand side operators. Therefore we propose to use Newton-Krylov matrix-free (see [3]) methods combined with mixed discretization in the implicitly defined Jacobian-Preconditioner. Numerical experiments that show the performance of our approach are then presented.

In the next section, we describe the Newton-Krylov methodology together with mixed discretization. We present, in the section 3, the Euler solver. The description of the implicit boundary conditions is also given in the section 3. Numerical experiments are presented in the section 4. The last section is devoted to some remarks and extensions.

2 Newton-Krylov Methods

Newton-Krylov methods first proposed by Brown and Saad [3], have been investigated for compressible Euler and Navier-Stokes equations using unstructured grids in [16], [17], and [7], and for structured grids in [4], and [5].

In [16] and [17], the author has applied the matrix-free Newton-Krylov methodology to both the transonic and supersonic compressible Navier-Stokes flows. In [4] and [5], the authors have studied a convection-diffusion model problem, full potential flows and the transonic compressible Euler flows. They have also proposed and studied the Newton-Krylov-Schwarz methodology. An application to incompressbile flows is reported in [9].

Newton-like methods, together with fully implicit linear solvers allow, in principle, a more rapid asymptotic approach to steady states, $f(u) = 0$, than do time-explicit methods or semi-implicit methods based on defect correction. Strict Newton methods have the disadvantage of requiring solutions of linear systems of equations based on the Jacobian, $f_u(u)$, of the true steady nonlinear residual and are often impractical in several respects:

1. Their quadratic convergence properties are realized only asymptotically. In early stages of the nonlinear iteration, continuation or regularization is typically required in order to prevent divergence.

2. Some popular discretizations (e.g., using limiters) of $f(u)$ are nondifferentiable, leaving the Jacobian undefined in a continuous sense.

3. Even if $f_u(u)$ exists, it is often inconvenient or expensive to form either analytically or numerically, and may be inconvenient to store.

4. Even if the true Jacobian is easily formed and stored, it may have a bad condition number.

5. The most popular family of preconditioners for large sparse Jacobians on structured or unstructured two- or three-dimensional grids, the incomplete factorizations, is difficult to parallelize efficiently.

In this paper we examine how points (1), (3) and (4) may be addressed through Newton-Krylov methods. For point (2) we refer to [18], and for point (5) we refer to [4] and [5].

The memory requirements and the computational complexity for the higher-order matrix representation, whether by analytical or numerical means, are prohibitive. In this context, matrix-free Newton-Krylov methods, in which the action of the Jacobian is required only on a set of given vectors are natural. To solve the nonlinear system $f(u) = 0$, given u^0, let $u^{l+1} = u^l + \lambda^l \delta u^l$, for $l = 0, 1, \ldots,$ until the residual is sufficiently small, where δu^l approximately solves the Newton correction equation $J(u^l)\delta u^l = -f(u^l)$, and parameter λ^l is selected by some line search or trust region algorithm [6]. Krylov methods, such as the method of conjugate gradients for symmetric positive definite systems or GMRES for general nonsingular systems, find the best approximation of the solution in a relatively small-dimensional subspace that is built up from successive powers of the Jacobian on the initial residual. The Krylov solver used throughout this paper is GMRES [13].

The action of Jacobian J on an arbitrary Krylov vector w can be approximated by

$$J(u^l)w \approx \frac{1}{\epsilon}\left[f(u^l + \epsilon w) - f(u^l)\right].$$

Finite-differencing with ϵ makes such matrix-free methods potentially much more susceptible to finite word-length effects than ordinary Krylov methods [9].

The selection of an optimal parameter ϵ, is non trivial. If ϵ is too small then the rounding errors made in the numerator are amplified by a factor of order $\frac{1}{\epsilon}$ which leads to an inaccurate result. If on the other hand ϵ is too large then the approximation of $J(u^l)w$ will be poor. Any reasonable choice of ϵ should attempt to reach a compromise between these two difficulties. The technique for choosing the scalar ϵ we use here is:

$$\epsilon \;=\; \frac{\sqrt{\varepsilon_{mach}}}{\|v\|_2^2} \cdot \; \max\{|(u^l, v)|, \mathrm{typ}u^l|v|\}.$$

where $|v| = (|v_1|, ..., |v_n|)^T$, and typ$u$ is given value depending on u and the problem to be solved. We note here that GMRES may have an advantage over other Krylov methods in the matrix-free context in that the vectors v that arise in GMRES have unit two-norm, but may have widely varying scale in other Krylov methods for nonsymmetric systems. Right preconditioning spoils the perfect unit two-norm. For an extended discussion of matrix-free applications of the Jacobian in the Krylov context, see [3].

Steady aerodynamics applications require the solution of linear systems that lack strong diagonal dominance, so it is important to verify that properly-scaled matrix-free methods can be employed in this context.

Although the matrix-free method is attractive because it does not form the matrix explicitly, the matrix is still required for preconditioning purposes. In [16], [17], and [7] the authors settled for a compromise that uses a block-diagonal preconditioner. However, most preconditioners require the matrix-explicitly. This is true for ILU preconditioner. Therefore, we form only, as in defect-corection method only the matrix of a lower order system to precondition the consistent higher order system. An approximation to the Jacobian can be used to precondition the Krylov process. Examples are:

1. the Jacobian of a lower-order discretization,

2. the Jacobian of a related discretization that allows economical analytical evaluation of elements, and

3. domain-parallel preconditioners composed of Jacobian blocks on subdomains of the full problem domain.

We consider here only cases 1 and 2. Case 3 can be combined with any of the split-discretization techniques (cases 1–2), in principle and is studied in [4] and [5].

Left preconditioning of the Jacobian with an operator B^{-1} can be accommodated via

$$B^{-1}J(u^l)w \approx \frac{1}{\epsilon}\left[B^{-1}f((u^l+\epsilon w)) - \tilde{f}(u^l)\right],$$

where $\tilde{f}(u^l) = B^{-1}f(u^l)$ is stored once, and right preconditioning via

$$J(u^l)B^{-1}w \approx \frac{1}{\epsilon}\left[f((u^l+\epsilon B^{-1}w)) - f(u^l)\right].$$

Right preconditioning is preferable when the focus is on comparing different preconditioners, since the residual norm measured as a by-product in GMRES and used in the termination test is independent of any right preconditioning. On the other hand, any left preconditioning changes the residual norm estimate available as a by-product in GMRES. Left preconditioning may be preferable when GMRES is applied in practice as the solver for an inexact Newton method. When the preconditioning B^{-1} is of high quality, the left-preconditioned residual serves as an estimate of the error in the Newton update vector. This leads to a useful termination condition when Newton step acceptance tests are based on $||\delta u||$.

3 Compressible Euler Equations

3.1 Governing Equations

The non-dimensional Euler Equations in three dimensions for the dependent variable vector $Q \equiv [\rho, \rho u, \rho v, \rho w, e]^T$ are

$$Q_t + F(Q)_x + G(Q)_y + H(Q)_z = 0, \tag{1}$$

After changing the variables into the curvilinear coordinates

$$\tau = t, \xi = \xi(x,y,z), \eta = \eta(x,y,z), \zeta = \zeta(x,y,z)$$

The equations are now expressed in the strong conservation laws as

$$\tilde{Q}_\tau + (\tilde{F})_\xi + (\tilde{G})_\eta + (\tilde{H})_\zeta = 0, \tag{2}$$

where $\tilde{Q}$ and the contravariant flux vectors, $\tilde{F}$ and $\tilde{G}$, are defined in terms of the Cartesian fluxes and the Jacobian determinant of the coordinate system transformation, through

$$\begin{aligned}\tilde{Q} &= J^{-1}Q\\ \tilde{F} &= J^{-1}\left(\xi_t Q + \xi_x F + \xi_y G + \xi_z H\right)\\ \tilde{G} &= J^{-1}\left(\eta_t Q + \eta_x F + \eta_y G + \eta_z H\right)\\ \tilde{H} &= J^{-1}\left(\zeta_t Q + \zeta_x F + \zeta_y G + \zeta_z H\right).\end{aligned}$$

and

$$\begin{aligned}J &= \frac{\partial(\xi,\eta,\zeta,\tau)}{\partial(x,y,z,t)}\\ &= \det\begin{pmatrix}\xi_x & \xi_y & \xi_z & \xi_t\\ \eta_x & \eta_y & \eta_z & \eta_t\\ \zeta_x & \zeta_y & \zeta_z & \zeta_t\\ 0 & 0 & 0 & 1\end{pmatrix}\\ &= \det\begin{pmatrix}\xi_x & \xi_y & \xi_z\\ \eta_x & \eta_y & \eta_z\\ \zeta_x & \zeta_y & \zeta_z\end{pmatrix}\end{aligned}$$

3.2 Finite volume scheme

By assuming the dependent variables to be constant in the interior of cell(i,j,k), and that the flux vectors F, G, and H are constant over the constant ξ, η, and ζ surfaces of the cell, respectively, then an implicit finite volume discretization of equation (1) can be written as

$$\begin{aligned}&(Q^{n+1}_{i,j,k} - Q^{n}_{i,j,k})\Delta\xi\Delta\eta\Delta\zeta + (F^{n+1}_{i+\frac{1}{2},j,k} - F^{n+1}_{i-\frac{1}{2},j,k})\Delta\eta\Delta\zeta\Delta\tau\\ &+(G^{n+1}_{i,j+\frac{1}{2},k} - G^{n+1}_{i,j-\frac{1}{2},k})\Delta\xi\Delta\zeta\Delta\tau + (H^{n+1}_{i,j,k+\frac{1}{2}} - H^{n+1}_{i,j,k-\frac{1}{2}})\Delta\xi\Delta\eta\Delta\tau = 0 \qquad (3)\end{aligned}$$

Using the flux split formula (see below), the above equations can be written as

$$\Delta Q^n + \Delta\tau(\delta_\xi(F^+ + F^-)^{n+1} + \delta_\eta(G^+ + G^-)^{n+1} + \delta_\zeta(H^+ + H^-)^{n+1}) = 0 \qquad (4)$$

where δ_ξ, for example, is defined by

$$\delta_\xi = \frac{1}{\Delta\xi}[F_{i+1/2,j,k} - F_{i-1/2,j,k}] \qquad (5)$$

and δ_η and δ_ζ are defined analogously. The split vector for F is given by

$$F = F^+ + F^-, \tag{6}$$

with similar expressions for G and H. F^+ is associated with the eigenvalues that have positive signs and F^- is associated with the eigenvalues that have negatives signs (see Steger-Warming)., and $G^+, G^-, H^+, \text{and} H^-$ are defined analogously. The implicit split-flux discretization is given by

$$\begin{aligned}[I + \Delta\tau[(\delta_\xi(F^+ + F^-)^{n+1} + \delta_\xi(G^+ + G^-)^{n+1} + \delta_\eta(H^+ + H^-)^{n+1}] \\ = -\Delta\tau(\delta^e_\xi F^n + \delta^e_\eta G^n + \delta^e_\zeta H^n)\end{aligned} \tag{7}$$

A linearization of first order in time of the above equation yields

$$\begin{aligned}[I + \Delta\tau(\delta^i_\xi A^+\cdot + \delta^i_\xi A^-\cdot + \delta^i_\eta B^+\cdot + \delta^i_\eta B^-\cdot + \delta^i_\zeta C^+\cdot + \delta^i_\zeta C^-\cdot)\Delta Q^n \\ = -\Delta\tau(\delta^e_\xi F^n + \delta^e_\eta G^n + \delta^e_\zeta H^n)\end{aligned} \tag{8}$$

Where a distinction has been made between the implicit spatial difference operator and the explicit spatial difference operators by using supersripts i and e, respectively. The dots indicate that the difference operators apply to the product of the Jacobian matrices with ΔQ^n. The matrices $A^+, A^-, B^+, B^-, C^+, \text{and} C^-$ are defined by

$$A^+ = \frac{\partial F^+}{\partial Q}, \qquad A^- = \frac{\partial F^-}{\partial Q}, \tag{9}$$

$$B^+ = \frac{\partial G^+}{\partial Q}, \qquad B^- = \frac{\partial G^-}{\partial Q}, \tag{10}$$

$$C^+ = \frac{\partial H^+}{\partial Q}, \qquad C^- = \frac{\partial H^-}{\partial Q}. \tag{11}$$

The finite volume discretization given by equation (3) requires the numerical flux at a cell face. These fluxes are computed using the Roe's approximate Riemann solver [12]. Three limiters are employed: minmod, Superbee, and Van Leer. The Jacobians are evaluated using first-order Roe's scheme, or the flux-vector split scheme [14], which corresponds to the true partials of the positive and negative flux vectors as described earlier. However, the flux-vector split scheme has been shown to give improved convergence rates over the Roe matrices. Because the Jacobian matrices corresponding to the flux-vector split scheme work better on the left hand side in the solution matrix than the Roe matrices, this is the scheme presently being employed in the context of defect correction. This results in inconsistent left and right hand side operators.

Remark 3.1 *For most CFD codes, the implicit spatial differences are only first-order accurate. Because deriving higher-order accuracy is difficult, and the resulting matrices are very large and require a lot of storage, large operation count in its evaluation, and may be very difficult to invert.*

Remark 3.2 *For Roe's scheme, it is difficult to obtain the true Jacobian. Barth [1], has obtained such Jacobian in two dimensions. However, for three dimensions, the evaluation of such Jacobian needs large operation count.*

Following these remarks, the implicit spatial differences in equation (1) are approximated, as mentioned above, through a first-order accurate scheme.

The explicit spatial differences in equation (1) are approximated using the higher order formulations of Roe's scheme, that are based on the work of Osher and Chakravarthy [11].

3.3 Explicit boundary conditions

The boundary conditions are derived using the locally one-dimensional characteristic variable boundary conditions, which yields (for the calculations see for example [10]):

3.3.1 Farfield-Subsonic Inflow

$$\begin{aligned}
P_b &= (1/2)P_a + P_i + \mathrm{sign}(\lambda_k^i)\rho_o c_o[\bar{k}_x(u_a - u_i) + \bar{k}_y(v_a - v_i) + \bar{k}_z(w_a - w_i)] \\
\rho_b &= \rho_a + [(P_b - P_a)/c_o^2] \\
u_b &= u_a + \bar{k}_x[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i) \\
v_b &= v_a + \bar{k}_y[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i) \\
w_b &= w_a + \bar{k}_z[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i)
\end{aligned}$$

Note that these signs correspond to the sign of the first three eigenvalues, and hence this is a means of writing the code for general applications with arbitrary orientation of the computational coordinates. The point a is outside the computational domain, point b is on the computational boundary, and i is inside the computational domain.

3.3.2 Farfield-Subsonic Outflow

$$\begin{aligned}
P_b &= P_a \\
\rho_b &= \rho_a + [(P_b - P_a)/c_o^2] \\
u_b &= u_a + \bar{k}_x[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i) \\
v_b &= v_a + \bar{k}_y[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i) \\
w_b &= w_a + \bar{k}_z[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i)
\end{aligned}$$

3.3.3 Impermeable Surface

$$\begin{aligned}
P_b &= P_r - +\rho_o c_o \\
u_b &= u_r - \bar{k}_x(\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \\
v_b &= v_r - \bar{k}_y(\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r) \\
w_b &= w_r - \bar{k}_z(\bar{k}_x u_r + \bar{k}_y v_r + \bar{k}_z w_r)
\end{aligned}$$

where the point r is the center of the first cell from the boundary and the minus sign in equation (1) is used if r is in the positive k direction from the boundary, and the plus sign is used if r is in the negative direction from the boundary.

3.3.4 Farfield-Supersonic Inflow

In this case all eigenvalues have the same sign. Since we have an inflow case all variables are specified.

3.3.5 Farfield-Supersonic Outflow

In this case also, all eigenvalues have the same sign. But now we have an outflow case, therefore, all variables must be obtained from the solution in the computational domain. All variables are extrapolated from inside the computational domain to the boundary.

3.4 Implicit boundary conditions

In the implicit form, the above boundary conditions can be written in the form of operators formulated as functions of the conservation vector W:

$$FB(W) = 0$$

and are implemented implicitly through:

$$\frac{\partial FB}{\partial W}\delta W = -FB(W).$$

Using these implicit boundary conditions, we show that starting from a small initial CFL number (10), CFL may be adaptively advanced according to:

$$\mathrm{CFL}^{l+1} = \mathrm{CFL}^{l} \cdot \frac{||f(u)||^{l-1}}{||f(u)||^{l}}.$$

This is the key to the successful implementation of Newton-Krylov matrix-free method studied in this paper.

4 Numerical Results

To test the different methodologies developed here we consider a NACA0012 steady transonic airfoil at an angle of attack of 1.25 degrees and a freestream Mach number of 0.8. We consider two meshes, with 2048 and 4096 cells, respectively. We call the first mesh, mesh1, while the second will be denoted by mesh2. In all computations performed herein the solution obtained agrees with the standard one.

The initial code is an EAGLE-derivative code [10] that employs the discretization described in section 3 with explicit boundary conditions, over a body-fitted grid, and which uses a linear solver of an approximate factorization (AF) type (see for example [2]).

We have implemented implicit boundary conditions as described in section 3. We have also replaced the (AF) solver by ILU/GMRES solver. And we have implemented the Newton-Krylov matrix-free methods.

We first compare the defect-correction procedures of (AF) type and ILU/GMRES with explicit boundary conditions on the test case described above. Then, we compare these results with those obtained by replacing explicit boundary conditions by implicit ones. Finally we study the performance of Newton-Krylov matrix-free. All calculations performed here are done on the same Sparc20 machine.

4.1 Defect-Correction procedures: mesh1 case

4.1.1 Explicit boundary conditions

We compare here the results obtained using approximate factorization (AF) method and ILU/GMRES when the boundary conditions are explicit. We observe that to reach the same level of accuracy, the CPU time necessary for AF method is almost double the time necessary to reach the same level of accuracy with the Krylov method (ILU/GMRES) as can be seen in figures 1 and 2, which show, respectively, the iteration count versus the steady residual norm and the CPU time versus the steady residual norm. Moreover, using (AF) method the steady residual norm cannot be dropped to the same final steady residual norm reached by the converged solution obtained using ILU/GMRES. As can be seen below, finer mesh is needed for the steady residual norm to be dropped to the same level as for the Krylov method, when AF is used.

4.1.2 Implicit boundary conditions

We first compare different calculations obtained using different CFL numbers. The results are presented in figures 3 and 4 where we show a comparison of the CPU time versus the steady residual norm. These calculations are performed using a CFL number equal to 6.5, 100, and 500 respectively. From these comparisons we can see that using high CFL number improves the convergence rate. However, when the CFL number is larger than 100, no further improvement can be obtained. This is due to the mismatch of the left and right hand side operators. We will see in the next section that this drawback can be removed using Newton-Krylov methodology described in section 2. We will now validate the CFL strategy described in section 3. In figures 5 and 6 we compare the calculations performed with a CFL number of 100 to the calculations performed with the CFL strategy described in section 3. This comparison shows the validation of using these techniques. It also shows that the converged solution is obtained in about the same CPU time. In figure 7, we show the CFL versus iteration count. Now we show in figures 8 and 9, comparisons of steady state residual norm versus the iteration count and CPU time for the converged solution obtained using explicit boundary conditions with a CFL number equal to 6.5 and using implicit boundary conditions with a CFL of 100. We observe that an improvement in terms of the CPU time is obtained when we use implicit boundary conditions as compared to explicit ones. This comparison highlights the gain obtained using implicit boundary conditions when Krylov methods are used as linear solvers. We should notice that using AF solver, the implicit boundary conditions do not improve the convergence rate because the AF method is based on an approximation of first order to the linear system to be solved.

4.2 Newton-Krylov matrix-free procedures: mesh1 case

We study here the Newton-Krylov matrix-free methodology described in section 2. The techniques used in the choice of the finite differencing parameter are described in section 2. To take full advantage of the power of Newton's method, and thus to allow a more rapid asymptotic convergence to the steady state solution, we use the CFL strategy described in section 3, and validated above. The ILU preconditioner we use here is formed from a lower order discretization and is exactly the same as that used already in the defect-correction procedure studied above. This results in a mixed Jacobian/Preconditioner discretization. More precisely, the explicitly available (Van Leer) first-order flux vector split Jacobian (J_{VL}) is used to precondition the implicitly defined (Roe) higher-order flux difference split Jacobian (J_R) at each implicit time step. In matrix terms, the correction u is obtained as the approximate solution of,

$$(J_{VL})^{-1} J_R u = -(J_{VL})^{-1} f_R.$$

While in the defect-correction context, this correction was obtained as the approximate solution of,

$$J_{VL} u = -f_R.$$

We first compare the results obtained using ILU/GMRES with implicit boundary conditions and with CFL of 100, and using Newton-Krylov matrix-free. The results are presented in figures 10 and 11, in which we show, respectively, the steady residual versus the iteration count and the steady residual versus the CPU time. We perform 4 Newton iterations in each implicit time step. The stopping criterion corresponds to a steady residual norm of 10^{-9}.

In figures 12 and 13 we show a comparison of the four methods studied in this paper. Clearly, we can see that Newton-Krylov matrix-free outperforms all the other three methods.

To refine our analysis we have performed the same study but on a much finer grid. The results are discussed below.

4.3 Defect-Correction procedures: mesh2 case

4.3.1 Explicit boundary conditions

We compare here the results obtained using approximate factorization (AF) method and ILU/GMRES when the boundary conditions are explicit. Similarly to the mesh1 case, we observe that in order to reach the same level of accuracy, the CPU time necessary for AF method is almost double the time necessary with the Krylov method (ILU/GMRES) as can be seen in figures 14 and 15, which show, respectively, the iteration count versus the steady residual norm and the CPU time versus the steady residual norm.

4.3.2 Implicit boundary conditions

As for the mesh1 case, we first compare different calculations obtained using different CFL numbers. The results are presented in figures 16 and 17 where we show a comparison of the steady state residual norm versus CPU time. These calculations are performed using a CFL number equal respectively to 5, 100, and 500. These comparisons show again that using high CFL number improves the convergence rate. However, when the CFL number is larger than 100, no further improvement can be obtained. This is due to the mismatch of the left and right hand side operators. We will see in the next section that this drawback can be removed using Newton-Krylov methodology described in section 2. We will now, validate the CFL strategy described in section 3. In figures 18 and 19 we compare the calculations performed with CFL 100 to the calculations performed with the CFL strategy described in section 3. This comparison shows again the validation of using these techniques. It also shows that the converged solution is obtained in about the same CPU time. In figure 20, we show the CFL versus iteration count. Now we show in figures 21 and 22, comparisons of steady state residual norm versus the iteration count and CPU time for the converged solution obtained using explicit boundary conditions with a CFL number equal to 5 and using implicit boundary conditions with a CFL number equal to 100. We observe that an improvement in terms of the CPU time is obtained when we use implicit boundary conditions as compared to explicit ones. This comparison highlights again the gain obtained using implicit boundary conditions when Krylov methods are used as linear solvers.

4.4 Newton-Krylov matrix-free procedures: mesh2 case

Here, we study here again the Newton-Krylov matrix-free methodology described in section 2. The techniques used in the choice of the finite differencing parameter are described in section 2. To take full advantage of the power of Newton's method, and thus to allow a more rapid asymptotic convergence to the steady state solution, we use the CFL strategy described in section 3, and validated above. The ILU preconditioner we use here is formed from a lower order discretization and is exactly the same as that used already in the defect-correction procedure studied above. This results in a mixed Jacobian/Preconditioner discretization. More precisely, the explicitly available (Van Leer) first-order flux vector split Jacobian (J_{VL}) is used to precondition the implicitly defined (Roe) higher-order flux difference split Jacobian (J_R) at each implicit time step. In matrix terms, the correction u is obtained as the approximate solution of,

$$(J_{VL})^{-1} J_R u = -(J_{VL})^{-1} f_R.$$

While in the defect-correction context, this correction was obtained as the approximate solution of,

$$J_{VL} u = -f_R.$$

We first compare the results obtained using ILU/GMRES with implicit boundary conditions and with CFL of 100, and using Newton-Krylov matrix-free. The results are presented in figures 23 and 24, in which we show, respectively, the steady residual versus the iteration count and the steady residual versus the CPU time. We perform 4 Newton iterations in each implicit time step. The stopping criterion corresponds to a steady residual norm of 10^{-9}.

In figures 25 and 26 we show a comparison of the four methods studied in this paper. The comparison highlights the efficiency and performance of the Newton-Krylov matrix-free method.

5 Conclusions

In this paper we have demonstrated the performance of Krylov based methods. The convergence rate is improved using implicit boundary conditions as compared to explicit ones. The higher-order Jacobian needed in the Newton's method need not be computed explicitly in the Newton-Krylov matrix-free context. The use of mixed discretization (higher-order) Jacobian / (lower-order) preconditioner, results in an efficient preconditioned Newton-Krylov matrix-free algorithm in terms of CPU time as has been shown in the numerical experiments presented in this study.

References

[1] T. J. Barth, *Analysis of Implicit Local Linearization Techniques for Upwind and TVD Algorithms*, AIAA Paper No. 87-0595, January 1987.

[2] R. M. Beam, and R. F. Warming, *An Implicit Factored Scheme for the Compressible Navier-Stokes Equations*, AIAA Journal, Vol. 16, No. 4, April 1978, pp. 393–402.

[3] P. N. Brown and Y. Saad, *Hybrid Krylov Methods for Nonlinear Systems of Equations*, SIAM J. Sci. Stat. Comp. **11**(1990), 450–481.

[4] X.-C. Cai, W. D. Gropp, D. E. Keyes and M. D. Tidriri, "Parallel implicit methods for aerodynamics," Seventh International Conference on Domain Decomposition Methods for Partial Differential Equations, D. Keyes, J. Xu, eds., AMS, 1994.

[5] X.-C. Cai, W. D. Gropp, D. E. Keyes and M. D. Tidriri, "Newton-Krylov-Schwarz Methods in CFD," Proceedings of the International Workshop on the Navier-Stokes Equations, Notes in Numerical Fluid Mechanics, R. Rannacher, eds. Vieweg Verlag, Braunschweig, 1994.

[6] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[7] Z. Johan, T. J.R. Hugues, K. K. Mathur, and S. L. Johnsson, *A data parallel finite element method for computational fluid dynamics on the Connection Machine system*, Computer Methods in Applied Mechanics and Engineering, Vol. 99 (1992), pp. 113–124.

[8] M.-S. Liou, and B. Van Leer, *Choice of Implicit and Explicit Operators for the Upwind Differencing Method*, AIAA Paper, AIAA-88-0624 (1988).

[9] P. R. McHugh and D. A. Knoll, *Inexact Newton's Method Solutions to the Incompressible Navier-Stokes and Energy Equations Using Standard and Matrix-Free Implementations*, AIAA Paper, 1993.

[10] J. S. Mounts, D. M. Belk and D. L. Whitfield, *Program EAGLE User's Manual, Vol. IV – Multiblock Implicit, Steady-state Euler Code*, Air Force Armament Laboratory TR-88-117, Vol. IV, September 1988.

[11] S. Osher, and S. R. Chakravarthy, *Very High Order Accurate TVD Schemes*, ICASE Report No. 84-44, September 1984.

[12] P. L. Roe, *Approximate Riemann Solvers, Parameter Vector, and Difference Schemes*, J. Comp. Phys. **43**(1981), 357–372.

[13] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comp. **7**(1986), 865–869.

[14] J. L. Steger and R. F. Warming, *Flux Vector Splitting of the Inviscid Gasdynamics Equations with Applications to Finite-Difference Methods*, J. Comp. Phys. **40**(1981), 263–293.

[15] W. T. Thomkins, Jr. and R. H. Bush, *Boundary Treatments for Implicit Solutions to Euler and Navier-Stokes Equations*, J. Comp. Phys. **48.** (1982), 302–311.

[16] M. D. Tidriri, *Coupling of different models and different approximations in the coputation of external flows*, PhD thesis, Univ. of Paris XI, 1992.

[17] M. D. Tidriri, *Domain Decompositions for Compressible Navier-Stokes Equations*, J. Comp. Phys. July, 1995.

[18] V. Venkatakrishnan, *Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters*, To appear in J. Comp. Phys.

[19] H. C. Yee, R. M. Beam, and R. F. Warming *Boundary Approximations for Implicit Schemes for One-Dimensional Inviscid Equations of Gasdynamics*, AIAA Journal, Vol.20, NO.9, September 1982.

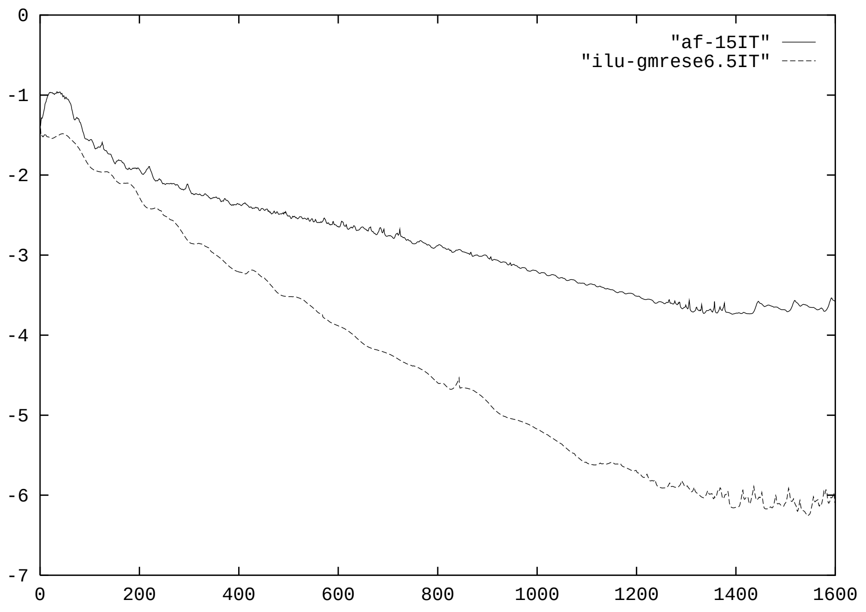

Figure 1: Steady-state residual versus iteration count for approximate factorization (AF) and ILU/GMRES solvers.

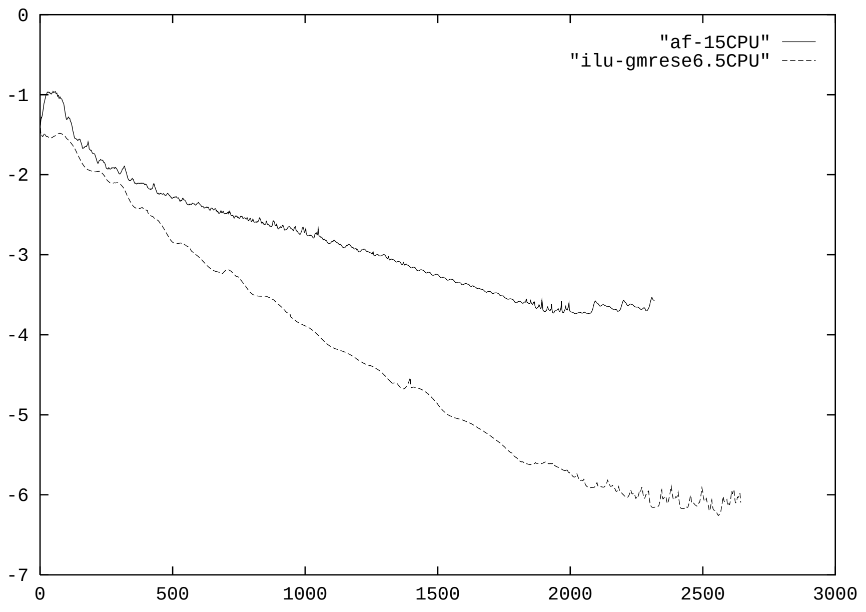

Figure 2: Steady-state residual versus CPU time for approximate factorization (AF) and ILU/GMRES solvers.

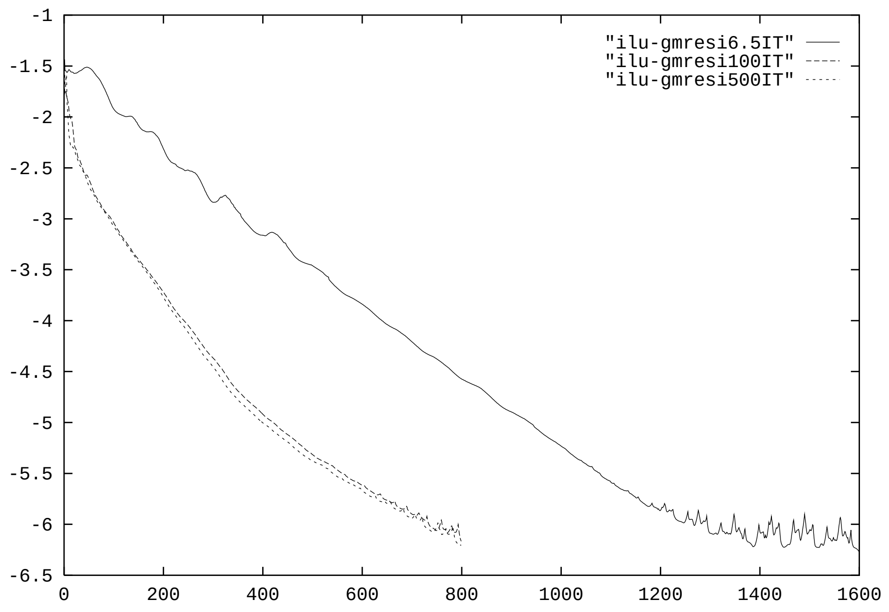

Figure 3: Steady-state residual versus iteration count for ILU/GMRES solver with different CFL: 6.5, 100, and 500.

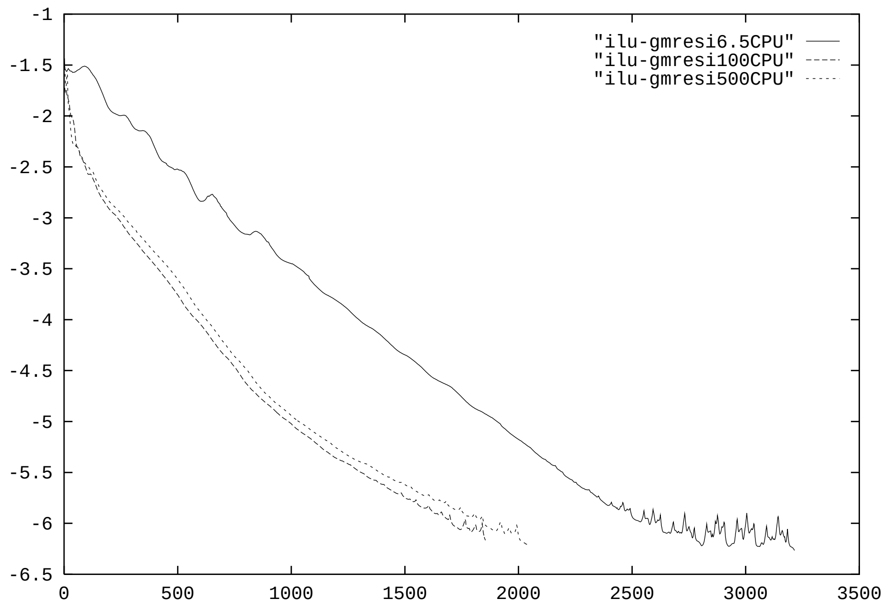

Figure 4: Steady-state residual versus CPU time for ILU/GMRES solver with different CFL: 6.5, 100, and 500.

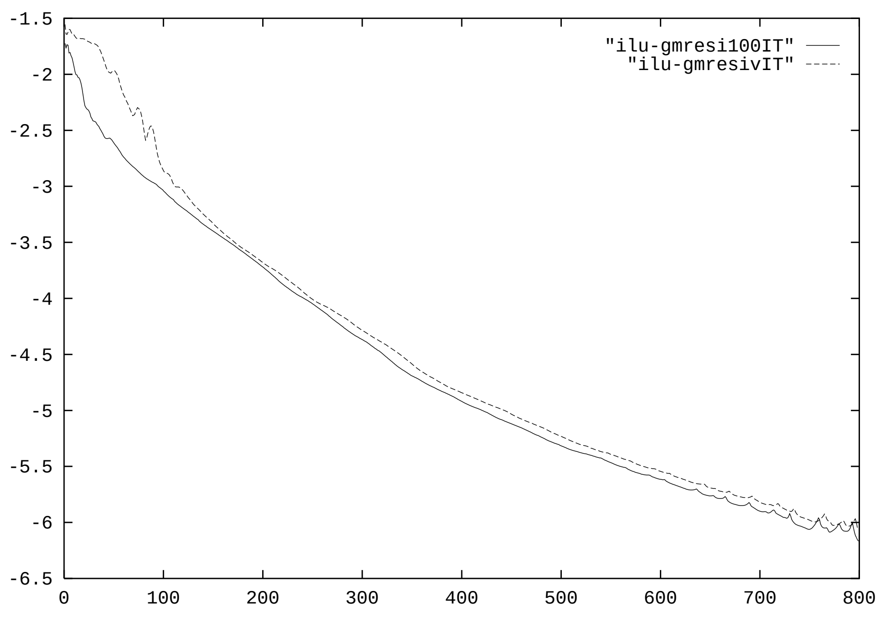

Figure 5: Steady-state residual versus iteration count for ILU/GMRES solvers with CFL constant equal 100, and with adaptively increasing CFL.

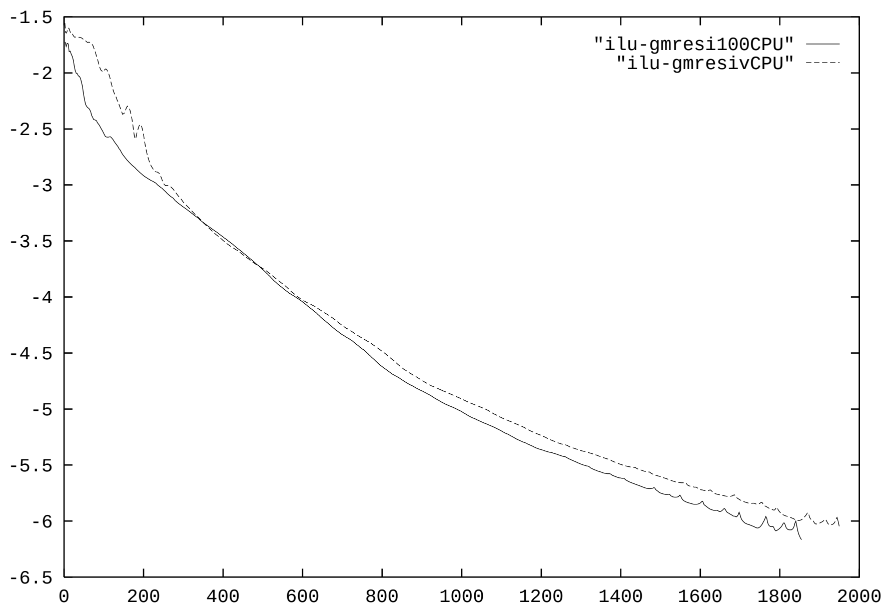

Figure 6: Steady-state residual versus CPU time for ILU/GMRES solvers with CFL constant equal 100, and with adaptively increasing CFL.

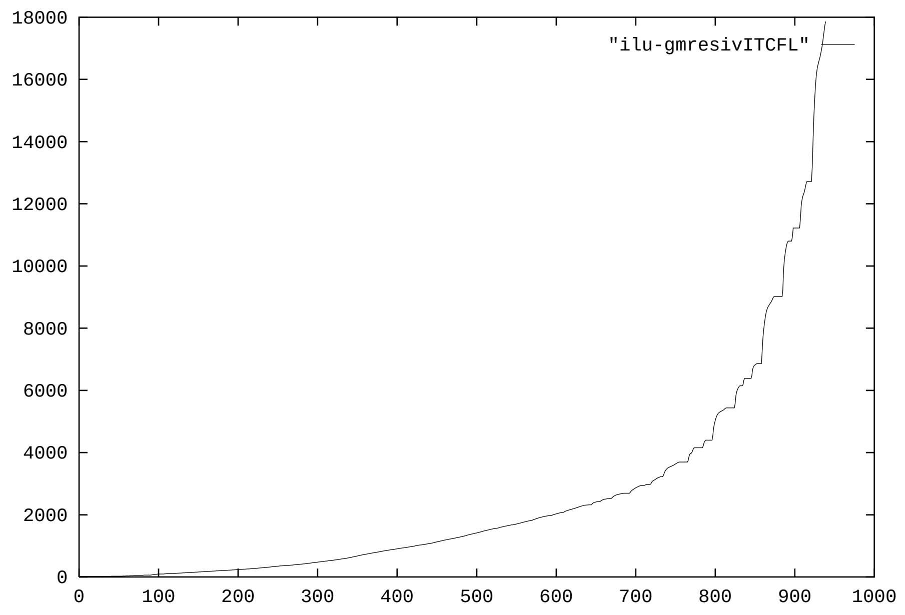

Figure 7: CFL versus iteration count for ILU/GMRES solvers.

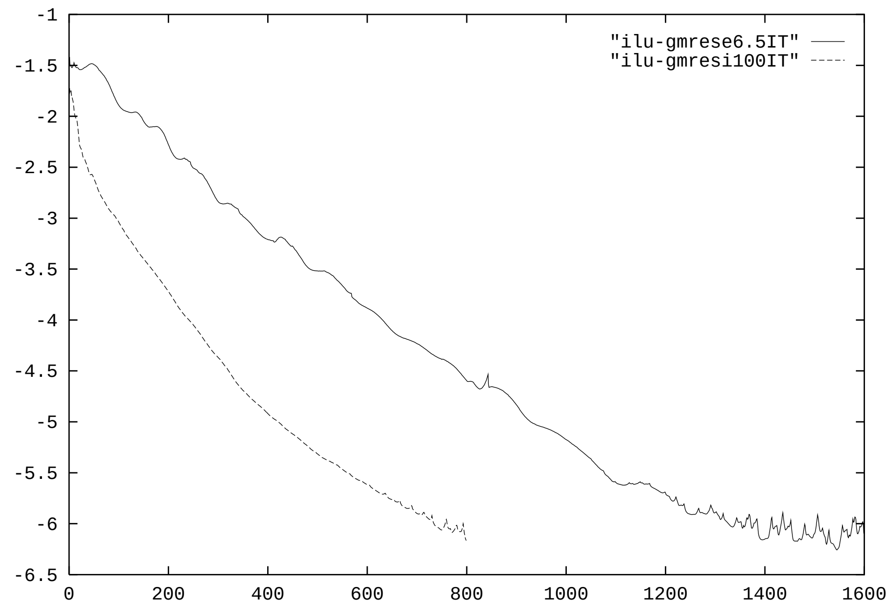

Figure 8: Steady state residual versus iteration count for ILU/GMRES solvers with explicit boundary conditions and implicit boundary conditions.

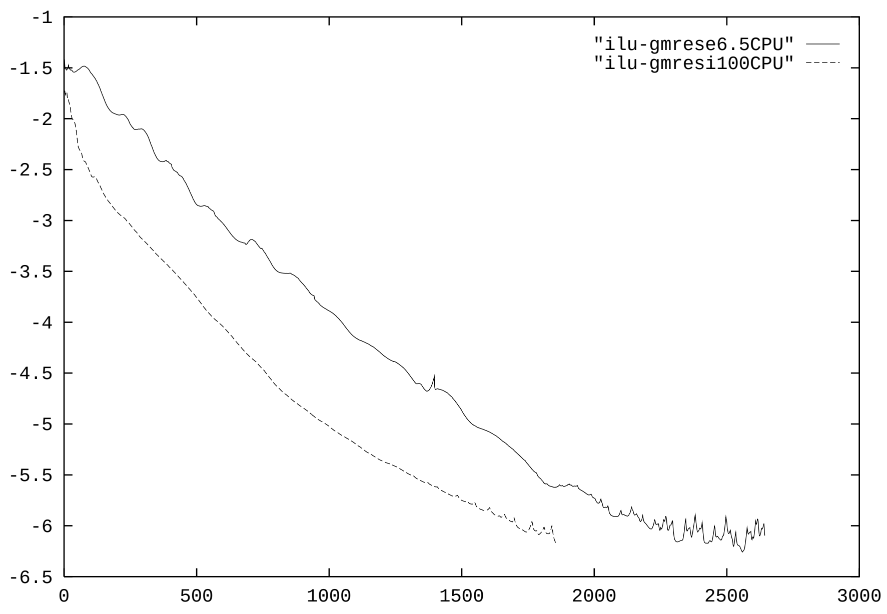

Figure 9: Steady state residual versus CPU time for ILU/GMRES solvers with explicit boundary conditions and implicit boundary conditions.

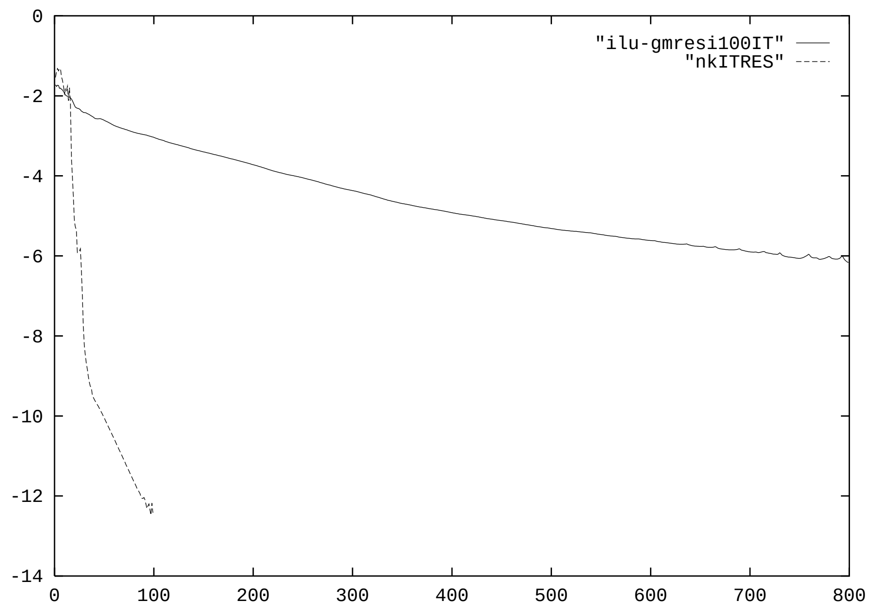

Figure 10: Steady-state residual versus iteration count for defect correction (ILU/GMRES) and Newton-Krylov matrix-free solvers.

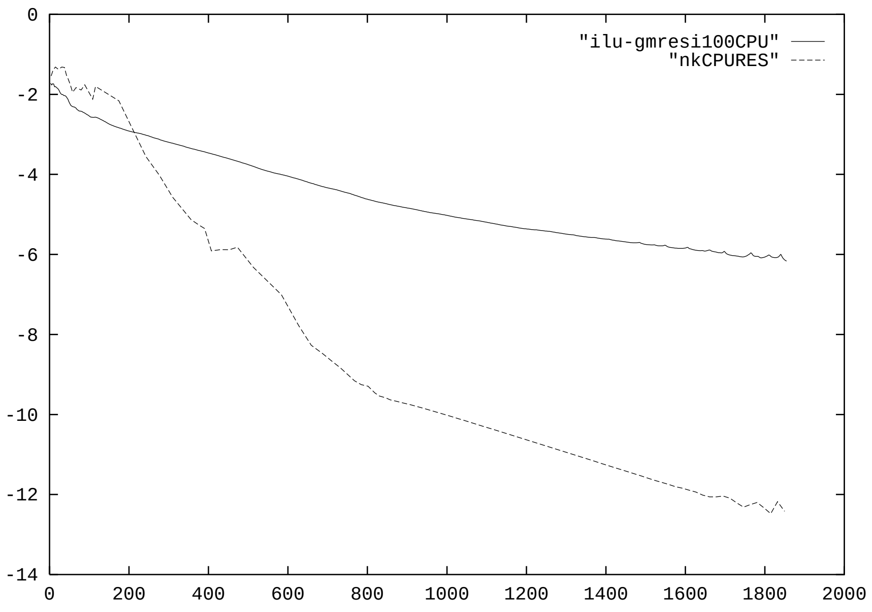

Figure 11: Steady-state residual versus CPU time for defect correction (ILU/GMRES) and Newton-Krylov matrix-free solvers.

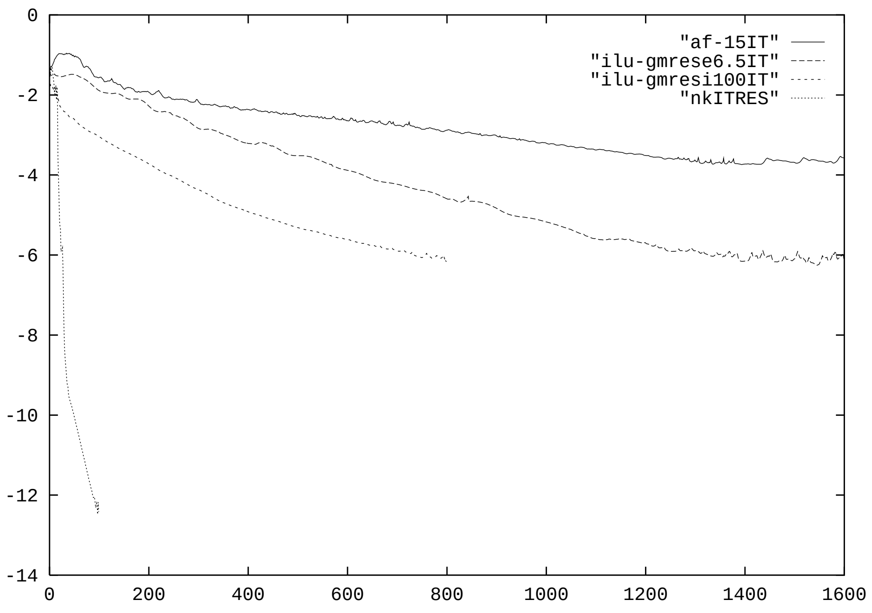

Figure 12: Steady-state residual versus iteration count for approximate factorization (AF), ILU/GMRES with explict boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton-Krylov matrix-free solvers.

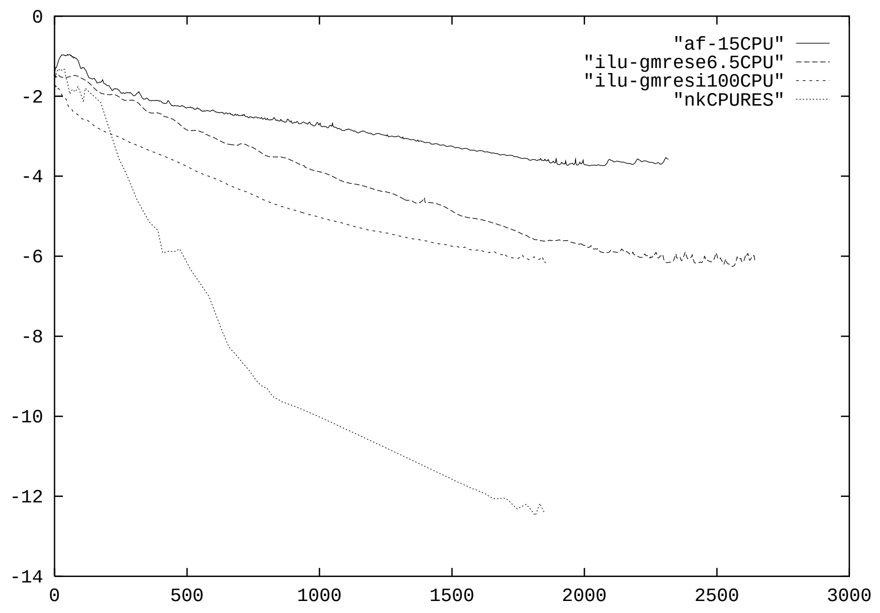

Figure 13: Steady-state residual versus CPU time for approximate factorization (AF), ILU/GMRES with explict boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton-Krylov matrix-free solvers.

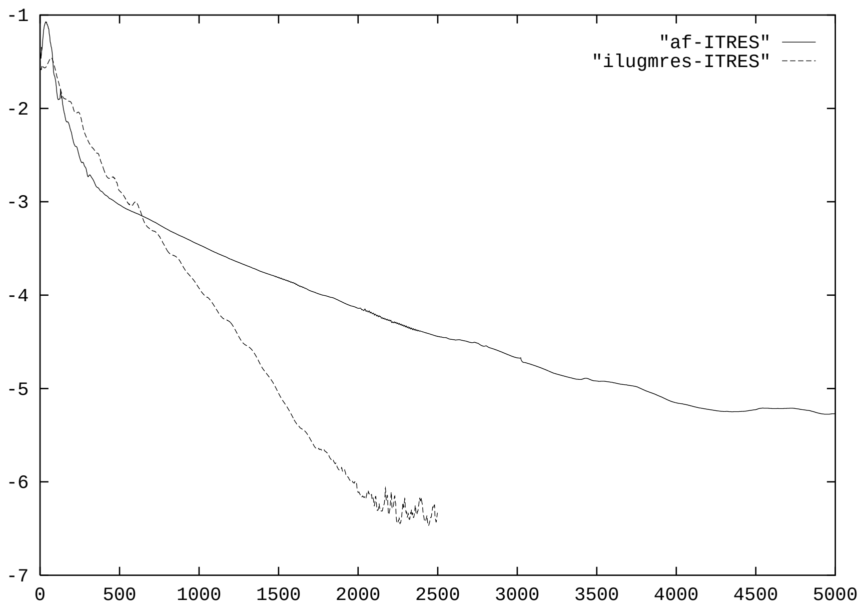

Figure 14: Steady-state residual versus iteration count for approximate factorization (AF) and ILU/GMRES solvers.

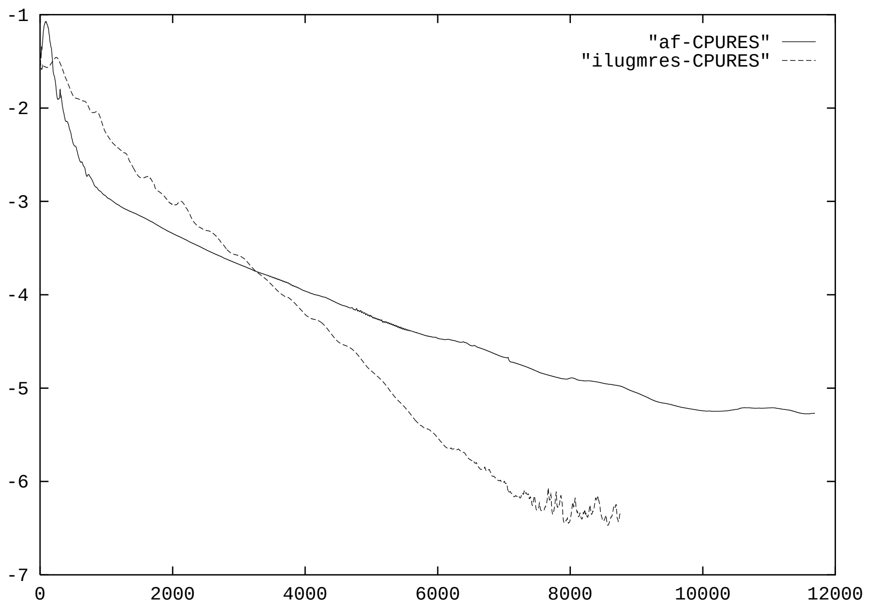

Figure 15: Steady-state residual versus CPU time for approximate factorization (AF) and ILU/GMRES solvers.

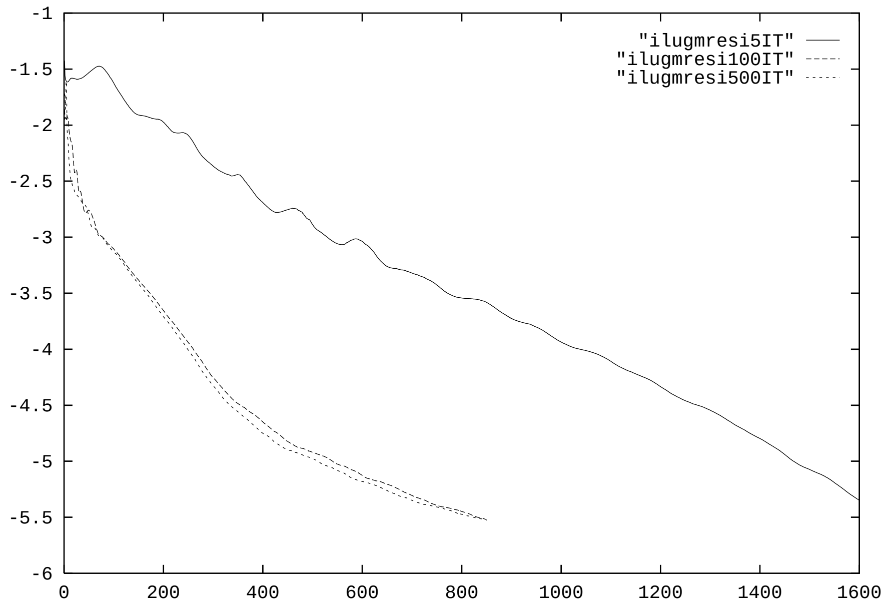

Figure 16: Steady-state residual versus iteration count for ILU/GMRES solver with different CFL: 5, 100, and 500.

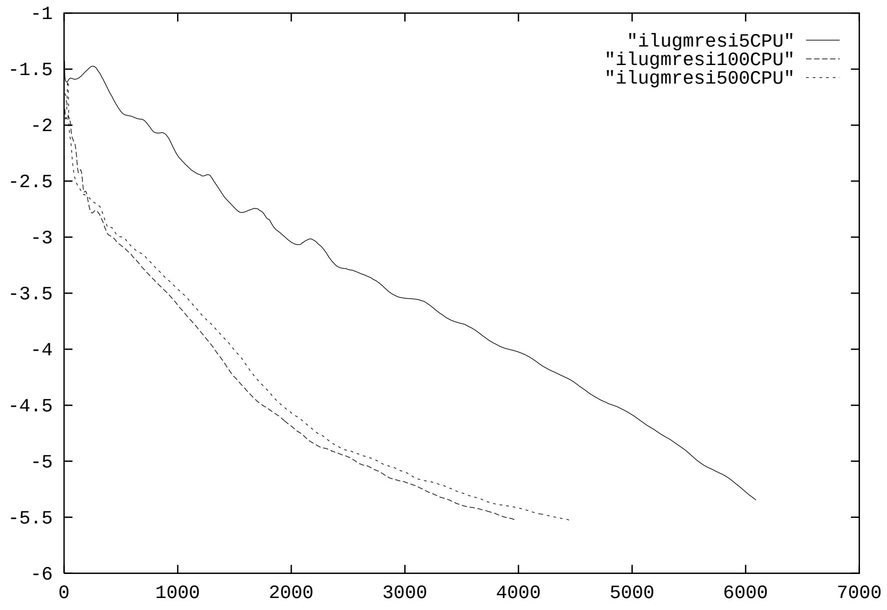

Figure 17: Steady-state residual versus CPU time for ILU/GMRES solver with different CFL: 5, 100, and 500.

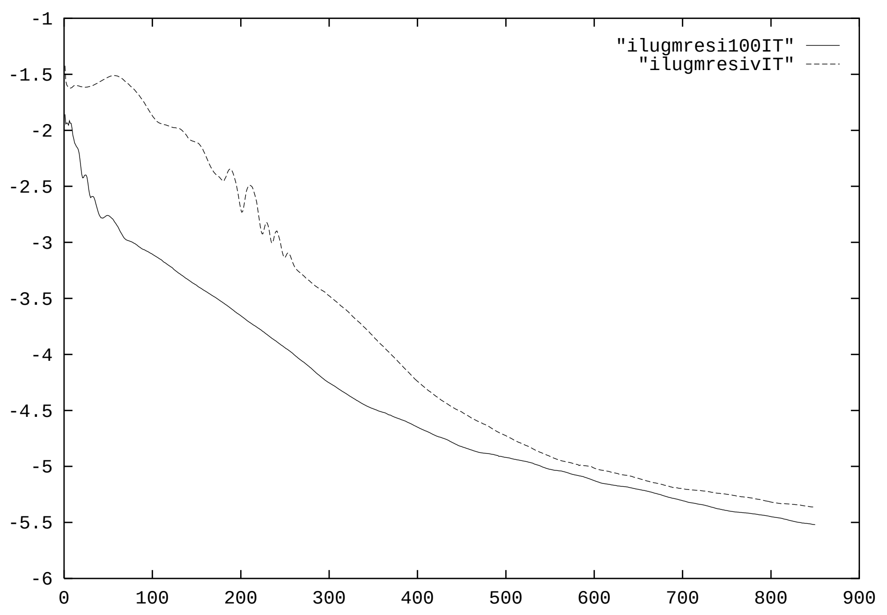

Figure 18: Steady-state residual versus iteration count for ILU/GMRES solvers with CFL constant equal 100, and with adaptively increasing CFL.

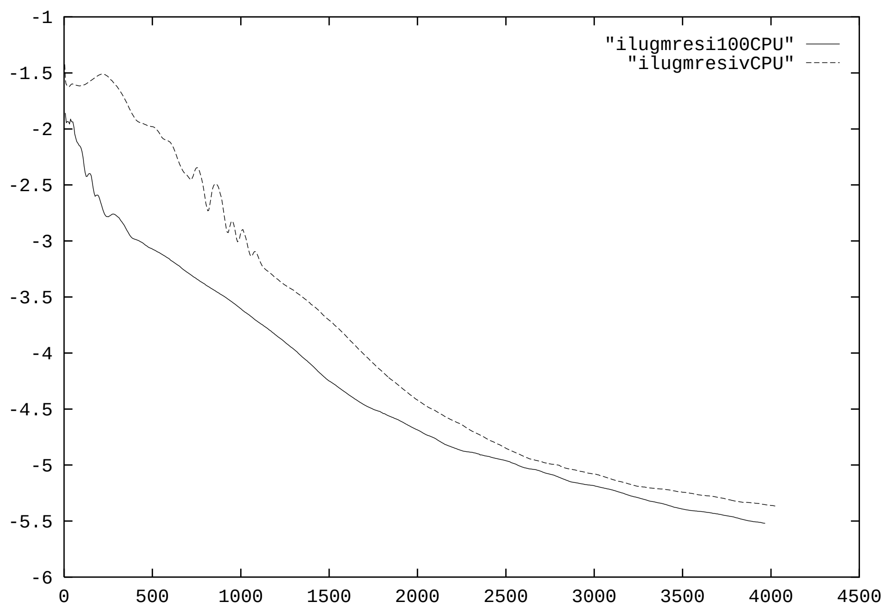

Figure 19: Steady-state residual versus CPU time for ILU/GMRES solvers with CFL constant equal 100, and with adaptively increasing CFL.

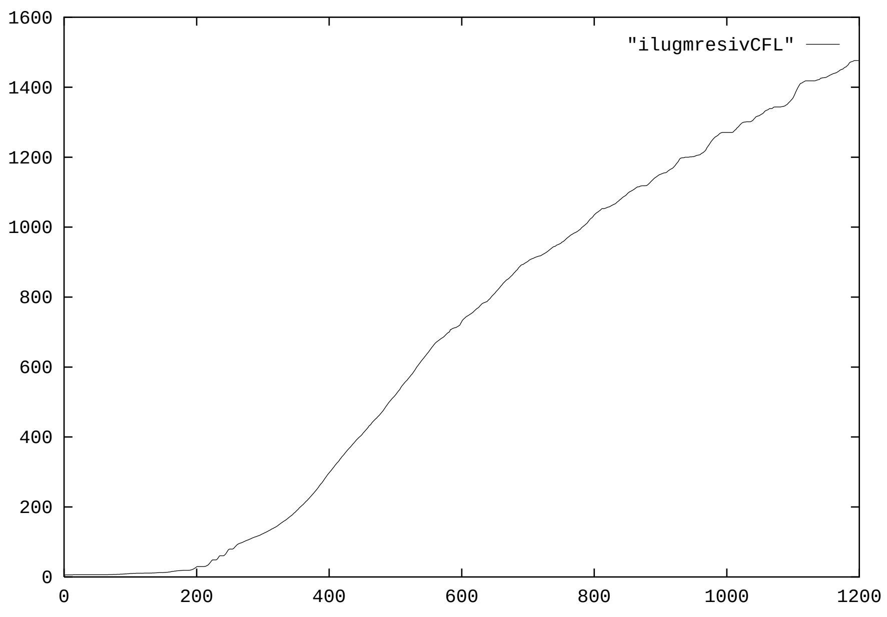

Figure 20: CFL versus iteration count for ILU/GMRES solvers.

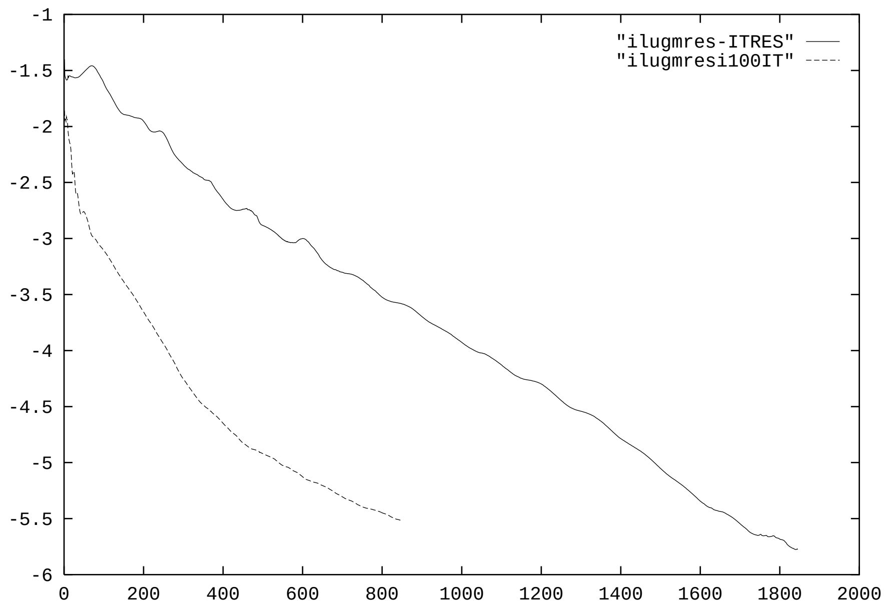

Figure 21: Steady state residual versus iteration count for ILU/GMRES solvers with explicit boundary conditions and implicit boundary conditions.

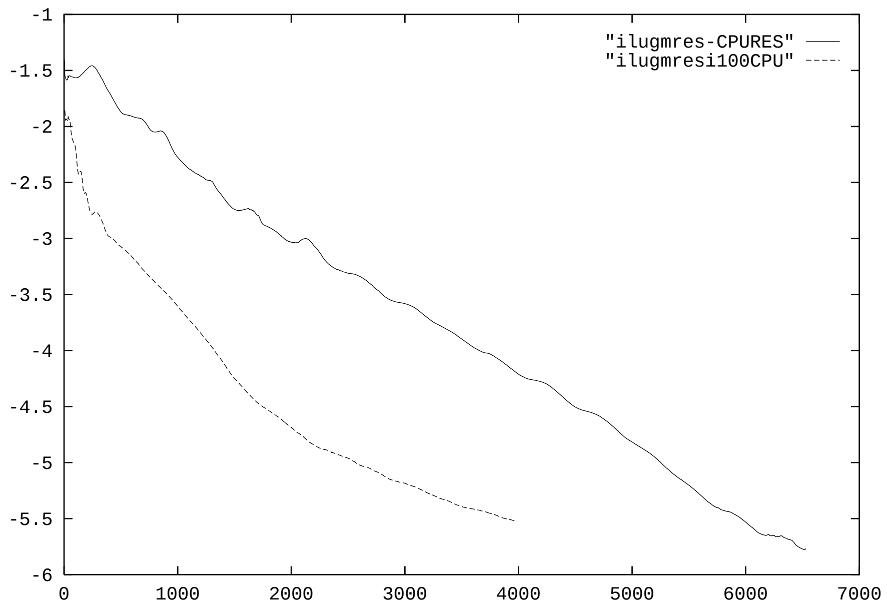

Figure 22: Steady state residual versus CPU time for ILU/GMRES solvers with explicit boundary conditions and implicit boundary conditions.

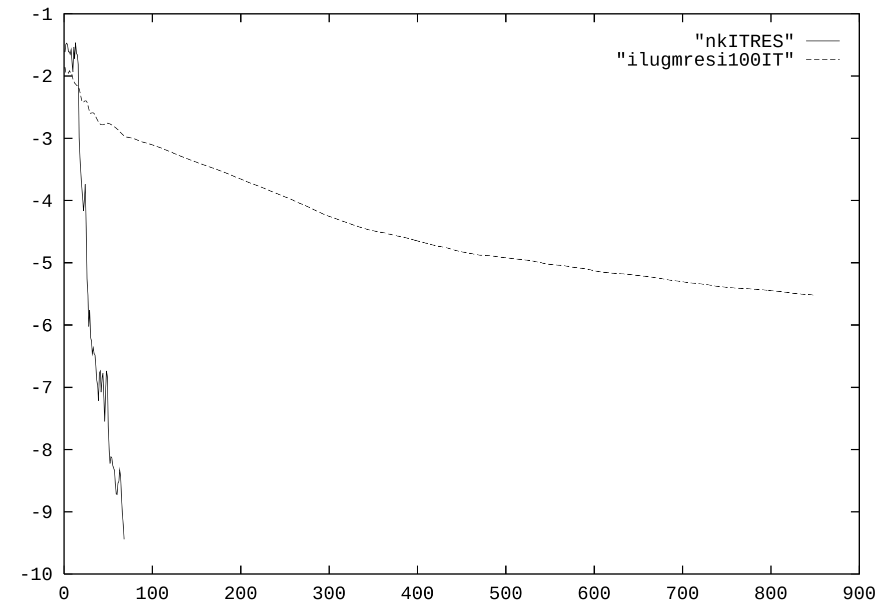

Figure 23: Steady-state residual versus iteration count for defect correction (ILU/GMRES) and Newton-Krylov matrix-free solvers.

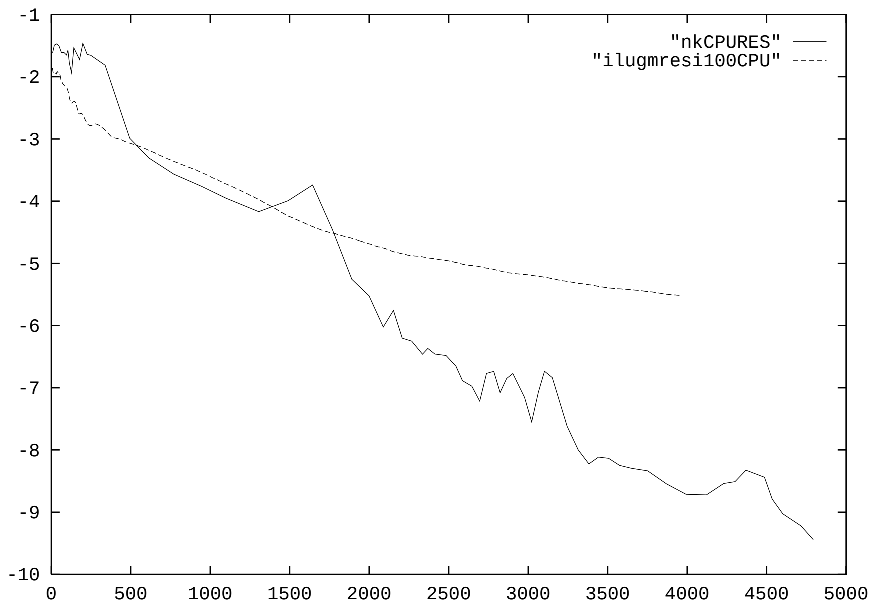

Figure 24: Steady-state residual versus CPU time for defect correction (ILU/GMRES) and Newton-Krylov matrix-free solvers.

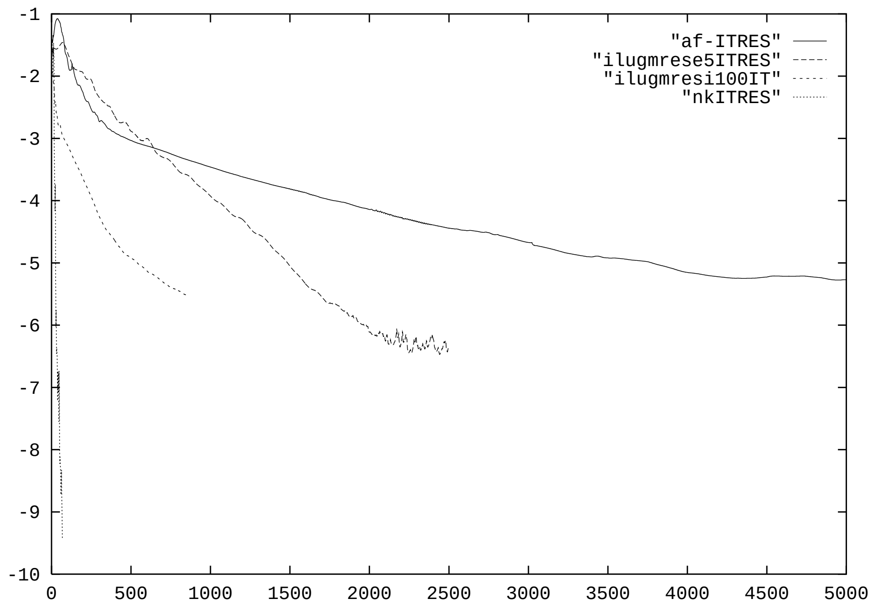

Figure 25: Steady-state residual versus iteration count for approximate factorization (AF), ILU/GMRES with explict boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton-Krylov matrix-free solvers.

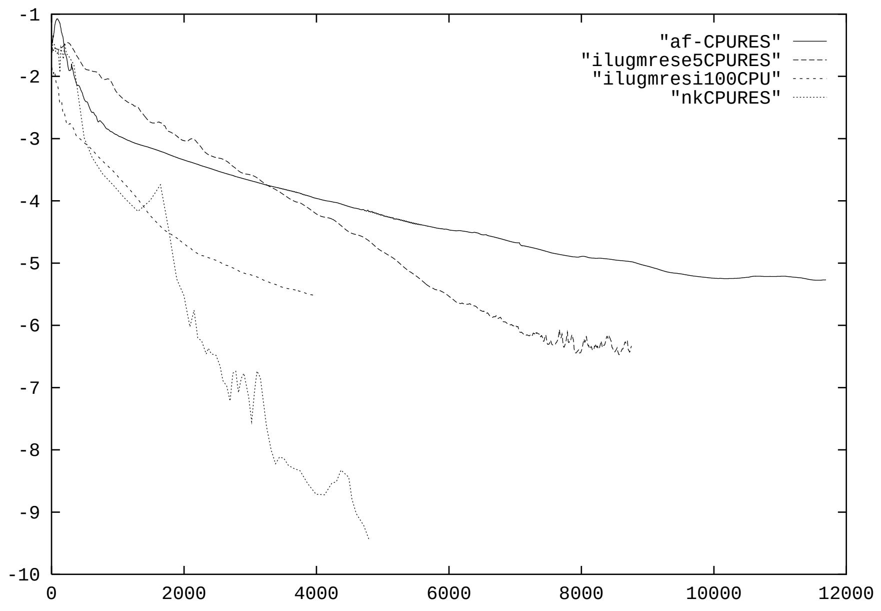

Figure 26: Steady-state residual versus CPU time for approximate factorization (AF), ILU/GMRES with explict boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton-Krylov matrix-free solvers.