ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 391–400

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

A scaled and minimum overlap restricted additive Schwarz method with application to aerodynamics

Marcus Sarkis [a,*], Bruno Koobus [b,1]

[a] *Department of Computer Science, University of Colorado, Boulder, CO 80309-0430, USA*
[b] *INRIA Sophia-Antipolis, 2004 Route des Lucioles 06902, Valbonne, France*

Abstract

A variant of the classical additive Schwarz preconditioner (AS) is presented and applied to the solution of a general class of two- and three-dimensional flow problems. The scaled restricted additive Schwarz (RAS) with minimal overlap preconditioner is easy to parallelize since all the local communications among processors only involve information pertaining to the interface of the nonoverlapping subdomains. The new method is superior to AS and the Jacobi algorithm in terms of both iteration counts and CPU time, as well as the communication cost when implemented on distributed memory computers.

1. Introduction

Numerical simulations of unsteady three-dimensional compressible flow problems require the solution of large, sparse, nonlinear systems of equations arising from the discretization of Euler or Navier–Stokes equations on unstructured. In this paper we study a highly parallel, scalable and robust nonlinear iterative method (DeC–Krylov–Schwarz) based on the Defect Correction method (DeC), the Krylov subspace method (Krylov), the minimum overlap restricted additive Schwarz method (RAS) and the incomplete LU factorization technique (ILU). We shall present the new method as algebraic preconditioners for general sparse linear systems. The "RAS" method converged faster than the additive Schwarz method, the GMRES method and Jacobi method both in terms of iteration counts and CPU time.

One important application of unsteady flow simulation is the case of flow problem with moving boundaries. In Section 2, we formulate the Navier–Stokes equations in Arbitrary Lagrangian Eulerian approach and overview a second-order discretization in space for unstructured finite volumes and/or finite elements. In Section 3, we present a discrete version of the GCL for second-order implicit temporal discretizations. In Section 4, we discuss the solution of the resulting system of nonlinear equation using the Defect Correction method. In Section 5, we discuss preconditioned iterative methods, Schwarz methods, and introduce our scaled Krylov–RAS solver with minimal overlap. In Section 6, we test the capability of the algorithms developed in this paper for a three-dimensional transonic Euler flow around an oscillating wing and low-speed Navier–Stokes flow past a square cylinder. Finally, in Section 7 we conclude this paper and comment on the parallel performance of the investigated methods.

* Corresponding author. Present address: Mathematical Sciences Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609-2280, USA.

E-mail addresses: msarkis@wpi.edu (M. Sarkis), koobus@math.univ-montp2.fr (B. Koobus).

[1] Present address: Université de Montpellier II, case 051, Département de Sciences Mathématiques, Place Eugène Bataillon, 34095 Montpellier Cedex 5, France.

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00236-4

2. Formulation and semidiscretization of the ALE Navier–Stokes equations

Let $\Omega(t) \subset \mathscr{R}^n$ $(n = 2, 3)$ be the flow domain of interest, and $\Gamma(t)$ be its moving and/or deforming boundary. We introduce a mapping function between $\Omega(t)$, where time is denoted by t and a grid point's coordinates by x, and a reference configuration $\Omega(0)$ and a grid point's coordinates by ξ, as follows:

$$x = x(\xi, t). \tag{1}$$

The ALE nondimensional conservative form of the Navier–Stokes equations describing viscous flows on dynamic meshes can be written as [11,14,15]

$$\left.\frac{\partial J\mathscr{W}}{\partial t}\right|_{\xi} + J\nabla_x \cdot \mathscr{F}^{\mathrm{c}}(\mathscr{W}, \dot{x}) = J\nabla_x \cdot \mathscr{R}(\mathscr{W}), \tag{2}$$

$$\mathscr{F}^{\mathrm{c}}(\mathscr{W}, \dot{x}) = \mathscr{F}(\mathscr{W}) - \dot{x}\mathscr{W},$$

where a dot superscript designates a time derivative, $J = \det(\mathrm{d}x/\mathrm{d}\xi)$, $\dot{x} = (\partial x/\partial t)|_{\xi}$ is the grid speed, $\mathscr{W}$ is the fluid state vector, $\mathscr{F}^{\mathrm{c}}$ denotes the ALE convective fluxes, $\mathscr{F}$ is the usual Euler fluxes and $\mathscr{R}$ the diffusive fluxes. Eq. (2) describes the conservation of the fluid state on the reference configuration $\Omega(0)$.

We semidiscretize Eq. (2) on a triangulation (two-dimensional problems) or a tetrahedral mesh (three-dimensional problems) from which we derive a dual mesh defined by control volumes or cells (Fig. 1).

We first integrate Eq. (2) over a reference cell $C_i(0)$ of the ξ space; next we switch from the ξ reference space to the x space at time t; and finally we integrate by parts the convective and diffusive fluxes which leads to

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{C_i(t)} \mathscr{W}\, \mathrm{d}\Omega_x + \int_{\partial C_i(t)} \mathscr{F}^{\mathrm{c}}(\mathscr{W}, \dot{x}) \cdot \vec{n}_i\, \mathrm{d}\sigma = \int_{\partial C_i(t)} \mathscr{R}(\mathscr{W}) \cdot \vec{n}_i\, \mathrm{d}\sigma, \tag{3}$$

where $\vec{n}_i$ denotes the normal to the cell boundary $\partial C_i(t)$ (see [4] for more details).

We resolve the ALE convective fluxes by a suitable Riemann solver [17,8,12,9], and approximate the diffusive terms by piecewise linear finite elements. The resulting semidiscrete version of Eq. (3) is

$$\frac{\mathrm{d}}{\mathrm{d}t}(M_i W_i) + F_i(W, X, \dot{X}) = R_i(W, X), \tag{4}$$

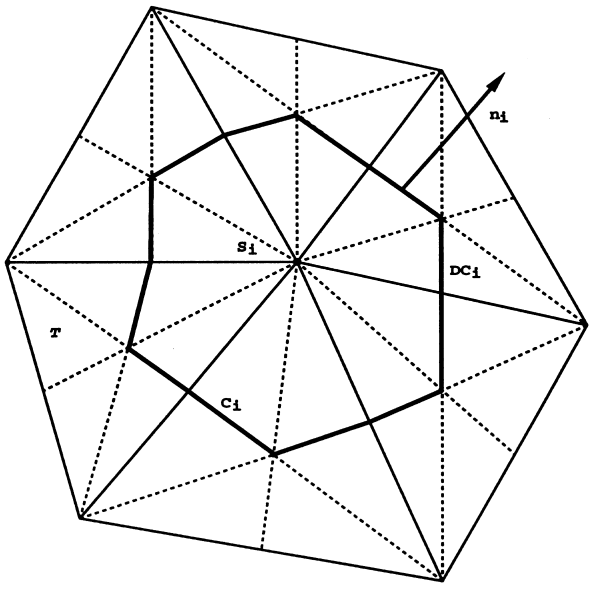

Fig. 1. A control volume in a two-dimensional unstructured mesh.

where $M_i = \int_{C_i(t)} \mathrm{d}\Omega_x$, W_i denotes the average value of $\mathscr{W}$ over the cell $C_i(t)$, F_i and R_i denote, respectively the semidiscrete ALE convective and diffusive fluxes, W is the vector formed by the collection of W_i, X is the vector of time-dependent grid point positions, and $\dot{X}$ the mesh velocities vector.

3. Implicit time-integration of the semidiscrete ALE Navier–Stokes equations

Let t^n and $\Delta t^n = t^{n+1} - t^n$ denote the nth time-station and the $(n+1)$th time-step, respectively. Integrating Eq. (4) between t^n and t^{n+1} leads to

$$\int_{t^n}^{t^{n+1}} \frac{\mathrm{d}}{\mathrm{d}t}(M_i W_i)\,\mathrm{d}t + \int_{t^n}^{t^{n+1}} F_i(W,X,\dot{X})\,\mathrm{d}t = \int_{t^n}^{t^{n+1}} R_i(W,X)\,\mathrm{d}t. \tag{5}$$

For the desired large time-steps, the proper evaluation of the integrals $\int_{t^n}^{t^{n+1}} F_i(W,X,\dot{X})\,\mathrm{d}t$ and $\int_{t^n}^{t^{n+1}} R_i(W,X)\,\mathrm{d}t$, which means the determination of the mesh configurations where these integrals have to be integrated, has a dramatic effect on accuracy. This specific issue has been addressed in [14,15] for first-order time-accurate schemes, and more recently in [4] for second-order time-accurate algorithms. Here, we summarize the approach presented in [15,4], and specify the second-order time-integration algorithm adopted.

A second-order time-accurate implicit algorithm that is popular in CFD is the second-order backward difference scheme. A generalization of this algorithm for dynamic meshes that addresses the questions raised above can be written as

$$\begin{aligned}&\alpha_{n+1} M_i^{n+1} W_i^{n+1} + \alpha_n M_i^n W_i^n + \alpha_{n-1} M_i^{n-1} W_i^{n-1} \\ &\quad + \Delta t^n \Psi_i(W^{n+1}, X^{n-l}, \ldots, X^n, \ldots, X^{n+m}, \dot{X}^{n-j}, \ldots, \dot{X}^n, \ldots, \dot{X}^{n+k}) = 0,\end{aligned} \tag{6}$$

where j, k, l and m are positive integers, $X^n = X(t^n)$,

$$\alpha_{n+1} = \frac{1+2\tau}{1+\tau}, \quad \alpha_n = -1-\tau, \quad \alpha_{n-1} = \frac{\tau^2}{1+\tau}, \quad \tau = \frac{\Delta t^n}{\Delta t^{n-1}}$$

and

$$\Psi_i = \sum_s w_s^c F_i(W^{n+1}, X^{n_s^c}, \dot{X}^{n_s^c}) - w_s^d R_i(W^{n+1}, X^{n_s^d}).$$

Here $M_i^n = M_i(X^n)$, $W_i^n = W_i(X^n)$, w_s^c and w_s^d are real coefficients that satisfy $\sum_s w_s^c = 1$, $\sum_s w_s^d = 1$. $X^{n_s^c}$, $X^{n_s^d}$ and $\dot{X}^{n_s^c}$ are some linear combinations of the mesh configurations $\{X^{n-l}, \ldots, X^n, \ldots, X^{n+m}\}$ and their velocities $\{\dot{X}^{n-j}, \ldots, \dot{X}^n, \ldots, \dot{X}^{n+k}\}$. An important issue is then the proper construction of Ψ_i so that the generalized algorithm (6) retains as much as possible second-order time-accuracy on moving grids.

It can be shown that a sufficient condition for the time-integrator (6) to be mathematically consistent – that is, to be at least first-order time-accurate – is to predict exactly the state of a uniform flow. This sufficient condition, which was formulated in [4] as a geometric conservation law (GCL), can be used to determine the coefficients w_s^c and the mesh configurations $(X^{n_s^c}, \dot{X}^{n_s^c})$. It is shown in [4] that for three-dimensional problems, the time-integrator (3) equipped with the following four w_s^c coefficients and four mesh configurations $(X^{n_s^c}, \dot{X}^{n_s^c})$ satisfies the GCL and achieves high accuracy:

$$w_1^c = \frac{\alpha_{n+1}}{2}; \quad w_2^c = \frac{\alpha_{n+1}}{2}; \quad w_3^c = -\frac{\alpha_{n-1}}{2\tau}; \quad w_4^c = -\frac{\alpha_{n-1}}{2\tau},$$

$$d_1 = \frac{1}{2}\left(1 - \frac{1}{\sqrt{3}}\right); \quad d_2 = \frac{1}{2}\left(1 + \frac{1}{\sqrt{3}}\right),$$

$$x^{n_1^c} = d_1 x^{n+1} + d_2 x^n; \quad x^{n_2^c} = d_2 x^{n+1} + d_1 x^n; \quad x^{n_3^c} = d_1 x^n + d_2 x^{n-1}; \quad x^{n_4^c} = d_2 x^n + d_1 x^{n-1},$$

$$\dot{x}^{n_1^c} = \frac{x^{n+1} - x^n}{\Delta t^n}; \quad \dot{x}^{n_2^c} = \frac{x^{n+1} - x^n}{\Delta t^n}; \quad \dot{x}^{n_3^c} = \frac{x^n - x^{n-1}}{\Delta t^{n-1}}; \quad \dot{x}^{n_4^c} = \frac{x^n - x^{n-1}}{\Delta t^{n-1}}.$$

The principle of conservation of the state of a uniform flow W^* cannot be used as a guideline for determining the viscous coefficients w_s^d and the corresponding mesh configurations $X^{n_s^d}$, because $R_i(W^*, X^{n_s^d}) = 0$. However, these unknowns can be determined by performing a truncation error analysis of $\int_{t^n}^{t^{n+1}} R_i(W^{n+1}, X)\,\mathrm{d}t$ using a Taylor series expansion, and requiring that the quantity $\Delta t^n \sum_s w_s^d R_i(W^{n+1}, X^{n_s^d})$ approximates this integral with an error $\mathrm{O}(\Delta t^3)$. It is shown in [5] that for three-dimensional problems

$$\int_{t^n}^{t^{n+1}} R_i(W,x)\,\mathrm{d}t = \Delta t R_i\left(W^{n+1}, \frac{x^n + x^{n+1}}{2}\right) + \mathrm{O}(\Delta t^3).$$

It follows that the time-integrator (3) can be used with the following w_s^d coefficient and mesh configuration $x^{n_s^c}$

$$w_1^d = 1,$$
$$x^{n_1^d} = \frac{x^n + x^{n+1}}{2}.$$

4. Implicit iterative defect correction method

The time-integration methodology described in the previous sections leads at each time-step to the following set on nonlinear equations

$$\alpha_{n+1} M_i^{n+1} W_i^{n+1} + \Delta t^n \Psi_i = -\alpha_n M_i^n W_i^n - \alpha_{n-1} M_i^{n-1} W_i^{n-1},$$

where

$$\Psi_i = \sum_s w_s^c F_i(W^{n+1}, X^{n_s^c}, \dot{X}^{n_s^c}) - w_s^d R_i(W^{n+1}, X^{n_s^d}),$$

where Ψ_i is second-order space accurate and nonlinear.

It is well-known that constructing a second-order accurate spatial discretization of the jacobian $\partial\Psi/\partial W$ is a complex and expensive task [3]. Then, the Newton's method becomes ineffective for many unsteady state aerodynamic simulations. One of the effective techniques for solving (4) is based on a defect-correction (Newton-like) method [1] in which a first-order semidiscretization of the jacobian $\partial\Psi/\partial W$ is used. The convergence properties of this method have been analyzed in [3,16]. For fixed meshes, it was shown in [16] that two iterations suffice to produce a solution that is second-order accurate both in space and time.

The so-called Defect Correction method is described as follows. Suppose that we have an initial guess $W_i^{n+1,0}$ for W_i^{n+1} obtained by using information calculated at previous time steps. In numerical examples of this paper we consider $W^{n+1,0} = W^n$. We iterate for $j = 0, 1, \ldots,$

$$W_i^{n+1,j+1} = W_i^{n+1,j} + \xi_i^j, \tag{7}$$

where ξ_i^j is the solution of the following linear system of equations

$$\left(\alpha_{n+1} M_i^{n+1} + \Delta t^n \frac{\partial \Psi_i^{(1\mathrm{st})}}{\partial W}(W^n)\right)\xi_i^j = g_i^{n+1,j}, \tag{8}$$

where

$$g_i^{n+1,j} = -\Delta t^n \Psi_i(W^{n+1,j}) - \alpha_{n+1} M_i^{n+1} W_i^{n+1,j} - \alpha_n M_i^n W_i^n - \alpha_{n-1} M_i^{n-1} W_i^{n-1}.$$

Here $\Psi^{(1\mathrm{st})}(\cdot)$ is a first order space accurate Roe's numerical flux. To simplify the notation, we denote $g^{n+1,j}$ the *nonlinear residual vector* at the jth DeC iteration of the $(n+1)$th time step and re-write (8) as

$$A_n \xi^j = g^{n+1,j}. \tag{9}$$

We remark that (4) does not have to be solved exactly. All we need is to drive the nonlinear residual to below a certain *nonlinear tolerance* $\tau > 0$, i.e.,

$$\|g^{n+1,j}\|_2 \leqslant \tau \|g^{n+1,0}\|_2, \tag{10}$$

such that $W^{n+1,j}$ gives a second order accurate solution in both space and time. Also (9) does not need to be solved very accurately either, as its solution provides only a search direction for the outer DeC iteration.

5. Preconditioned iterative methods

Preconditioned iterative methods are often used for finding a $\tilde{\xi}^j = C_n \eta^j$ such that

$$\|B_n(A_n C_n \eta^j - g^{n+1,j})\|_2 \leqslant \delta \|B_n g^{n+1,j}\|_2$$

for a certain *linear tolerance* $\delta > 0$. Here B_n and C_n are left and right preconditioners for A_n and $\tilde{\xi}^j$ is an approximation of ξ^j.

The effectiveness of the nonlinear implicit solver depends heavily, among other things, on the choice of the preconditioner and a balanced selection of the nonlinear and linear stopping tolerance τ and δ. In this paper, we focus on the study of a parallel restricted additive Schwarz preconditioned iterative method for solving (9). For the numerical examples that we test in this paper, we use $\delta = 0.01$, and as a result only two Defect Correction iterations are need to obtain a good accuracy.

We next describe four different linear solvers for solving a sparse linear system $Ax = g$ such as (9). The difference between the linear solvers is in how we define the left preconditioner B, the right preconditioner C, and the accelerator that is used. The accelerators that we consider are GMRES ([18]) and Richardson methods. For all the following iterative linear solvers, we take as initial guess $x_0 = 0$. In order to compare the performance of the different methods, we take B to be the same for all methods and consequently the same stopping criterion for solving the linear system can be used, i.e.,

$$\|B(Ax_K - g)\|_2 \leqslant \delta \|Bg\|_2. \tag{11}$$

Here, x_K is an approximation of x at iteration K.

5.1. Block Jacobi method

Let the matrix $B = D^{-1}$, where the block diagonal matrix D is defined as the diagonal blocks 5×5 of A. Let the matrix C be the identity matrix. The Block Jacobi method, also called Richardson iterative method with block diagonal preconditioning, is defined as follows: for $k = 0, 1, \ldots, K$

$$x_{k+1} = x_k - D^{-1}(Ax_k - g).$$

We note that the stopping criterion (11) is equivalent to

$$\|x_{K+1} - x_K\|_2 \leqslant \delta \|x_1\|_2. \tag{12}$$

The stopping criteria (12) is commonly used by scientists.

5.2. GMRES with block diagonal preconditioning

The matrices B and C are defined as in Section 5.1 We then apply GMRES to the problem $D^{-1}Ax = D^{-1}g$. As a stopping criteria for the GMRES method we use (11), i.e.,

$$\|D^{-1}(Ax_K - g)\|_2 \leqslant \delta \|D^{-1}g\|_2.$$

5.3. The RAS and AS with minimun overlap and block diagonal preconditioning

We now describe a version of the RAS preconditioner, that was recently introduced in [2], but with the smallest possible non-zero overlap and with a left block diagonal preconditioning. We consider a sparse linear system

$$\tilde{A}x = D^{-1}Ax = D^{-1}g, \tag{13}$$

where A is an $n \times n$ nonsingular sparse matrix obtained by discretizing a system of partial differential equations, such as (9), on a tetrahedral mesh $\mathscr{M} = \{K_i, i = 1, \ldots, M\}$, where K_i are the tetrahedra. Using an element-based partitioning, $\mathscr{M}$ can be decomposed into N nonoverlapping sets of elements, or equivalently into N overlapping sets of nodes (since tetrahedra in different subsets may share the same nodes). Let us denote the node sets as $W_i, i = 1, \ldots, N$. Let W be the set of all the nodes, then we say that the node-based partition

$$W = \bigcup_{i=1}^{N} W_i$$

is a minimum overlap partition of W. Here "minimum" refers to the fact that the corresponding element-based partition has zero overlap. The nodes belonging to more than one subdomain are called interface nodes. To obtain a node-based nonoverlapping partition, we identify a unique subdomain as the sole owner of each interface node. This leads to a node-based nonoverlapping partition of W, as shown in Fig. 2 for a two-dimensional mesh, or more precisely $W_i^{(0)} \subset W_i$, and

$$\bigcup_{i=1}^{N} W_i^{(0)} = W \quad \text{and} \quad W_i^{(0)} \cap W_j^{(0)} = \emptyset \text{ for } i \neq j.$$

Let m be the total number of nodes in W. Associated with each W_i^0 we define a restriction operator R_i^0. In matrix terms, R_i^0 is an $m \times m$ block-sub-identity matrix whose diagonal blocks are set to $I_{5\times5}$ if the corresponding node belongs to W_i^0 and to a 5×5 zero block otherwise. Similarly we can define R_i for each W_i. Note that both R_i^0 and R_i are of size $n \times n$. With this we define the matrix,

$$\tilde{A}_i = R_i \tilde{A} R_i.$$

Note that although $\tilde{A}_i$ is not invertible, we can invert its restriction to the subspace

$$\tilde{A}_i^{-1} \equiv \left((\tilde{A}_i)_{|L_i} \right)^{-1},$$

where L_i is the vector space spanned by the set W_i in $\mathbb{R}^n$.

The RAS preconditioner is defined by

$$C_{\mathrm{RAS}} \equiv R_1 \tilde{A}_1^{-1} R_1^0 + \cdots + R_N \tilde{A}_N^{-1} R_N^0.$$

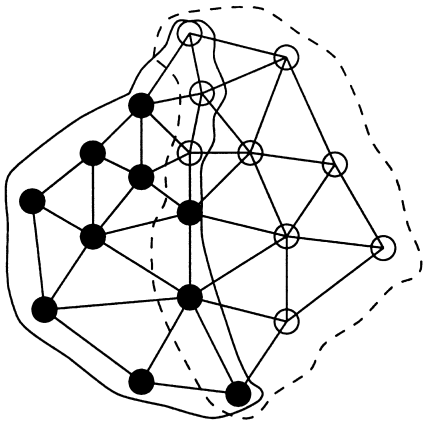

Fig. 2. A minimum overlap two-subdomain partition. $W_1^{(0)}$ contains all the '•' nodes, and $W_2^{(0)}$ contains all the '○' nodes, therefore $W_1^{(0)} \cap W_2^{(0)} = \emptyset$. $W_1^{(1)}$ contains all the nodes bounded inside the solid curve, and $W_2^{(1)}$ contains all the nodes bounded inside the dotted curve.

We recall that the additive Schwarz (AS) preconditioner [7,19] is defined by

$$C_{\mathrm{AS}} = R_1\tilde{A}_1^{-1}R_1 + \cdots + R_N\tilde{A}_N^{-1}R_N.$$

Our GMRES/RAS algorithm can be simply described as follows: obtain an approximate solution for $x = C_{\mathrm{RAS}}\eta$ by solving the right-preconditioned system

$$\tilde{A}C_{\mathrm{RAS}}\eta = D^{-1}g$$

with a GMRES method. As a stopping criteria, we use that

$$\|\tilde{A}C_{\mathrm{RAS}}\eta_K - D^{-1}g\|_2 \leqslant \delta\|D^{-1}g\|_2.$$

In the numerical experiments to be reported in Section 6, all subdomain problems are solved with ILU(0) and GMRES with a restart dimension equals to five. We remark that the action of R_i^0 to a vector involve less communication in a parallel implementation than R_i does. As a result, RAS is cheaper than AS in terms of the communication cost. We will show in the numerical experiments that RAS is in fact also cheaper than AS in terms of iteration counts and CPU time.

6. Numerical results

We implement the investigated algorithms on two parallel machines, and the top-level message-passing calls are implemented through MPI [13]. We partition the mesh by using the TOP/DOMDEC package [10]. We require that all subdomains have more or less the same number of mesh points. An effort is made to reduce the number of mesh points along the interfaces of the subdomains to reduce interprocessor communication cost. The mesh generation and partitioning steps are considered as pre-processing steps, and therefore not accounted for in the CPU reporting. The sparse matrix A is constructed at every time step and stored in an edge-based sparse format. In order to save CPU time on factorization, the local sparse matrices $\tilde{A}_i$ are constructed and factorized at every other time step. The $\tilde{A}_i$ are stored in 5×5 block diagonal compressed row format.

We consider the simulation of an Euler flow around the flexible AGARD Wing 445.6 [20] (Fig. 3) set in a prescribed motion. The wing is forced to vibrate along its fundamental flexible mode shape X_W^1 with a constant circular frequency ω and an amplitude a. Hence, the position X_B of the fluid points lying on the surface of the wing is forced into the harmonic motion

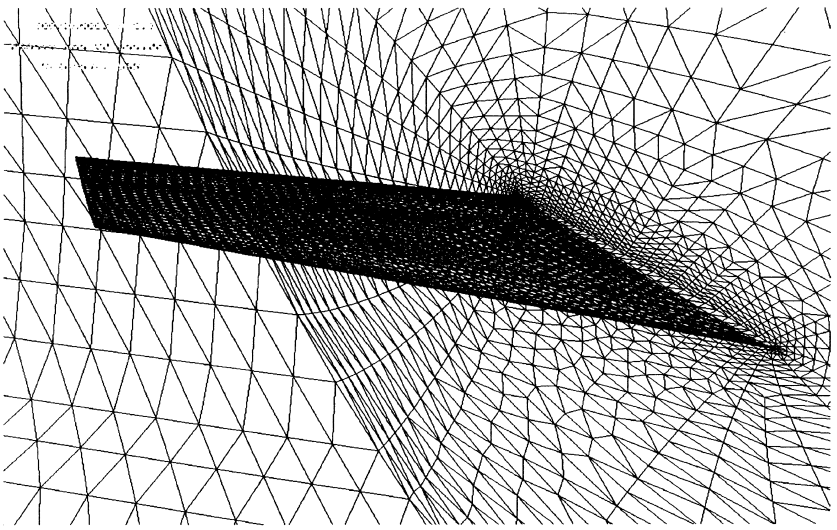

Fig. 3. Partial view of the CFD mesh.

$$X_B = X_W^0 + a(X_W^1 - X_W^0)\sin \omega t,$$

where X_W^0 denotes the initial shape of the wing. The amplitude a is chosen such that the maximum vertical deflection of the wing is equal to 3% of the wingspan. The circular frequency is set to $\omega = 95$. Note that the associated frequency is $f = 15$ Hz which corresponds to the first torsional mode of a realistic aircraft wing.

The CFL number is chosen such that the time-step Δt satisfies $\Delta t = T/30$, where $T = 1/f$ denotes the period of oscillation. This time-step is typical of a second order implicit time-integration scheme which preserves the GCL.

We first discretize the computational domain around the Agard Wing using two three-dimensional unstructured tetrahedral meshes with 22 014 and 331 233 nodes.

We run our code for 100 time steps, where in each time step we solve two linear systems. We focus on the average performance of the algorithms for solving a single linear system. The CPU time reported includes all the computation related to solving the linear systems including the factorizations. The results on the coarser grid are summarized in Table 1. Tables 2 and 3 are for the finer mesh. Due to the special choice of the CFL numbers the time steps for the two test cases are roughly the same. Comparing the RAS columns in Tables 1–3 we see that there is little dependence on the mesh sizes. However, we observe clearly that Block Jacobi (BJAC) and GMRES with block diagonal preconditioning (BGMRES) have a strong dependence on the mesh sizes. For the Agard Wing with 331233 nodes, the RAS algorithm is roughly three

Table 1
Iteration counts (CPU time in seconds). Euler flow passing an oscillating wing at Mach 0.89. The mesh contains 22 014 nodes. subd is the number of subdomains and δ is the stopping condition. Performed on a 4-, 8- and 16-processors SGI Origen 2000

subd	$\delta = 10^{-2}$			
	BJAC	BGMRES	AS	RAS
4	41 (3.23)	27 (2.78)	13 (2.63)	10 (2.09)
8	41 (1.61)	27 (1.44)	13 (1.36)	10 (1.05)
16	41 (0.82)	27 (0.80)	14 (0.71)	10 (0.52)

Table 2
Iteration counts (CPU time in seconds). Euler flow passing an oscillating wing at Mach 0.89. The mesh contains 331 233 nodes. subd is the number of subdomains and δ is the stopping condition. Performed on a 4-, 8- and 16-processors SGI Origen 2000

subd	$\delta = 10^{-2}$			
	BJAC	BGMRES	AS	RAS
4	81 (162)	57 (156)	13 (68.3)	10 (57.2)
8	81 (80.3)	57 (79.1)	13 (35.7)	11 (29.7)
16	81 (40.6)	57 (41.1)	14 (21.5)	11 (16.9)

Table 3
Iteration counts (ITER), computational time (COMP), and communication time (COMM) in seconds. Euler flow passing an oscillating wing at Mach 0.89. The mesh contains $n = 331\,233$ nodes. subd is the number of subdomains δ is the stopping condition. Performed on a 40-processors SP2

subd = 40	$\delta = 10^{-2}$				$\delta = 10^{-8}$			
	BJAC	BGMRES	AS	RAS	BJAC	BGMRES	AS	RAS
ITER	81	57	15	11	444	312	78	58
COMP	7.7	6.7	3.9	2.9	40.2	38.1	17.1	12.7
COMM	0.65	0.80	0.49	0.35	2.4	2.5	1.6	1.1

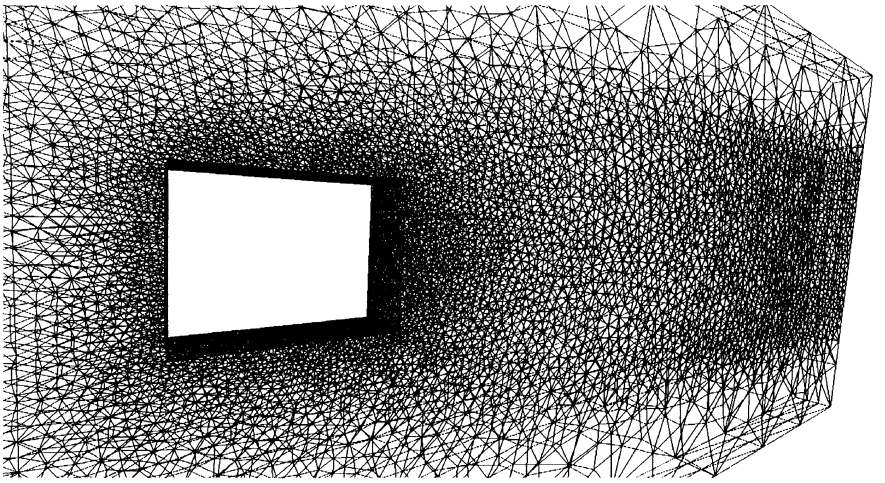

Fig. 4. Three-dimensional discretization of the computational mesh (partial view).

Table 4
Iteration counts (CPU time in seconds). Low-speed flow passing a square cylinder at Mach 0.10. The mesh contains 43 154 nodes. subd is the number of subdomains and δ is the stopping condition. Performed on a 4-, 8- and 16-processors SGI Origen 2000

subd	$\delta = 10^{-2}$			
	BJAC	BGMRES	AS	RAS
4	88 (12.5)	57 (11.9)	20 (7.88)	17 (6.81)
8	88 (6.24)	57 (6.09)	22 (4.34)	18 (3.58)
16	88 (3.11)	57 (3.10)	23 (2.28)	19 (1.82)

times faster than BJAC and BGMRES for different stopping conditions δ. We expect that RAS will perform even better for larger meshes. As the number of subdomains grows from 4 to 16 or 40, the number of iterations of RAS stays roughly constant even though the preconditioner lacks a coarse space. Another observation is that RAS requires 20–30% fewer number of iterations than AS for the test cases. Some of the CPU timings were obtained on a 4-, 8- and 16-processors SGI Origin 2000. Even though this is a shared memory machine, we still trea t it as a message-passing machine.

We next investigate the behavior of the previous linear solvers for the simulation of vortex shedding flows by solving Navier–Stokes equations equipped with a k–ϵ turbulence model and a wall function [6]. This turbulence model is popular in the engineering community. We consider the three-dimensional numerical simulation of the low-speed flow past a square cylinder using unstructured mesh with 43 154 nodes (Fig. 4). The cylinder has a 1 cm $\times$ 1 cm cross section. The far-field flow is assumed to be uniform. The free-stream Mach number is $M_\infty = 0.1$, and the Reynolds number is $Re = 22\,000$. We select a time-stepping strategy that corresponds to sampling the captured vortex shedding in 100 times-steps.

Comparing the columns in Table 4 we conclude that the RAS algorithm is 20% faster than ASM and 80% faster than BJAC and BGMRES. Also, we see that the number of iterations of RAS stays roughly constant as the number of subdomains grows from 4 to 16.

7. Concluding remarks

We studied the performance of a newly introduced RAS preconditioner and tested it in transonic flow calculations over an oscillating wing and low-speed vortex shedding flow calculations. A scaled GMRES/RAS with minimun overlap compares very favorably against traditional methods in terms of iteration

counts, CPU time and communication time when implemented on a parallel computer. Even though we the RAS method do not have a coarse space, the number of iterations is nearly independent of the number of subdomains for all the test cases.

References

[1] K. Bohmer, P. Hemker, H. Stetter, The defect correction approach, Comput. Supp. 5 (1984) 1–32.
[2] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput., 21 (1999) 792–797.
[3] J.-A Désidéri, P.W. Hemker, Convergence analysis of the defect-correction iteration for hyperbolic problems, SIAM J. Sci. Comput. 16 (1995) 88–118.
[4] B. Koobus, C. Farhat, Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes, Comput. Meth. Appl. Mech. Engrg. 170 (1999) 103–129. Also published as AIAA Paper 98-0113, 36th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 12–15 January 1998.
[5] B. Koobus, C. Farhat, Time-accurate algorithms for the evaluation of the viscous fluxes associated with unsteady flows and unstructured moving meshes, AIAA Paper 96–2384, 14th AIAA Applied Aerodynamics Conference, New Orleans, Louisiana, 18–20 June 1996.
[6] H. Tran, B. Koobus, C. Farhat, Numerical simulation of vortex shedding flows past moving obstacles using the $k-\epsilon$ turbulence model on unstructured dynamic meshes, La Revue Européenne des Eléments Finis 6 (5/6) (1998) 611–642.
[7] M. Dryja, O.B. Widlund, Domain decomposition algorithms with small overlap, SIAM J. Sci. Comput. 15 (1994) 604–620.
[8] C. Farhat, L. Fezoui, S. Lantéri, Two-dimensional viscous flow computations on the connection machine: unstructured meshes upwind schemes and massively parallel computations, Comput. Meth. Appl. Mech. Engrg. 102 (1993) 61–88.
[9] C. Farhat, S. Lantéri, Simulation of compressible viscous flows on a variety of mpps: computational algorithms for unstructured dynamic meshes and performance results, Comput. Meth. Appl. Mech. Engrg. 119 (1994) 35–60.
[10] C. Farhat, S. Lanteri, H. Simon, TOP/DOMDEC: a software tool for mesh partitioning and parallel processing and applications to CSM and CFD computations, Comput. Sys. Engrg. 46 (1995) 13–26.
[11] C. Farhat, M. Lesoinne, N. Maman, Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation geometric conservation and distributed solution, Internat. J. Numer. Meth. Fluids 21 (1995) 807–835.
[12] L. Fezoui, B. Stoufflet, A class of implicit upwind schemes for euler simulations with unstructured meshes, J. Comp. Phys. 84 (1989) 174–206.
[13] W. Gropp, E. Lusk, A. Skjellum, Using MPI – Portable Parallel Programming with the Message-Passing Interface, The MIT Press, Cambridge, 1995.
[14] M. Lesoinne, C. Farhat, Geometric conservation laws for aeroelastic computations using unstructured dynamic meshes, AIAA Paper 95–1709, in: 12th AIAA Computational Fluid Dynamics Conference, San Diego, 19–22 June 1995.
[15] M. Lesoinne, C. Farhat, Geometric conservation laws for flow problems with moving boundaries and deformable meshes and their impact on aeroelastic computations, Comput. Meth. Appl. Mech. Engrg. 134 (1996) 71–90.
[16] R. Martin, H. Guillard, A second-order defect correction scheme for unsteady problems, Comput. Fluids 25 (1996) 9–27.
[17] B. Nkonga, H. Guillard, Godunov type method on nonstructured meshes for three-dimensional moving boundary problems, Comp. Meth. Appl. Mech. Engrg. 113 (1994) 183–204.
[18] Y. Saad, M. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1996) 856–869.
[19] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial, Cambridge University Press, Cambridge, 1996.
[20] E.C. Yates, AGARD Standard Aeroelastic Configuration for Dynamiic Response, Candidate Configuration I. -Wing 445.6, NASA, TM-100492, 1987.