Schwarz-based algorithms for compressible flows

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

Abstract

We investigate in this paper the application of Schwarz-based algorithms to compressible flows. First, we study the combination of these methods with defect-correction procedures. We then study the effect on the Schwarz-based methods of replacing the explicit treatment of the boundary conditions by an implicit one. In the last part of this paper we study the combination of these methods with Newton-Krylov matrix-free methods. Numerical experiments that show the performance of our approaches are then presented.

*ICASE, MS 132C, NASA-LaRC, Hampton, VA 23681-0001 *email:*tidriri@icase.edu. 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.

1 Introduction

To compute steady compressible flows one often uses an implicit discretization approach, which leads to a large sparse linear system that must 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 with a lower order approximation than that used 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 schemes 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. However, for unstructured grids, such techniques are no longer valid, and the system is solved using direct or iterative techniques. 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 implemented 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 a 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 [25], [21], and [13]). Although they clearly demonstrate that high CFL numbers are possible, the reduction in CPU time is not clear cut.

The investigation of defect-correction procedures based on Krylov methods, together with implicit treatment of the boundary conditions has been done by the author in [24]. In [24] the author has also studied Newton-Krylov matrix-free (see also [3], [22], [23], and [10]) methods combined with mixed discretization in the implicitly defined Jacobian Preconditioner. The preconditioner based on incomplete factorizations studied in [24] is difficult to parallelize efficiently. The focus in this work is on the developement of algorithms that are suitable for the parallel computing environment. In this case, domain decomposition methods that allow the reduction of the global solution of a given problem to the solutions of local subproblems are preferred. We propose, therefore, to combine these methods with the preconditioned Newton-Krylov matrix-free methods developed in [24].

One of the domain decomposition algorithms that has potential applications on parallel computers is the additive Schwarz algorithm [8]. The other Schwarz-based method; the multiplicative Schwarz method [8] can also be used in the parallel environment by using a multi-coloring process. The proposed algorithm is, therefore, to combine the Newton-Krylov matrix-free methods with the Schwarz-based methods. The combination of Newton-Krylov matrix-free with domain decomposition methods was first introduced by the author in [22] and [23]. More precisely, the author has combined the Newton-Krylov matrix-free method with the Domain Decomposition Time

Marching Algorithm that was introduced by Le Tallec and Tidriri in [11] (see also [22] and [23]).

In the next section, we describe the Euler solver. In section 3, we describe the methodology studied in this paper. In section 4, a comprehensive study of Schwarz-based methods combined with defect-correction procedures with explicit and implicit boundary conditions is performed. We then study the combination of the Schwarz-based methods with the Newton-Krylov matrix-free methods. The last section is devoted to some conclusions and extensions.

2 Description of the Euler solver

2.1 Governing Equations

The bidimensional Euler Equations in conservative form writes

$$W_t + F(W)_x + G(W)_y = 0, \tag{1}$$

where $W = (\rho, \rho u, \rho v, e)^T$, $F = (\rho u, \rho u^2 + p, \rho uv, u(e+p))^T$, and $G = (\rho u, \rho uv, \rho v^2 + p, v(e+p))^T$.
Above ρ is the density, u, v are the velocity components, e is the internal energy, p is the pressure defined by $p = (\gamma - 1)(e - (\rho(u^2 + v^2)/2))$, and finally, γ is a constant with $\gamma \approx 1.4$ for air.

After changing the variables into the curvilinear coordinate

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

we obtain the following set of equations

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

where $\tilde{W}$ 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

$$\begin{aligned} \tilde{W} &= J^{-1} W \\ \tilde{F} &= J^{-1} \left(\xi_t W + \xi_x F + \xi_y G \right) \\ \tilde{G} &= J^{-1} \left(\eta_t W + \eta_x F + \eta_y G \right), \end{aligned}$$

and

$$
\begin{aligned}
J &= \frac{\partial(\xi,\eta,\tau)}{\partial(x,y,t)} \\
&= \det\begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix}
\end{aligned}
$$

From now on, the tilde in the expressions of $\tilde{W}$, $\tilde{F}$, and $\tilde{G}$ will be omitted.

2.2 Finite volume scheme

An implicit finite volume discretization of equation (2) can be written as

$$
\begin{aligned}
(W_{i,j}^{n+1} - W_{i,j}^{n})\Delta\xi\Delta\eta + (F_{i+\frac{1}{2},j}^{n+1} - F_{i-\frac{1}{2},j}^{n+1})\Delta\eta\Delta\tau \\
+(G_{i,j+\frac{1}{2}}^{n+1} - G_{i,j-\frac{1}{2}}^{n+1})\Delta\xi\Delta\tau = 0,
\end{aligned} \tag{3}
$$

where the values are taken at the center of either the cell (i,j) or the interfaces of the cell (i,j) and its neighbours. To compute the fluxes above, we shall use a flux splitting approach, which is defined for F by (see [20])

$$
F = F^{+} + F^{-},
$$

with similar expressions for G. F^{+} is associated with the positive eigenvalues whereas F^{-} is associated with the negative ones, and G^{+}, $G^{-}$are defined analogously.
Let $\delta W = W_{i,j}^{n+1} - W_{i,j}^{n}$, then the implicit split-flux discretization of (3) is given by

$$
\delta W^{n} + \Delta\tau(\delta_{\xi}(F^{+} + F^{-})^{n+1} + \delta_{\eta}(G^{+} + G^{-})^{n+1}) = 0,
$$

where δ_{ξ} is defined by

$$
\delta_{\xi}F = \frac{1}{\Delta\xi}[F_{i+1/2,j} - F_{i-1/2,j}] \tag{4}
$$

and δ_{η} is defined similarly. This yields the following nonlinear system

$$
f(W^{n+1}) = 0. \tag{5}
$$

This nonlinear system will be solved by using the proposed approach of this paper, which is based on a Newton-Krylov method (see next sections). Now, we shall describe the more standard defect-correction method, which is based on the following linearization of first order in time of the nonlinear system above

$$[I + \Delta\tau(\delta^i_\xi A^+\cdot + \delta^i_\xi A^-\cdot + \delta^i_\eta B^+\cdot + \delta^i_\eta B^-\cdot)]\delta W^n$$
$$= -\Delta\tau(\delta^e_\xi F^n + \delta^e_\eta G^n).$$

The superscripts i and e above indicate that the implicit and explicit operators are discretized using different schemes. The dots indicate that the difference operators apply to the product of the Jacobian matrices with δW^n. The matrices $A^+, A^-, B^+,$ and B^- are defined by

$$A^+ = \frac{\partial F^+}{\partial W}, \qquad A^- = \frac{\partial F^-}{\partial W},$$
$$B^+ = \frac{\partial G^+}{\partial W}, \qquad B^- = \frac{\partial G^-}{\partial W}.$$

The compact form of the above equation corresponds to the following defect-correction procedure

$$\mathbf{A}\delta W^n = b. \tag{6}$$

The different fluxes above are computed using the Roe's approximate Riemann solver [17]. Three limiters are employed: minmod, Superbee, and Van Leer. The Jacobians are evaluated using first-order Roe's scheme, or the first-order flux-vector split scheme [20], which corresponds to the true partials of the positive and negative flux vectors as described earlier. However, in the context of defect-correction method the flux-vector split scheme has been shown to give improved convergence rates over the Roe matrices. Therefore, for the defect-correction approach the Jacobian matrices corresponding to the flux-vector split scheme are used in the left-hand side. This results in an inconsistent left and right-hand side operators.

Remark 2.1 *For most CFD codes, the implicit spatial differences are only first-order accurate. The higher-order matrix representation is difficult to obtain, even if it is possible the resulting matrix is very large, requires a lot of storage, large operation count in its evaluation, and may be very difficult to invert.*

Following this remark, the implicit spatial differences (the left-hand side) in equation (6) are approximated, only, through a first-order accurate scheme. The explicit spatial differences (right-hand side) in equation (6) are approximated using the higher-order formulations of Roe's scheme, that are based on the work of Osher and Chakravarthy [16].

2.3 Explicit boundary conditions

The boundary conditions are derived using the locally one-dimensional characteristic variable boundary conditions, which yields (for the derivations see for example [15]):

2.3.1 Farfield-Subsonic Inflow

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

Above, the point a is outside the computational domain, point b is on the computational boundary, and i is inside the computational domain.

2.3.2 Farfield-Subsonic Outflow

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

2.3.3 Impermeable Surface

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

Where the point r is the center of the first cell from the boundary and the minus sign in equation (2) 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.

2.3.4 Farfield-Supersonic Inflow

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

2.3.5 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.

2.4 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:

$$f_b(W) = 0 \tag{7}$$

and are implemented implicitly through:

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

Using these implicit boundary conditions the author showed in [24], that starting from a small initial CFL number (10), CFL may be adaptively advanced according to:

$$\mathrm{CFL}^{n+1} = \mathrm{CFL}^{n} \cdot \frac{||f(W)||^{n-1}}{||f(W)||^{n}},$$

where the uperscripts denote the iteration in time. This is the key to the successful implementation of the preconditioned Newton-Krylov matrix-free method studied in [24], and which we combine here with the Schwarz-based methods.

3 Description of the methodology

Newton-Krylov methods first proposed by Brown and Saad [3], have been investigated for compressible Euler and Navier-Stokes equations using unstructured grids in [22], [23], and [10], and for structured grids in [4], and [5], and [24].

In [22] and [23], the author has studied both transonic and supersonic compressible Navier-Stokes flows. In [4], [5], and [24] a study of a convection-diffusion model problem, the full potential flows and the transonic compressible Euler flows have been performed. implicitly defined Jacobian preconditioner.

The most effective preconditioner, ILU, is difficult to parallelize efficiently. On the other hand domain decomposition methods appear to be effective for the parallel solution of large systems of linear or nonlinear algebraic equations resulting from the application of finite element methods or finite difference methods to fluid dynamics problems. The alternating method introduced by H. A. Schwarz in 1890 [19] appears to be the earliest domain decomposition method. For two subdomains this algorithm is intrinsically sequential. Its extension to include the case of many subdomains was done by P. L. Lions [12]. As a consequence of this work, the additive Schwarz methods were developed. Another method, which is a direct generalization of the original alternating method is the multiplicative algorithm. These methods reduce the solution of the global problem on the global domain to the solution of subproblems on local subdomains, obtained by considering an overlapping subdivision of the global domain.

Most of the theory and applications of the Schwarz-based methods have been primarily performed for elliptic and parabolic boundary value problems discretized using finite element methods. In this paper we shall focus on their applications to the hyperbolic problems. We shall also study their combination with the Newton-Krylov matrix-free methods studied in [24].

3.1 Newton's Method

Consider the following nonlinear system of equations

$$f(W) = 0, \tag{8}$$

where f is a nonlinear function from $\mathbb{R}^2$ to $\mathbb{R}^2$. Newton's method applied to (8) results in the following iteration

- Define u_0, an initial guess
- For $k = 0, 1, 2, ...$ until convergence do

$$\text{Solve} \quad J(W_k)\delta W_k = -f(W_k), \tag{9}$$

$$\text{Set} \quad W_{k+1} = W_k + \delta W_k, \tag{10}$$

where $J(W_k) = \dfrac{\partial f}{\partial W}(W_k)$ is the sytem Jacobian.

For the compressible Euler case (see section 2) this Jacobian corresponds to a higher- order matrix-representation. Using direct-methods to solve the system (9), the memory requirements and the computational complexity are prohibitive. In this case iterative methods are preferred and the system (9) is solved only approximately. The resulting method is called the inexact newton method [6], and corresponds to the following iteration

- Define u_0, an initial guess
- For $k = 0, 1, 2, \cdots$ until convergence do

$$\text{Solve} \quad J(W_k)\delta W_k = -f(W_k), \tag{11}$$

$$\text{Set} \quad W_{k+1} = W_k + \alpha\delta W_k, \tag{12}$$

where $J(W_k) = \dfrac{\partial f}{\partial W}(W_k)$ denotes the sytem Jacobian as before, and α is a parameter selected using a line search or trust region method ([3] and [7]).

3.2 Krylov methods

The iterative methods we will use to solve the linear system (11) which we rewrite as

$$J\delta w = -f, \tag{13}$$

where f and its Jacobian J are evaluated at the current iterate, are the Krylov method. If w_0 is an initial guess for the true solution of (13), then letting $w = w_0 + Z$, we have the equivalent system

$$JZ = r^0,$$

where $r^0 = -f - Jw_0$ is the initial residual. Let K_m be the Krylov subspace

$$K_m := \text{Span}\{r^0, Jr^0, \cdots, J^{m-1}r^0\}.$$

Arnoldi's method and GMRES both find an approximate solution

$$w_m = w_0 + Z_m, \quad \text{with} \quad Z_m \in K_m,$$

such that either

$$(-f - Jw_m) \perp K_m$$

for Arnoldi's method or

$$\|f + Jw_m\|_2 = \min_{w \in w_0 + K_m} \|f + Jw\|_2 (= \min_{Z \in K_m} \|r^0 - JZ\|_2)$$

for GMRES. Here, $\|.\|_2$ denotes the Euclidien norm on $\mathbb{R}^2$ and orthogonality is meant in the usual Euclidien sense.
In these Krylov methods only the action of the Jacobian J times a vector w, and not J explicitly is required. In the context of problem (8), this action can be approximated by difference quotient of the form

$$J(u)w \approx \frac{f(u + \epsilon w) - f(u)}{\epsilon},$$

where u is the current approximation to a root of (8) and ϵ is a scalar.
Selecting an optimal parameter ϵ in the difference formula for approximating $J(u)w$ might be a difficult problem. 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 $J(u)w$ 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:

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

where $|v| = (|v_1|, ..., |v_n|)^T$, and typ$u$ is a given value depending on u and the problem to be solved. The Krylov method retained in this paper is GMRES. For more detail we refer to [3].

3.3 Preconditioned Newton-Krylov matrix-free methods

The combination of the Krylov matrix-free methods and the inexact-Newton method described above results in the Newton-Krylov matrix-free algorithm introduced in [3]. Although the matrix-free method is attractive because it does not form the matrix explicitly, the matrix is still required for preconditioning purposes. In [22], [23], and [10] 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. However as we mentioned earlier, the prohibitive memory requirements and the computational complexity for the higher-order matrix representation, whether by analytical or numerical means, makes the explicit calculation of such matrix a difficult problem. Moreover, if we decide to compute this matrix explicitly the advantage of the matrix-free method will be lost. In order to overcome these difficulties, we proposed in [24] to form only, the explicit Jacobian matrix corresponding to a discretization that is similar to the defect-correction procedure described in section (2). We derived then an ILU preconditioner based on a lower-order approximation to the true Jacobian. This included: **a)** the Jacobian of a lower-order discretization, **b)** and the Jacobian obtained using a discretization that allows a less expensive analytical evaluation of elements. However, the ILU preconditioner studied there is difficult to parallelize efficiently. Therefore, we propose in this paper to use parallel preconditioners based on Schwarz domain decomposition methods. In which case, the approximation of the global Jacobian is reduced to the approximation of local Jacobians defined on subdomains. The latter case can be combined with any of the first two cases **a)** and **b)**. This results in a mixed discretization in which the preconditioner of the consistent higher-order system (11) is derived using an approximation of the Jacobian matrix that employs a lower-order discretization.

Applying the method proposed above to the fully implicit nonlinear system (5) and (7), yields the following algorithm

- Define δW_0^n, an initial guess.
- For $k = 0, 1, 2, \cdots$ until convergence do

$$\text{Solve} \quad M^{-1}\frac{f(W_k^n + \epsilon\delta W_k^n) - f(W_k^n)}{\epsilon} = -M^{-1}f(W_k^n). \tag{14}$$

$$\text{Set} \quad W_{k+1}^n = W_k^n + \delta W_k^n.$$

Using right preconditioning, (14) is replaced by

$$\text{Solve} \quad \frac{f(W_k^n + \epsilon M^{-1}\delta W_k^n) - f(W_k^n)}{\epsilon} = -f(W_k^n). \tag{15}$$

The preconditioner M^{-1} is constructed using an approximation of a lower-order similar to that used in the defect-correction method to derive the matrix $\mathbf{A}$ as described above.

3.4 Additive and Multiplicative Schwarz methods

Let Ω be a polygonal region in $\mathbb{R}^2$ with boundary $\partial\Omega$. Let n be the total number of interior nodes in Ω. Let

$$Au = f \tag{16}$$

be the linear system of algebraic equations resulting from the application of a finite element, or finite difference discretization of a given set of partial differential equations. Let $\{\Omega_i', \quad i = 1, N_{sd}\}$ be an overlapping decomposition of Ω. Let n_i $(i = 1, \cdots, N_{sd})$ denote the number of nodes in the interior of Ω_i', and A_i the $n_i \times n_i$ matrix corresponding to the discretization scheme on the mesh in Ω_i'. Let R_i denote the $n_i \times n$ matrix corresponding to the algebraic restriction of a vector of length n defined on Ω to a vector of length n_i defined on Ω_i'. The transpose $(R_i)^T$ corresponds to an algebraic extension in which a vector of length n_i defined on Ω_i' is extended to a vector of length n defined on the whole domain Ω using an extension by zero on $\Omega \setminus \Omega_i'$.

Let u^0 be a given initial guess, and let u^k be the current iterate. The discrete form of the Schwarz method applied to the problem (16) writes

$$u^{k+i/N_{sd}} = u^{k+(i-1)/N_{sd}} + R_i^T A_i^{-1} R_i (f - A u^{k+(i-1)/N_{sd}}), \quad i = 1, \cdots, N_{sd} \tag{17}$$

Under the notation $P_i = R_i^T A_i^{-1} R_i A$, we have

$$u^{k+1} = (I - P_{N_{sd}}) \cdots (I - P_1) u^k + g, \tag{18}$$

with appropriate g. We note that in this paper we do not use the coarse mesh operator, and therefore it is not introduced in the definition of the Schwarz methods given here. Let $O_I = (I - P_{N_{sd}}) \cdots (I - P_1)$ denote the iteration operator. If the iteration (18) converges then its solution v verifies

$$(I - O_I) v = g. \tag{19}$$

The equation above defines the multiplicative algorithm for the solution of the linear system (16).
Now we shall define the additive Schwarz method. We obtain the additive Schwarz method by modifying the iteration (17) into the following algorithm

$$u^{k+i/N_{sd}} = u^{k+(i-1)/N_{sd}} + R_i^T A_i^{-1} R_i (f - A u^k), \quad i = 1, \cdots, N_{sd} \tag{20}$$

This gives the following iteration

$$u^{k+1} = u^k + \sum_{i=1}^{N_{sd}} R_i^T A_i^{-1} R_i (f - Au^k), \quad i = 1, \cdots, N_{sd} \tag{21}$$

If the iteration (21) converges then its solution v is also solution of the following problem

$$\sum_{i=1}^{N_{sd}} R_i^T A_i^{-1} R_i A v = g, \tag{22}$$

with an appropriate g. The above equation defines the additive Schwarz preconditioner for A. We notice that the multiplicative algorithm is a generalization of the block Gauss-Seidel method with overlapping blocks, while the additive method is a generalization of the block Jacobi method (which corresponds to zero overlap).

3.5 Application to the Steady Compressible Euler Problem

The applications of the Schwarz-based methods have been primarily applied to elliptic and parabolic boundary value problems discretized using finite element methods or finite difference methods. In this paper we shall focus on their applications to the hyperbolic Euler problem using the finite volume discretization described in section 2. More precisely, the additive method (22) and the multiplicative algorithm (19) are applied to the linear system (6) in which the matrix $\mathbf{A}$ corresponds to the discretization scheme of section 2. The resulting method is a defect-correction procedure. The numerical performance of this method is studied in the next section.

Now, instead of solving the defect-correction iteration (6), we propose to solve the nonlinear system (5) obtained using the implicit finite volume method (see section 2) by using the Newton-Krylov matrix-free method in which the action of the Jacobian on a any given vector is computed using a finite difference method as described in the subsection 3.3. We then apply the additive method and the multiplicative algorithm to each linear step of the Newton-Krylov matrix-free iteration. The Schwarz preconditioner is constructed using the matrix of a lower-order discretization obtained in a similar fashion to that used to construct the matrix $\mathbf{A}$ in the defect-correction (6) of section 2. The numerical study of this combination of the Schwarz-based methods with the Newton-Krylov matrix-free methods are presented in the next section.

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 (the coarse mesh) and 4096 (the fine mesh) cells, respectively. In all computations performed herein the solution obtained agrees

with the standard one. All these calculations are performed on the same Sparc10 machine. Since we are dealing with different methods which require varying amounts of work at each time step we believe that CPU time is the only true measure for comparing them. In spite of this, we present also comparisons of the iteration counts. The relative tolerance in the solution of the linear system is 10^{-3} for the preconditioned Krylov methods (ILU/GMRES). The steady state regime is declared when the nonlinear residual norm reaches a value of (or less than) 10^{-5}. And in all tables presented in this study, we show the number of nonlinear iterations (time steps) and the CPU time necessary for the solution to reach the steady state regime. We consider also the terminology x-decomposition, y-decomposition and xy-decomposition. The first terminology denotes the decomposition in the x-axis direction, the second one denotes the decomposition in the y-axis direction, and the third one denotes a decomposition in both directions. They are respectively illustrated in part 1, part 2, and part 3 of Table 3, for example.

The implementation of the Newton-Krylov matrix-free methods described in section 3, together with ILU/GMRES solver with explicit and implicit boundary conditions, correspond to the code developed by the author in [24]. This code is based, in its turn, on an EAGLE-derivative code [15] that employs the discretization described in section 2 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]). The Schwarz-based domain decomposition solver uses the PETSc library that was developed at Argonne National Laboratory [9].

Next, a comprehensive study of the combination of the Schwarz-based methods with defect-correction procedures with explicit and implicit boundary conditions is reported. It is then followed by a study of the combination of the Schwarz-based methods with the Newton-Krylov matrix-free methodology.

4.1 Study of Schwarz-based methods combined with defect-correction procedures: Coarse mesh and explicit boundary conditions case

We first study the performance of Schwarz-based methods combined with the defect-correction procedures. This study is done for both explicit and implicit boundary conditions. We focus first, on the use of the full nested dissection method as a subdomain solver. The use of incomplete factorizations together with GMRES methods, in replacement of the full nested dissection methods for the subdomain solvers, is then considered. We note that one often uses the full nested dissection methods for the solution of subdomain problems. Using then the preconditioned Krylov methods as subdomain solvers, we perform several comparisons of different Schwarz-based methods on the test problem described above for various decompositions. An important parameter related to the use of the Schwarz-based methods is the choice of the over-

lap. This crucial issue is also addressed here, for both the full nested dissection and the preconditioned Krylov subdomain solvers. Another important issue is the choice of a suitable decomposition of the global domain into local subdomains. This is also addressed thoroughly in this study.

4.1.1 Study of the overlap

To study the choice of the overlap for the Schwarz-based methods, we first present in Table 1 the results for different Schwarz methods with an overlap of one mesh size. To see the effect of the overlap on the Schwarz-based methods studied here, we present in Table 2 the results corresponding to an overlap of two mesh sizes for different Schwarz-based methods using the iterative subdomain solvers (ILU/GMRES). We observe first that, for a given subdomain number the number of nonlinear iterations (time steps) varies slightly as we change the subdomain decomposition and/or the Schwarz-based method. Moreover, we observe that the multiplicative Schwarz algorithm outperforms the additive method for all of the various decompositions studied here. We compare now the results performed here to those performed in 4.1.3 corresponding to an overlap of one mesh size (Table 1). We observe that, when the subdomain number increases, the difference between the CPU time cost of the Schwarz algorithms with two and one mesh size overlap increases. And this is even more prohibitive for the additive algorithm. Furthermore this difference is more important for the x-decomposition than for the xy-decomposition.

4.1.2 The full nested dissection subdomain solvers

We shall study next the Schwarz-based methods using the full nested dissection as subdomain solvers. In Table 3 we present the number of nonlinear iterations (time steps) and the CPU time at convergence (steady state regime), for the different Schwarz-based methods and for various decompositions, employing the full nested dissection methods as subdomain solvers. These calculations were performed with a CFL number equal to 6.5. We observe first that, for a given subdomain solver, the number of nonlinear iterations (Table 3), is nearly the same for all of the different Schwarz-based methods and for the various decompositions types. Comparing the block Jacobi method with the multiplicative Schwarz algorithm (Table 3), we observe that the former outperforms the latter for all of the decompositions cosidered here, with the exception of the first x-decomposition. We compare now, the additive with the multiplicative Schwarz methods. For the x-decomposition the latter outperforms the former for up to 8 subdomains, while the situation is reversed for a decomposition of 16 or more subdomains. In the case of the y-decomposition, the multiplicative Schwarz outperforms the additive Schwarz for the first decomposition, and the situation is reversed for the second decomposition. Furthermore, the latter prevails over the former for the last two decompositions. Finally, for the xy-decompositions, the multiplicative Schwarz

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×1	1152	4334	1151	4829	1155	4697
4×1	1151	4349	1150	4987	1157	4746
8×1	1151	4319	1148	4837	1158	4819
16×1	1149	4471	1147	5420	1160	5083
32×1	1149	5116	1144	6208	1162	5957
64×1	1154	6703	1145	8122	1163	7834
128×1	1163	10720	1161	11982	1163	10987
1×2	1152	4525	1160	6395	1158	5423
1×4	1152	4536	1154	6512	1162	6274
1×8	1154	5507	1154	8871	1163	7129
1×16	1158	8226	1157	12477	1164	10036
2×2	1151	4779	1158	6129	1158	5056
4×4	1148	4398	1145	6664	1163	5532
8×8	1151	5356	1143	9485	1163	6788

Table 1: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and incomplete factorizations on the coarse-mesh case.

	Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time
2×1	1153	5155	1155	4900
4×1	1151	5074	1158	4681
8×1	1150	5225	1159	4786
16×1	1149	5639	1161	5195
32×1	1147	6889	1163	6152
64×1	1146	9704	1163	8682
128×1	1161	15550	1163	13467
1×2	1165	6743	1160	5918
1×4	1159	8352	1163	6695
1×8	1158	10744	1164	8423
1×16	1157	18147	1164	12524
2×2	1165	7093	1160	5535
4×4	1149	8896	1163	6169
8×8	1147	12002	1163	8043

Table 2: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and incomplete factorizations and with an overlap of two mesh sizes on the coarse-mesh case.

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×1	927	8150	926	8297	926	8060
4×1	927	7033	926	7248	926	7125
8×1	927	6159	926	6483	926	6427
16×1	926	5052	926	6280	926	6742
32×1	926	4969	927	6746	926	7129
64×1	926	5691	926	8490	926	8503
128×1	927	7864	926	11555	926	11623
1×2	927	6464	927	7664	926	7507
1×4	928	5385	927	7635	927	7813
1×8	927	5443	926	9683	927	9378
1×16	929	6879	926	13289	927	12848
2×2	927	6010	927	7877	926	7000
4×4	927	4746	928	7589	926	6937
8×8	927	4692	926	9129	927	8060

Table 3: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and full nested dissection on the coarse mesh-case.

methods outperform the additive Schwarz methods for the four cases. It is also interesting to notice that, for the x-decomposition with 16 or more subdomains the additive method prevails over the multiplicative one, while this situation is reversed for the xy-decompositions (4×4 and 8×8).

4.1.3 The preconditioned Krylov subdomain solvers

We shall study now, the Schwarz-based methods using the preconditioned Krylov methods as subdomain solvers. In Table 1 we present the number of nonlinear iterations (time steps) and the CPU time at convergence (steady state regime), for the different Schwarz-based methods and for various decompositions. These calculations were performed using a CFL number equal to 5. We notice here that, this CFL is smaller than the one used for the direct subdomain solver (CFL=6.5). For this subdomain solver, the situation is quite smooth. More precisely, comparing the results given in Table 1, we observe that the block Jacobi method outperforms the multiplicative Schwarz algorithm, which in its turn, prevails over the additive method. Moreover, the above observations are valid for all of the various decompositions studied here. For a given subdomain solver, the number of nonlinear iterations (time steps) (Table 1), is also

nearly the same for all of the different Schwarz-based methods and for the various decomposition types.

4.1.4 The full nested dissection versus the preconditioned Krylov subdomain solvers

Next, we perform comparisons of the subdomain solvers studied above, and study the effect of replacing the full nested dissection subdomain solver by the preconditioned Krylov subdomain solver. In Table 3, the results are obtained using the full nested dissection methods as subdomain solvers, while in Table 1, those results are obtained using the preconditioned Krylov methods (ILU/GMRES). We observe first that, for a given subdomain solver the number of nonlinear iterations (time steps) (Table 3 and 1) is nearly the same for all of the different Schwarz-based methods and for the various decomposition types. For the x-decomposition and xy-decomposition we observe (Table 3 and 1) that, using the full nested dissection as subdomain solvers is more CPU time consuming than using the preconditioned Krylov methods (ILU/GMRES) for up to 16 subdomains. As for the y-decomposition, the preconditioned Krylov subdomain solvers are as attractive as the full nested dissection subdomain solvers only, in the case of a decomposition of the domain into no more than 8 subdomains.

4.1.5 Global ILU and LU solvers

In this section, we discuss the use of ILU and LU as global solvers. In Table 4, we present the number of nonlinear iterations (time steps) and the CPU time at convergence (steady state regime) for the full nested dissection (LU) and the preconditioned Krylov methods (ILU/GMRES) used globally. We observe clearly in Table 4 that to reach the steady state regime the full nested dissection needs more than four times the CPU time corresponding to the preconditioned Krylov method (ILU/GMRES). These observations are in fact not new. It is well known that the full nested dissection methods are prohibitive, both in terms of the memory requirements and the CPU time. However, in the context of Schwarz-based methods the use of the full nested dissection methods is reduced to a local level as subdomain solvers. This makes them more attractive and efficient to use. (Further discussion is reported in the following section). Nonethless, for large problems the use of the full nested dissection methods becomes again prohibitive. And therefore, replacing the full nested dissection solver by the preconditioned Krylov solver results in a more efficient algorithm, as shown above.

4.1.6 Comparisons of the Schwarz-based methods with the global ILU and LU solvers

Comparing the results of Table 3 and Table 4 we clearly see that, the block Jacobi outperforms the global full nested dissection method for all of the decompositions

LU		ILU	
Iterations	CPU time	Iterations	CPU time
926	8516	929	2192

Table 4: Iterations counts and CPU times (in seconds) for steady transonic flow at convergence, employing full nested dissection and incomplete factorizations on the coarse mesh.

considered here. The other two Schwarz-based methods outperform up to 64 subdomains the full nested dissection method only in the case of the x-decomposition and the xy-decomposition and with the exception of the additive Schwarz method for the decomposition 8×8. It is also very clear from Tables 1 and 4 that, the global preconditioned Krylov method (ILU/GMRES) outperforms all of the different Schwarz-based methods for all of the decompositions considered in this study. However, the Schwarz-based methods have several advantages over the global preconditioned Krylov methods (ILU/GMRES). The ILU preconditioner is difficult to parallelize efficiently. Moreover, the Schwarz-based methods and more particularly, the additive Schwarz algorithm, provide efficient and more attractive parallel algorithms. By reducing the solution of the global problem into the solution of local subproblems the Schwarz-based methods allow also to solve very large problems, and therefore, they are preferable to use.

4.1.7 Study of the different decomposition strategies

The above study shows that the use of the preconditioned Krylov methods as subdomain solvers for the different Schwarz-based methods studied in this paper is more attractive than that of the full nested dissection methods. Therefore, we shall study the different decomposition strategies only, for the iterative solver (Table 1). We shall compare the three decomposition startegies for each class of Schwarz methods reported in this paper. For the block Jacobi method it is clear that better performance in terms of the CPU time is obtained using the x-decomposition than the y-decomposition. Moreover, the xy-decomposition prevails over the x-decomposition for the last two decompositions (4×4 and 8×8), but not for the first one (2×2). For the additive Schwarz method the x-decomposition prevails over the xy-decomposition which, in its turn, prevails over the y-decomposition. The same conclusions are also true for the multiplicative Schwarz method with the particular exception for the 64 subdomains case where the xy-decomposition prevails over the x-decomposition.

4.2 Study of Schwarz-based methods combined with defect-correction procedures: Fine-mesh case and explicit boundary conditions

We study now the performance of Schwarz-based methods combined with the defect-correction procedures in the case of explicit boundary conditions, using the fine mesh described earlier. First, the study of the choice of the overlap is performed. In Table 7 we represent the results corresponding to an overlap of two mesh size for the different Schwarz algorithms using the iterative subdomain solver (ILU/GMRES) that we compare to the results obtained using one mesh size overlap Table 6. The conclusions are similar to the coarse mesh case. The use of the full subdomain solvers. The results are illustrated in Table 5 where the number of nonlinear iterations (time steps) and the CPU time at convergence (steady-state regime) for the different Schwarz-based methods and for various decompositions are presented. These calculations are performed with a CFL number equal to 5. Similar conclusions as those obtained for the coarse mesh are drawn. The replacement of the full subdomain solver by the preconditioned Krylov subdomain solver (ILU/GMRES) is then performed. Table 6 illustrates the number of nonlinear iterations (time steps) and the CPU time at convergence (steady-state regime) for the different Schwarz-based methods and for various decompositions, employing this iterative subdomain solver. These calculations were performed with a CFL number equal to 4.5. Again, we obtain the same conclusions as those obtained for the coarse mesh case.

4.2.1 Conclusions

In the above sections, we have studied several aspects of the Schwarz-based algorithms with explicit treatment of the boundary conditions. We have shown that, the preconditioned Krylov subdomain solvers result in a more efficient algorithm in terms of the CPU time and the memory requirements as compared to the full nested subdomain solvers. We have also shown that, the block Jacobi method with a large number of subdomains becomes more prohibitive in terms of the convergence rate using the preconditioned Krylov subdomain solvers than the full nested dissection subdomain solvers. However, using the additive and multiplicative methods the preconditioned Krylov subdomain solvers prevail over the direct subdomain solvers. This clearly shows that, the block Jacobi methods perform well with the direct subdomain solvers. This has to be expected since the use of the block Jacobi method relies on giving up some information. It shows also that for the Schwarz-based methods to be efficient with the iterative subdomains solvers an overlap is needed. Moreover, the study of Schwarz-based methods with different overlaps leads to the fact that, an overlap of one mesh size corresponds to an optimal and efficient choice in terms of the convergence rate. Finally, both the x-decomposition and the xy-decomposition are found to be preferable to

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×1	1483	37573	1483	38048	1483	37213
4×1	1483	31215	1483	32672	1483	32169
8×1	1484	22930	1483	26475	1483	26007
16×1	1483	18445	1483	23234	1483	23351
32×1	1483	16442	1483	23643	1483	25048
64×1	1483	18410	1483	29007	1483	28858
128×1	1483	24298	1483	46536	1483	46931
1×2	1485	29566	1484	33559	1483	33294
1×4	1484	21728	1484	27205	1484	27064
1×8	1485	18574	1484	27407	1484	28104
1×16	1485	19169	1484	33095	1484	32892
2×2	1484	27090	1484	31576	1483	28936
4×4	1483	18677	1484	26393	1484	23356
8×8	1484	15233	1485	25755	1484	23382

Table 5: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and full nested dissection on the fine-mesh case.

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×1	1602	13269	1460	13866	1482	13657
4×1	1599	13260	1502	14142	1576	13538
8×1	1599	12855	1484	14111	1535	13219
16×1	1601	12680	1469	14422	1489	13426
32×1	1599	14142	1457	16602	1480	15813
64×1	1606	17779	1458	20924	1483	19755
128×1	1620	28350	1519	33944	1483	28968
1×2	1604	13755	1612	16588	1610	15840
1×4	1599	14014	1598	17665	1614	16735
1×8	1598	14475	1599	20843	1618	18542
1×16	1605	17975	1595	25266	1619	23222
2×2	1602	13220	1609	17939	1611	14749
4×4	1586	12452	1584	18671	1612	15039
8×8	1590	13104	1585	21186	1616	17285

Table 6: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and incomplete factorizations on the fine-mesh case.

	Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time
2×1	1602	15477	1605	14817
4×1	1600	15438	1613	14301
8×1	1605	15969	1609	15014
16×1	1597	16961	1612	16337
32×1	1596	20423	1618	18155
64×1	1589	29459	1620	25604
1×2	1617	17512	1612	16591
1×4	1603	19975	1617	17890
1×8	1601	25888	1619	20550
1×16	1600	33261	1620	27890
2×2	1618	19727	1613	15728
4×4	1591	21481	1616	16777
8×8	1588	28923	1618	19464

Table 7: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with explicit boundary conditions and incomplete factorizations with an overlap of two mesh sizes on the fine-mesh case.

the y-decompositiion. However, for large number of subdomains the xy-decomposition might be preferred.

Therefore, in the rest of this paper we will take into account the above conclusions. More particularly, the Schwarz-based methods combined with other algorithms and/or other treatments of the boundary conditions will be studied only, for the following cases:

i) The subdomain solver corresponds to the preconditioned Krylov method.

ii) Only an overlap of one mesh size will be considered.

iii) And the type of decomposition retained is the xy-decomposition.

4.3 Study of Schwarz-based methods combined with defect-correction procedures: Implicit boundary conditions case

Now, we study the performance of the combination of the different Schwarz-based methods and the defect-correction procedures using implicit boundary conditions. This study is performed for both the coarse-mesh and the fine-mesh cases. A comparison with the previous study for the explicit boundary conditions is also reported.

4.3.1 Study of the performance of Schwarz-based methods

We shall compare here, the different Schwarz-based methods with an implicit treatment of the boundary conditions. For the coarse-mesh case (Table 8), we observe clearly that, the block Jacobi method outperforms the additive Schwarz method. Moreover, the multiplicative Schwarz method outperforms the block Jacobi method for the last two decompositions. For the first decomposition the latter prevails over the former. As for the fine-mesh case (Table 9) we observe again that, the block Jacobi method outperforms the additive method. Furthermore, the block Jacobi method prevails also over the multiplicative Schwarz method with a close performance for the 8×8 decomposition.

4.3.2 Comparisons of the different Schwarz-based methods using explicit and implicit boundary conditions

We compare now, the performance of the different Schwarz-based methods using implicit and explicit boundary conditions. The author has shown in [24], that the implicit treatment of the boundary conditions improves the convergence rate for the preconditioned Krylov methods used globally as compared to the explicit one. We focus here, on the local performance of such methods; i.e. their use as subdomains solvers. We start first by the coarse-mesh case. For block Jacobi method the use of implicit

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×2	423	3310	405	4080	432	3459
4×4	425	4031	430	5053	430	3707
8×8	418	6219	422	7099	414	4147

Table 8: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with implicit boundary conditions and ILU/GMRES as a subdomain solver on the coarse-mesh case. These calculations were performed with a CFL number equal to 100.

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×2	547	8911	553	11096	566	9123
4×4	540	9114	552	11899	577	9717
8×8	539	11430	546	16215	574	11482

Table 9: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods with implicit boundary conditions and ILU/GMRES as a subdomain solver on the fine-mesh case. These calculations were performed with a CFL number equal to 100.

treatment of the boundary conditions improves the rate of convergence as compared to the explicit treatment only, in the case of modest numbers of subdomains (Table 1 and 8). For a large number of subdomains say 8×8, the explicit treatment of the boundary conditions prevails over the implicit one. One can find an explanation of this in the conclusions of section 4.2.1. For the overlapped Schwarz-based methods the situation is quite different. For the additive method, the gain in terms of the CPU time realized using implicit boundary conditions is more than 25% (Table 1 and 8). This is even better for the multiplicative method where a gain of more than 32% is realized repectively for the decompositions 2×2, 4×4, and 8×8. The situation becomes even more interesting for the fine-mesh case. First, for the block Jacobi method, the implicit treatment of the boundary conditions improves the rate of convergence for all of the decompositions considered. And a gain of 32%, 27%, and 32% is realized for the decompositions 2×2, 4×4, and 8×8 respectively. A gain in terms of the CPU time of respectively, 38%, 36%, and 24% is also observed for the additive Schwarz. For the multiplicative method, this gain is respectively 38%, 35%, and 34%. From the above study we clearly see that, the Schwarz-based methods with overlap perform better than the zero-overlap block Jacobi method in the context of implicit boundary conditions as compared to explicit one.

4.4 Study of Schwarz-based methods combined with the Newton-Krylov matrix-free methods

We study here, the combination of Schwarz-based methods with the Newton-Krylov matrix-free methods discussed in section 3. For the coarse-mesh case the starting CFL is 60. The same starting CFL was used for all of the methods and decompositions studied here. For the fine-mesh case the starting CFL is 30. And this CFL choice is the same for all of the methods and decompositions studied here. We start by studying the performance of each Schwarz-based method combined with newton-Krylov matrix-free method. This is followed by a comparison of this combination with the previous combination studied in the previous sections, namely the combination of the Schwarz-based method with the defect-correction procedures.

4.4.1 Performance of Schwarz-based methods combined with Newton-Krylov matrix-free methods

We shall focus now, on the comparison of the performance of the different Schwarz-based methods combined with the Newton-Krylov matrix-free methodology. For the coarse mesh case (Table 10), we observe that the block Jacobi method outperforms the additive method only in the case of the first two decompositions. For the third decomposition the latter prevails over the former. The multiplicative method outperforms the additive method with a reduction of the CPU time of more than 50%. We shall

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×2	23	1251	26	2401	22	1132
4×4	26	2115	26	2572	22	1279
8×8	31	3301	33	2988	27	1660

Table 10: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods combined with Newton-Krylov matrix-free methods with implicit boundary conditions on the coarse-mesh case.

	Block Jacobi		Add. Schwarz		Mult. Schwarz	
Decomp.	Iterations	CPU time	Iterations	CPU time	Iterations	CPU time
2×2	31	5474	31	6102	33	8409
4×4	32	5384	28	5708	30	4759
8×8	32	6594	35	7493	25	4106

Table 11: Iteration count and CPU time (in seconds) for steady transonic flow at convergence, for various preconditioner/decomposition pairs, and employing Schwarz-based methods combined with Newton-Krylov matrix-free methods with implicit boundary conditions on the fine-mesh case.

compare now, the performance of the above methods for the fine-mesh case. It is clear from Table 11 that the block Jacobi method prevails over the additive-Schwarz method for the first decomposition. For the two other decompositions the latter prevails over the former with a gain of more than 45% for the last decomposition. We should notice here as a consequence of the above discussion that, the block-Jacobi method does not outperform the multiplicative method. This was the case for the Schwarz-based method combined with defect-correction procedures studied in the previous sections.

4.4.2 Comparison of the Schwarz-based methods combined with Newton-Krylov methods and with the defect-correction procedures

We start first, by comparing the results for the the coarse mesh case. The block Jacobi method in combination with the Newton-Krylov matrix-free methods reduces the CPU time by more than half as compared to its combination with the defect-correction procedures (Table 8 and 10). The same conclusion is valid for the additive Schwarz methods. Furthermore, the multiplicative Schwarz method combined with the Newton-Krylov matrix-free has a distinct advantage over its combination with the

defect-correction procedures. The CPU time corresponding to the latter is almost three times the one corresponding to the former. We shall focus now on the fine-mesh case (Table 9 and 11). The block Jacobi method in combination with the Newton-Krylov methods reduces the CPU time by 39%, 41%, and 42% respectively as compared to its combination with the defect-correction methods, for the decompositions 2×2, 4×4, and 8×8. The additive algorithm combined with the Newton-Krylov methodology reduces the CPU time by 45%, 52%, and 54% respectively for the decompositions 2×2, 4×4, and 8×8, as compared to its combination with the defect-correction procedures. Finally, for the first decomposition the multiplicative Schwarz method combined with the Newton-Krylov methodology reduces the CPU time only by 8% as compared to its combination with the defect-correction procedures. The results are more impressive for the last two decompositions, where the former reduces the CPU time by 51% and 64% respectively, as compared to the latter. To illustrate the overall benefit of the combination of the Schwarz-based algorithms with the Newton-Krylov matrix-free methods as compared to their combination with the defect-correction procedures, using explicit and implicit boundary conditions, we present in Figures 1-6 the curves presenting the logarithm of the nonlinear steady-state residual versus the CPU time. These curves correspond to the three class of Schwarz-based methods studied here (block Jacobi, additive Schwarz, and multiplicative Schwarz) using both the coarse mesh and the fine-mesh. They correspond also to the particular 8×8 subdomain decomposition.

5 Conclusions

In this paper we have proposed and studied several Schwarz-based methods. More particularly, we have performed the following developments:

i) A full study of the Schwarz-based methods combined with the standard defect-correction procedures with explicit boundary conditions.

ii) The effect of implicit treatment of the boundary conditions on the above combination.

iii) The study of Schwarz-based methods combined with Newton-Krylov matrix-free methods and their comparisons with the combination studied in **i)**.

The different issues related to the use of Schwarz-based methods such as the size of the overlap, the choice of the decomposition, and the use of the direct and iterative methods as subdomain solvers, were thoroughly investigated. Taking into account the different conclusions of these investigations, we have studied the effect of replacing the explicit treatment of the boundary conditions by an implicit one. We have shown in particular that, an important gain in terms of the rate of convergence can be achieved through the use of implicit boundary conditions in the context of Krylov subdomain

solvers as compared to the use of explicit boundary conditions. The development and the study of the combination of Schwarz-based methods with Newton-Krylov matrix-free methods has been then performed. The performance of the preconditioned Newton-Krylov matrix-free methods used globally has been done in [22]-[24]. We have shown in this paper, the performance of this methods used locally; i.e. in combination with the Schwarz-based methods.

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, 16 (1978), 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] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton Methods,* SIAM J. Numer. Anal. **19** (1982), 400–408.

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

[8] M. Dryja and O. Widlund, *Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems,* in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, 1989.

[9] W. D. Gropp and B. F. Smith, *Simplified Linear Equations Solvers Users Manual*, ANL-93/8, Argonne National Laboratory, 1993.

[10] 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.

[11] P. Le Tallec and M. D. Tidriri, *Convergence of domain decomposition algorithms with full overlapping for the advection-diffusion problems*, INRIA Research Report RR-2435, Oct. 1994, also submitted to Math. Comp.

[12] P.-L. Lions, *On the Schwarz alternating method. I.,* in First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations (R. Glowinski, G. H. Golub, G. A. Meurant and J. Périaux, eds), SIAM, Philadelphia.

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] A. H. Schwarz, *Gesammelete Mathematische Abhandlungen,* vol. 2, Springer, Berlin, 1890, pp. 133-143.

[20] 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.

[21] 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.

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

[23] M. D. Tidriri, *Domain Decompositions for Compressible Navier-Stokes Equations*, J. Comp. Phys. **119** (1995), 271-282.

[24] M. D. Tidriri, *Krylov Methods for Compressible Flows*, ICASE Report No. 95-48 June, 1995.

[25] 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.

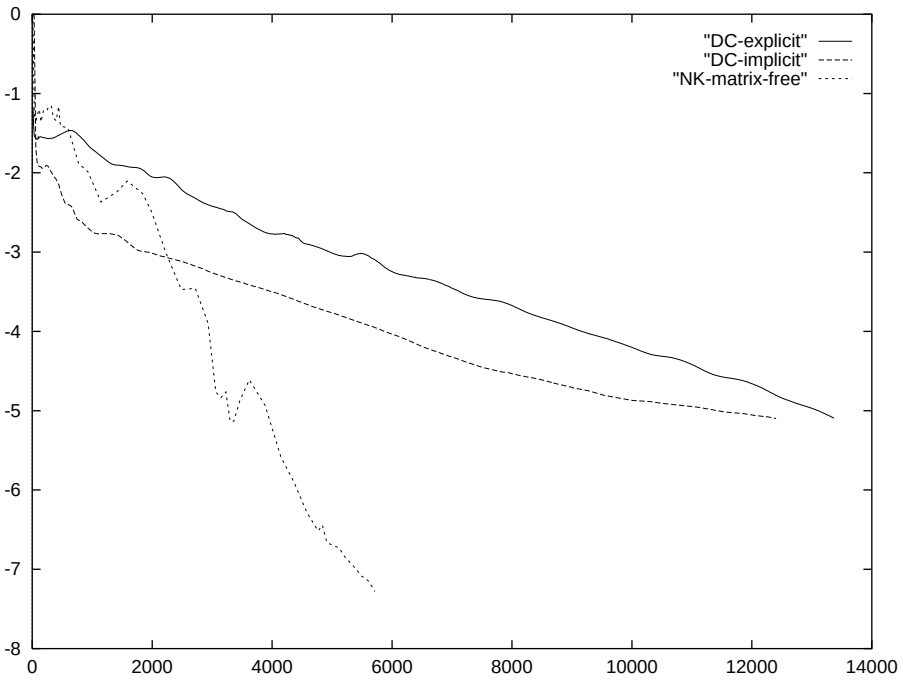

Figure 1: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the block Jacobi algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the coarse mesh.

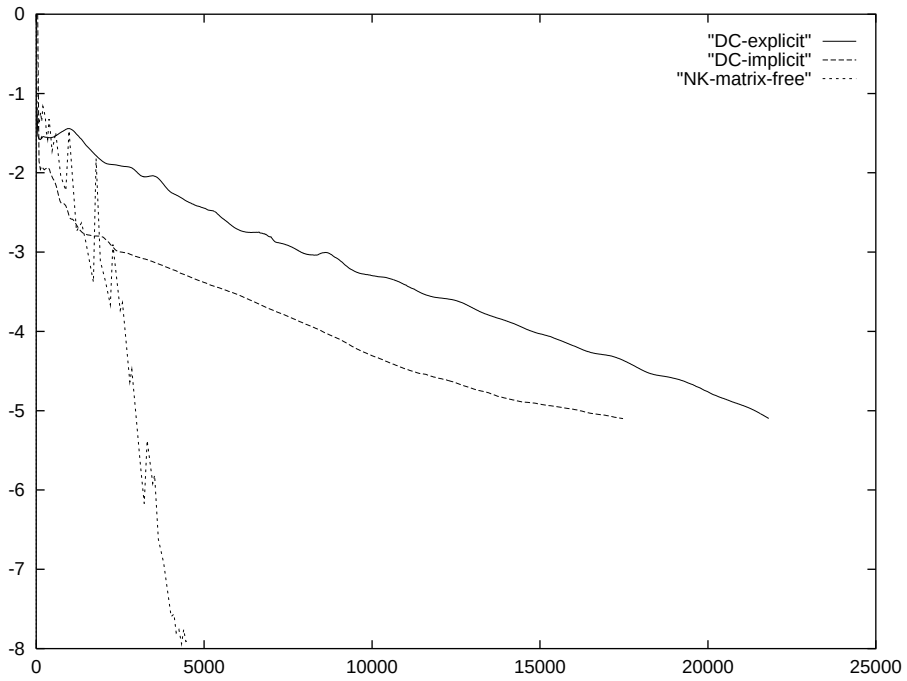

Figure 2: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the additive Schwarz algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the coarse mesh.

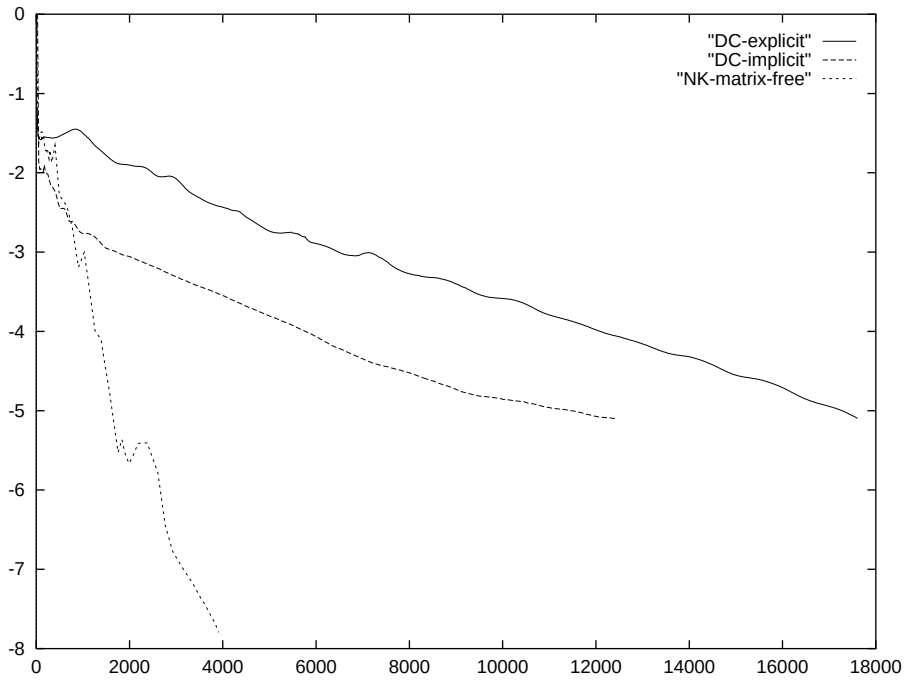

Figure 3: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the multiplicative Schwarz algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the coarse mesh.

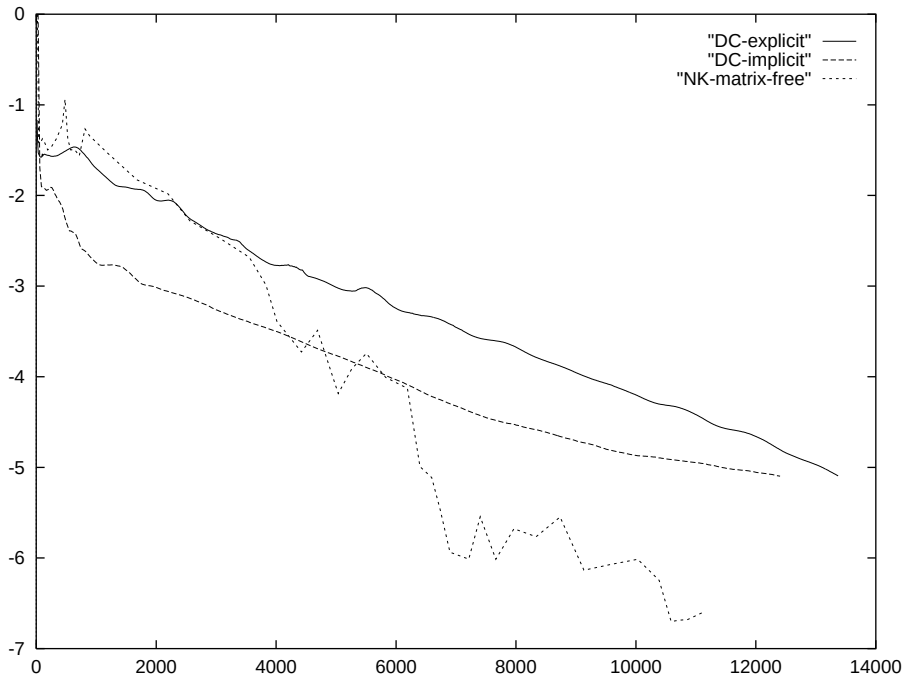

Figure 4: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the block Jacobi algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the fine mesh.

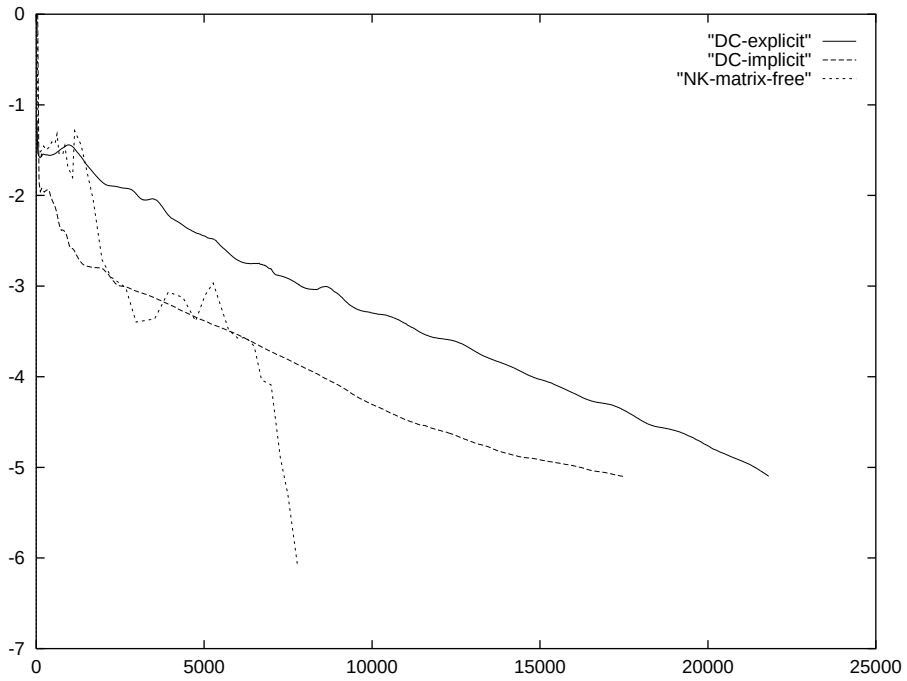

Figure 5: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the additive Schwarz algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the fine mesh.

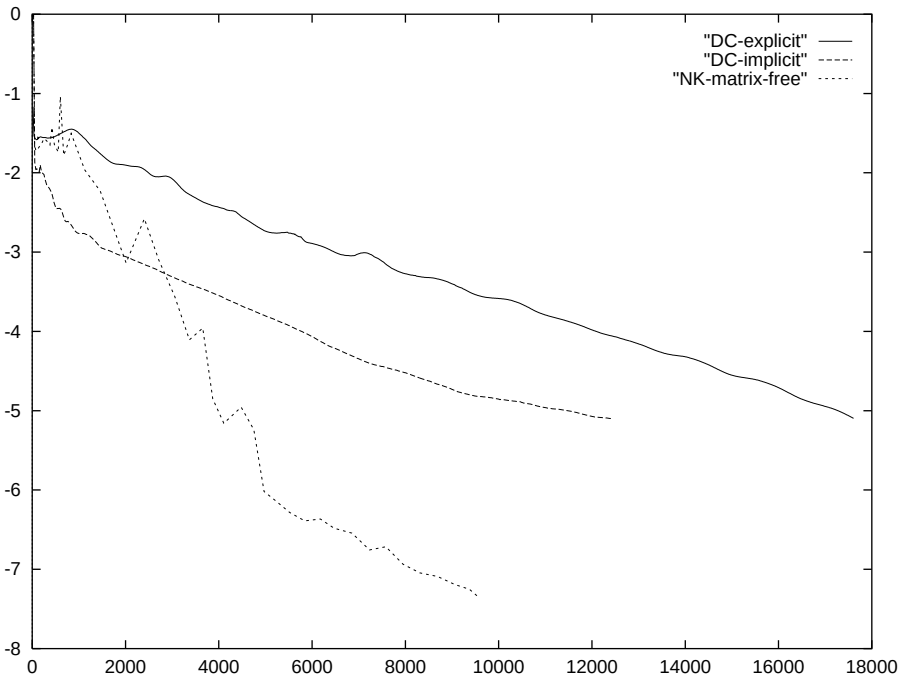

Figure 6: Steady-state residual versus CPU time (in seconds) for steady transonic flow at convergence for the 8×8 decompositions, employing the multiplicative Schwarz algorithm combined with defect-correction procedures with explicit (DC-explicit) and implicit (DC-implicit) boundary conditions, and with Newton-Krylov matrix-free (NK-matrix-free) methods on the fine mesh.