Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Performance Analysis of Parallel Schwarz Preconditioners in the LES of Turbulent Channel Flows

Pasqua D'Ambra, Daniela di Serafino, Salvatore Filippone

RT-ICAR-NA-2011-02 **Dicembre 2011**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Performance Analysis of Parallel Schwarz Preconditioners in the LES of Turbulent Channel Flows*

Pasqua D'Ambra[1], *Daniela di Serafino*[2], *Salvatore Filippone*[3]

Rapporto Tecnico N: **RT-ICAR-NA-2011-02** ***Data:*** **Dicembre 2011**

* Preprint sottomesso a Computers and Mathematics with Applications.

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, via P. Castellino, 111, 80131 Napoli.
[2] Dipartimento di Matematica, Seconda Università di Napoli, Viale Lincoln, 5, 81100 Caserta and ICAR-CNR, Napoli.
[3] Dipartimento di Ingegneria Meccanica, Università di Roma "Tor Vergata", v.le del Policlinico, 1, 00133 Roma.

Performance Analysis of Parallel Schwarz Preconditioners in the LES of Turbulent Channel Flows

Pasqua D'Ambra[a,*], Daniela di Serafino[b], Salvatore Filippone[c]

[a] *Institute for High-Performance Computing and Networking (ICAR), CNR, Via P. Castellino 111, I-80131 Naples, Italy.*
[b] *Department of Mathematics, Second University of Naples, Viale A. Lincoln 5, I-81100 Caserta, Italy, and ICAR-CNR.*
[c] *Department of Mechanical Engineering, University of Rome "Tor Vergata", Viale del Politecnico 1, I-00133, Rome, Italy.*

Abstract

We present a comparative study of parallel Schwarz preconditioners in the solution of linear systems arising in a Large Eddy Simulation (LES) procedure for turbulent plane channel flows. This procedure applies a time-splitting technique to suitably filtered Navier-Stokes equations, in order to decouple the continuity and momentum equations, and uses a semi-implicit scheme for time integration and finite volumes for space discretization. This approach requires the solution of four sparse linear systems at each time step, which accounts for a large part of the overall simulation and hence is a crucial task in the whole procedure. Several preconditioners are applied in the simulation of a reference test case for the LES community, using discretization grids of different sizes, with the aim of analysing the effects of different algorithmic choices defining the preconditioners, and identifying the most effective ones for the selected problem. The preconditioners, coupled with the GMRES method, are run within SParC-LES, a recently developed LES code based on the PSBLAS and MLD2P4 libraries for parallel sparse matrix computations and preconditioning.

Keywords: Schwarz Preconditioners, Large Eddy Simulation, Parallel Software.
2010 MSC: 65F08, 65Y05, 76F65

1. Introduction

Large Eddy Simulation (LES) is a widely used approach for detailed study of turbulent flows in small/medium-scale applications. Although it has a lower

*Corresponding author.
Email addresses: pasqua.dambra@cnr.it (Pasqua D'Ambra), daniela.diserafino@unina2.it (Daniela di Serafino), salvatore.filippone@uniroma2.it (Salvatore Filippone)

computational cost than Direct Numerical Simulation (DNS), its application to realistic flows remains a computationally intensive procedure. In this work we focus on large and sparse linear systems arising in a LES procedure for turbulent wall-bounded flows, and analyse the performance of parallel Schwarz preconditioners in solving these systems by GMRES [1]. The linear systems stem from the application of a time-splitting technique to suitably filtered Navier-Stokes equations, to decouple the continuity and momentum equations, and from a finite-volume discretization of the resulting equations (see the next section for details). Their solution accounts for a significant part of the overall computational effort, thus the use of efficient preconditioners is critical for the efficiency of the overall simulation.

In our analysis we use the preconditioners implemented in the Multilevel Domain Decomposition Parallel Preconditioners Package based on PSBLAS (MLD2P4) [2], coupled with the GMRES solver from the Parallel Sparse BLAS (PSBLAS) library [3]. The solution of the systems is performed within SParC-LES [4], a parallel code recently developed for the simulation of turbulent channel flows, which is run on a classical test case used as benchmark in the Italian LES community [5]. This study differs from previous work in [6, 7] because it provides a more detailed analysis of several preconditioners for all the systems arising in the LES procedure, applied to a widely used reference test case. It is also worth noting that the results discussed in this paper guided our choice of the preconditioners in a complete simuation of the selected test case with the SParC-LES code (see [4]).

The paper is organized as follows. In Section 2 we briefly outline the LES approach implemented in SParC-LES, with the aim of introducing the abovementioned linear systems. In Section 3 we give a very brief overview of the preconditioners implemented in MLD2P4. Finally, in Section 4 we present our analysis of the preconditioners. We also provide some conclusions, in Section 5.

2. Sparse linear systems in the LES of turbulent channel flows

We consider the approach proposed in [8, 9] for simulating an incompressible and homothermal flow in a plane channel. The LES governing equations are obtained by applying a top-hat filter coupled with a differential deconvolution operator to the Navier-Stokes (N-S) equations in non-dimensional weak conservation form. Periodic boundary conditions are prescribed in the streamwise (x) and spanwise (z) directions, and no-slip conditions on the walls (y). The numerical solution of the filtered N-S equations is based on a time-splitting technique, using an approximate projection method, to decouple the continuity equation from the momentum equation. Thus, at the n-th time step, the unknown velocity field $\widetilde{\mathbf{v}}^n$ is computed through the following predictor-corrector formula:

$$\widetilde{\mathbf{v}}^n = \mathbf{v}^* - \Delta t \nabla \phi,$$

where $\mathbf{v}^*$ is an intermediate velocity field, Δt is the time step, and ϕ is a suitable scalar field.

The intermediate velocity $\mathbf{v}^*$ is computed by solving a modified momentum equation, where the pressure term is neglected, with Dirichlet boundary conditions at the walls. By appling a second-order Adams-Bashforth/Crank-Nicolson semi-implicit scheme, the following equation is obtained:

$$\left(A_{\mathbf{x}}^{-1} - \frac{\Delta t}{2Re} D_2\right) \mathbf{v}^* = \left(A_{\mathbf{x}}^{-1} + \frac{\Delta t}{2Re} D_2\right) \widetilde{\mathbf{v}}^{n-1} + \frac{\Delta t}{2} \left(3 \left(\frac{1}{Re}(D_1 + D_3)\widetilde{\mathbf{v}}^{n-1} + \mathbf{f}_{conv}^{n-1}\right) - \left(\frac{1}{Re}(D_1 + D_3)\widetilde{\mathbf{v}}^{n-2} + \mathbf{f}_{conv}^{n-2}\right)\right), \tag{1}$$

where $A_{\mathbf{x}}$ is the differential deconvolution operator, D_1, D_2 and D_3 are suitable diffusion operators, $\mathbf{f}_{conv}$ is the so-called convective flux, Re is the Reynolds number, and the superscripts $n-1$ and $n-2$ indicate that a quantity is computed at the $(n-1)$-th and $(n-2)$-th time steps, respectively.

The scalar field ϕ is obtained by solving the following Poisson-like equation:

$$(D_1 + D_2 + D_3)\phi \;=\; \frac{1}{\Delta t \left|\Omega\left(\mathbf{x}\right)\right|} \int\limits_{\partial\Omega(\mathbf{x})} \mathbf{v}^* \cdot \mathbf{n} \; dS \;, \tag{2}$$

where $\Omega(\mathbf{x})$ is a finite volume contained into the region of the flow and $\mathbf{n}$ is the outward-oriented unit vector normal to $\partial\Omega(\mathbf{x})$. Neumann boundary conditions are prescribed in the wall-normal direction, which satisfy the compatibility conditions and hence ensure the existence of a solution that is unique up to an additive constant. Note that (2) is also known as pressure equation, since $\nabla\phi$ provides an $O(\Delta t)$ approximation of the pressure gradient.

The spatial discretization of the equations is obtained by using a structured Cartesian grid, with uniform spacings in the streamwise and spanwise directions, where the flow is assumed to be homogeneous, and non-uniform grid spacing with refinement near the walls in the y direction, to suitably describe the boundary layer. The equations are discretized by using a finite volume method, with flow variables co-located at the centres of the control volumes.

In equation (1), a third-order multidimensional upwind scheme is used for the convective fluxes, and a classical second-order central scheme for the diffusive ones; a fourth-order central scheme is applied for the discretization of the inverse deconvolution operator. The discrete deconvolved momentum equation consists of three linear systems:

$$A_v v_r^* = w_r, \quad r = 1, 2, 3, \tag{3}$$

where v_r is the discrete approximation of the component of $\mathbf{v}^*$ along the r-th coordinate axis. Henceforth, these systems are referred to as velocity systems. The linear equation corresponding to a grid point (i, j, k) that is two cells away from the boundary of the computational domain takes the following form:

$$\begin{aligned}\rho_j(v_r^*)_{i,j,k} &+ \sigma\,((v_r^*)_{i-1,j,k} + (v_r^*)_{i+1,j,k} + (v_r^*)_{i,j,k-1} + (v_r^*)_{i,j,k+1}\,)\\ &+ \tau\,((v_r^*)_{i-2,j,k} + (v_r^*)_{i+2,j,k} + (v_r^*)_{i,j,k-2} + (v_r^*)_{i,j,k+2}\,)\\ &+ \rho_{j-2}(v_r^*)_{i,j-2,k} + \rho_{j-1}(v_r^*)_{i,j-1,k} + \rho_{j+1}(v_r^*)_{i,j+1,k} + \rho_{j+2}(v_r^*)_{i,j+2,k}\\ &= (w_r)_{i,j,k},\end{aligned}$$

where the coefficients σ and τ are independent of (i, j, k), and the coefficient ρ_{j+s} $(s = -2, -1, 1, 2)$ depends on j as well as on the Reynolds number. Some changes are required in the equations corresponding to the remaining grid cells. The matrix A_v is unsymmetric, but has a symmetric sparsity pattern, with 13 nonadjacent nonzero diagonals; it is diagonally dominant and well conditioned. Its dimension is equal to the total numer of grid cells. Furthermore, A_v does not depend on the time step, therefore it can be computed just once, at the beginning of the LES procedure. Because of the features of A_v, we expect that GMRES with a simple preconditioner is very effective on (3).

Equation (2) is discretized by using a classical second-order central scheme, obtaining a linear system

$$A_p\varphi = g, \tag{4}$$

where φ denotes the discrete approximation of ϕ. In the following, this system is called pressure system. As in the previous case, A_p has dimension equal to the number of grid cells, is unsymmetrix, but with symmetric sparsity pattern, and is independent of the time step. Furthermore, it has at most seven entries per row, distributed over seven nonadjacent diagonals plus four diagonals arising from the periodicity in the x and z directions. A_p is singular, but its null space and range space have trivial intersection, so the GMRES method can compute a solution of (4) before a breakdown occurs [10]. The linear equation corresponding to the the grid cell (i, j, k) can be written as

$$\begin{aligned}\alpha_j\phi_{i,j,k} + \beta\,(\phi_{i+1,j,k} + \phi_{i-1,j,k}) + \gamma\,(\phi_{i,j,k+1} + \phi_{i,j,k-1})&\\ + \delta_j\phi_{i,j+1,k} + \eta_j\phi_{i,j-1,k} = g_{i,j,k},&\end{aligned}$$

where β and γ are constant, and α_j, δ_j and η_j depend on the cell grid spacing in the y direction. This equation is suitably modified near the boundaries to accomplish the non-homogeneous Neumann conditions; obvious modifications of the indices i and k are also applied to take into account the periodicity in the streamwise and spanwise directions. Taking into account that system (4) arises from the discretization of an elliptic equation, we expect that multilevel Schwarz preconditioners, coupled with GMRES, are effective in its solution. However, several algorithmic choices must be done to choose a specific preconditioner, with the aim of getting the best tradeoff between effectivenes and parallel performance.

For further details on the LES model and its discretization we refer to [8, 9].

3. Algebraic one-level and multilevel Schwarz preconditioners

We provide a short description of one-level and multilevel Schwarz preconditioners, to better understand the comparative analysis performed in the next section. We consider algebraic preconditioners, which are built by using information coming from the matrix to be preconditioned, but do not assume any specific knowledge of the discretization grid. This makes the implementation of the preconditioners independent of a particular application, thus providing general and flexible tools; furthemore, on "standard" elliptic test problems, the algebraic preconditioners are able to achieve results close to that of geometric methods, which explicitly use information from the discretization of the problem.

Roughly speaking, one-level Schwarz preconditioners are domain decomposition methods based on the idea of dividing into submatrices the matrix A to be preconditioned, solving a linear system involving each submatrix, and combining the local solutions for building a preconditioner for A. Among them, we consider the additive Schwarz (AS) preconditioners, which are well suited to parallel computation. In this case, the number of submatrices usually matches the number of available processing units. The submatrices usually overlap, i.e., they have some common rows; the overlap level is defined recursively, by using the adjacency graph of A. More precisely, a submatrix A_i^{δ} with overlap δ is obtained by extending a submatrix $A_i^{\delta-1}$ with the entries that are adjacent to $A_i^{\delta-1}$ in the sense of the graph of A; the overlap 0 indicates that the submatrices do not overlap. The well-known block-Jacobi preconditioner is a special case of the AS preconditioners, corresponding to the overlap 0. Some variants of the AS methods have been developed; the most effective one in terms of both convergence behaviour and parallel performance is the so-called restricted additive Schwarz (RAS) [11, 12]. The theory states that the effectiveness of the preconditioners generally improves as the overlap increases; on the other hand, larger overlaps require more data movements on parallel computers. Unfortunately, the convergence rate of iterative solvers coupled with AS preconditioners deteriorates as the number of submatrices increases, and a coarse-level correction must be added to introduce a global coupling such that the number of iterations is bounded independently of the submatrices (see, e.g., [13]).

The recursive application of AS preconditioners and coarse-level corrections generates multilevel Schwarz preconditioners. In other words, these preconditioners are obtained by applying basic Schwarz methods to a hierarchy of matrices that represent the matrix A in increasingly coarser spaces, and by combining the contributions coming from each space. In this context, the one-level Schwarz preconditioners applied to each matrix of the hierarchy play the role of smoothers, as in the multigrid methods.

Here we refer to the multilevel Schwarz preconditioners implemented in the MLD2P4 package [2]. In this case, the hierarchy of coarser matrices is built by using the smoothed aggregation technique [14], briefly described next. Given the set $\Omega^l = \{1, 2, \ldots, n_l\}$ of row (column) indices of the matrix A^l at a certain level l of the hierarchy, the smoothed aggregation generates a coarser index

set $\Omega^{l+1} = \{1, 2, \ldots, n_{l+1}\}$ by suitably grouping strongly coupled indices into disjoint subsets called aggregates. Two indices $i, j \in \Omega^l$ are considered strongly coupled if

$$|a_{ij}^l| \geq \epsilon \sqrt{|a_{ii}^l a_{jj}^l|},$$

where $\epsilon > 0$ is a given threshold. The first matrix of the hierachy is $A^1 = A$, which is called the finest matrix. In MLD2P4, a parallel decoupled aggregation is implemented, where each processor generates coarser index spaces by using only the local indices assigned to it. This approach has the advantage of being embarrassingly parallel, but the quality and the dimension of the resulting coarse matrices are dependent on the initial data distribution and on the number of processors. Work is in progress for developing new aggregation schemes to be included in MLD2P4 with the aim of overcoming the above limitation [15]. Once the coarse index set at level $l+1$ has been built, a tentative prolongator $\widetilde{P}^l \in \Re^{n_l \times n_{l+1}}$ is constructed, where each column identifies an aggregate. Generally $\widetilde{P}^k$ is defined as a piecewise constant interpolation operator whose range includes the space spanned by the vector of all ones. A damped Jacobi smoother is applied to $\widetilde{P}^l$ to obtain the actual prolongator P^l. The coarse matrix A^{l+1} is built by using the Galerkin approach, i.e.,

$$A^{l+1} = R^l A^l P^l,$$

where $R^l = (P^l)^T$. Convergence results concerning multilevel methods based on the previous aggregation strategy confirm the effectiveness of this approach for symmetric positive definite matrices [16]. A Petrov-Galerkin generalization of the above smoothed aggregation algorithm is also available in MLD2P4. This technique was proposed in [17] with the aim of achieving the effectiveness of the symmetric positive definite case in the case of highly nonsymmetric systems, such as those arising in advection-dominated flows.

Different combinations of the AS smoothers with the coarse matrices and relevant transfer operators may be considered, resulting in different multilevel preconditioners. A discussion on this issue is beyond the scope of this paper; for details the reader is referred to [13]. To give an example, in Fig. 1 we report the algorithm for the application of a V-cycle preconditioner to the system $A^l u^l = b^l$. Here L corresponds to the coarsest level and M_{AS} is a one-level AS preconditioner. This is the algorithm used in the experiments discussed in the next section.

A key issue to achieve efficient parallel multilevel Schwarz preconditioners is finding a suitable combination of the number of levels with the coarsest system solver, for the problem and the machine under consideration. In particular, a tradeoff between the accuracy and cost of the coarsest-level correction must be found to achieve a significant time reduction; the cost for the application of the AS smoother must be also taken into account.

Many variants of multilevel Schwarz preconditioners are available in MLD2P4, as explained in [18]. The simple Jacobi and Gauss-Seidel methods are also provided as smoothers and coarsest-level solvers.

V-cycle$\left(l, A^l, b^l, u^l\right)$

if $(l \neq L)$ then

$u^l = u^l + M^l_{AS}\left(b^l - A^l u^l\right)$

$b^{l+1} = R^{l+1}\left(b^l - A^l u^l\right)$

$u^{l+1} =$**V-cycle**$\left(l+1, A^{l+1}, b^{l+1}, 0\right)$

$u^l = u^l + P^{l+1} u^{l+1}$

$u^l = u^l + M^l_{AS}\left(b^l - A^l u^l\right)$

else

$u^l = \left(A^l\right)^{-1} b^l$

endif

return u^l

Figure 1: V-cycle algorithm.

4. Performance analysis

We analyse the performance of parallel preconditioners from the MLD2P4 library, coupled with the GMRES solver implemented in PSBLAS, in the simulation of a turbulent flow in a plane channel at $\mathrm{Re}_\tau = 590$. This is a classical test case for wall-bounded flows, for which DNS data are also available [19]. The height of the channel is 2δ, while the streamwise and spanwise lengths are $2\pi\delta$ and $\pi\delta$, respectively; these lengths are made non-dimensional by using the channel half-width δ. The turbulent flow, assumed to be periodic along x and y, is sustained in the x direction by a forcing pressure gradient, which is kept constant; an initial parabolic velocity profile with a superimposed Gaussian perturbation field is assigned along the streamwise direction.

This problem has been discretized using the two Cartesian grids proposed in [5] as reference grids for large-eddy simulations of the selected test problem. Both grids have $N_x = N_z = 64$ equally spaced nodes in the streamwise and spanwise directions, corresponding (approximately) to the grid spacings $\Delta x = 0.098$ and $\Delta z = 0.049$; they use a different number of nodes, N_y, in the wall-normal direction, distributed according to a trigonometric stretching law. For one of the grids, named Grid 1, $N_y = 32$ and the grid spacing varies from $\Delta y_{min} = 0.0025$ to $\Delta y_{max} = 0.1$, so that the boundary layer is not resolved. For the other grid, named Grid 2, $N_y = 100$, $\Delta y_{min} = 0.00025$ and $\Delta y_{max} = 0.03$, thus allowing to resolve the boundary layer. The corresponding dimensions of the matrices of the velocity and pressure systems are $N = 126976$ for Grid 1, and $N = 405504$ for Grid 2. A third grid (Grid 3), with N_y defined as in Grid 2 and $N_x = N_z = 100$ has been also considered to better analyse the performance of the preconditioners as the size of the matrix increases. In this case $N = 990000$. The time step used in the simulation is $\Delta t = 10^{-5}$. The grids

have been partitioned according to a 3D block decomposition; each subgrid has been assigned to a process in a virtual 3D Cartesian topology, where each process matches an available processor. In all the tests, only 1000 time steps have been considered, because we have previously observed that the behaviour of the linear solvers in this time interval is representative of the behaviour in a complete simulation, which requires about 10^7 time steps.

For each type of system, GMRES has been coupled with different preconditioners, chosen according to the characteristics of the system itself (actually, restarted GMRES, with restarting parameter equal to 30, has been applied, but the number of iterations is generally lower than 30). A reduction of the 2-norm of the residual by a factor of 10^{-7} has been used as stopping criterion. For each velocity or pressure system, the solution obtained at the previous time step has been chosen as starting guess, except at the first time step where the zero vector has been considered.

All the experiments have been carried out on a HP XC 6000 Linux cluster with Intel Itanium Madison bi-processor nodes, operated by the Naples branch of ICAR-CNR, using from 1 to 64 processors. This machine has been equipped with PSBLAS 2.4.0, MLD2P4 1.2.1., HP MPI 2.01, and the GNU 4.6.1 Fortran compiler.

4.1. Preconditioners for the velocity systems

GMRES has been applied to the velocity systems with the simplest Schwarz preconditioner, i.e. block-Jacobi, as well as with the Jacobi (i.e., diagonal) preconditioner and without any preconditioner. The ILU(0) factorization has been applied to the blocks in the block-Jacobi method. Other preconditioners have not been tested because the matrix of the velocity systems is well conditioned and diagonally dominant.

The results obtained on Grid 1 and Grid 2 are shown in Tab. 1. For all the runs we report the mean time (in seconds) for solving a single velocity system and, in brackets, the average number of iterations, rounded to the nearest integer. The time for building the Jacobi and block-Jacobi preconditioners is neglected, since the matrix of the velocity systems does not change during the whole simulation and hence the preconditioner must be built only once. For both grids we see that the number of iterations with no preconditioner is small and no gain is obtained by using the Jacobi preconditioner (surprisingly, the latter produces an increase of 1 in the mean number of iterations). As expected, a reduction of the iterations is achieved with the block-Jacobi preconditioner; nevertheless, the smallest execution time is obtained without preconditioner, except on 1 and 2 processors (on 1 processor the preconditioner is ILU(0)). However the execution times corresponding to the three cases get closer as the number of processors increase.

Fig. 2 shows the speedups in the solution of the velocity systems, computed by using the mean execution time. As expected, the best speedups are obtained with no preconditioner and with the Jacobi one; on 64 processsors, they are about 30 for Grid 1, and more than 41 for Grid 2. On 64 processors, the

GRID 1			
NP	NOPREC	JAC	BJAC
1	0.742 (10)	0.814 (11)	0.427 (4)
2	0.392 (10)	0.423 (11)	0.339 (5)
4	0.197 (10)	0.211 (11)	0.208 (6)
8	0.104 (10)	0.111 (11)	0.127 (7)
16	0.063 (10)	0.067 (11)	0.070 (7)
32	0.035 (10)	0.037 (11)	0.038 (7)
64	0.025 (10)	0.027 (11)	0.025 (7)

GRID 2			
NP	NOPREC	JAC	BJAC
1	2.365 (9)	2.572 (10)	1.483 (4)
2	1.222 (9)	1.312 (10)	1.102 (6)
4	0.673 (9)	0.723 (10)	0.678 (6)
8	0.324 (9)	0.357 (10)	0.394 (7)
16	0.172 (9)	0.183 (10)	0.204 (7)
32	0.102 (9)	0.108 (10)	0.116 (7)
64	0.057 (9)	0.061 (10)	0.064 (7)

Table 1: Solution of the velocity systems: mean execution times and iterations (in brackets) on GRID 1 (top) and GRID 2 (bottom).

block-Jacobi preconditioner achieves a speedup of 17 on GRID 1 and of 23 on GRID 2. As expected, the strong scalability improves with the size of the grid.

The results on GRID 3 show about the same behaviour, therefore we do not report them for the sake of brevity. We can conclude that it is not worth using more "sophisticated" preconditioners on the velocity systems, and that GMRES with no preconditioner is generally the best choice on the available machine.

4.2. Preconditioners for the pressure systems

As observed in Section 2, multilevel Schwarz preconditioners are a natural choice for the pressure systems; for comparison purposes, we also consider one-level AS preconditioners. We note that, by varying algorithmic parameters, several preconditioners with different performances can be obtained. The results presented here are aimed at identifying the best tradeoff between the effectiveness and the (strong) scalability of the preconditioner for the problem under consideration.

Among the one-level AS preconditioners, we consider RAS with overlap 0, 1 and 2 and the incomplete LU factorization with no fill-in (ILU(0)) to the local systems arising in its application. In the following, we use RAS(n) to denote RAS with overlap n. Concerning the multilevel preconditioners, we discuss the results obtained with the two- and three-level V-cycles, using 1 block-Jacobi sweep as pre-/post smoother, and four Jacobi or block-Jacobi sweeps, as well as the ILU(0) factorization, to (approximately) solve the coarsest-level system. The coarsest matrix is distributed among the processors when the Jacobi methods are used, otherwise it is replicated on each processor. ILU(0) is applied

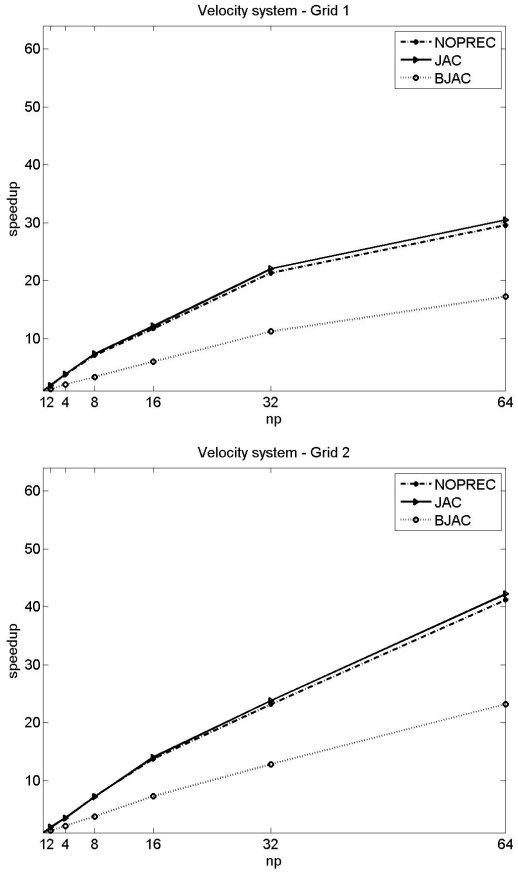

Figure 2: Solution of the velocity systems: speedups on Grid 1 (top) and Grid 2 (bottom).

to the blocks within the block-Jacobi sweeps. Furthermore, the threshold in the coarsening algorithm is set to 10^{-2}, to take into account the anisotropy of the grids in the wall-normal direction. By numerical experiments we found that using more than 3 levels does not improve the overall performance of the preconditioners on our problems, therefore we do not show the results obtained with a larger number of levels.

In the following, the multilevel preconditioners are denoted by mLDPJ, mLDBJ and mLRIF, where mL indicates that m levels are used, D or R indicates that the coarsest matrix is distributed or replicated, and BJ, PJ and IF denote the Jacobi (also called point-Jacobi), block-Jacobi and incomplete factorization methods used at the coarsest level, respectively.

In Tab. 2 and 3 we report, for the three grids, the mean time (in seconds), and the corresponding average number of iterations, for solving a single pressure system with RAS and with the two-level and three-level preconditioners. In the multilevel case, we also report the size of the coarsest matrix. Note that RAS(0),

Grid 1			
NP	RAS(0)	RAS(1)	RAS(2)
1	1.135 (13)	1.135 (13)	1.135 (13)
2	0.735 (17)	0.788 (17)	0.813 (17)
4	0.376 (18)	0.387 (17)	0.410 (17)
8	0.203 (18)	0.209 (17)	0.227 (17)
16	0.135 (22)	0.142 (20)	0.153 (20)
32	0.069 (21)	0.073 (19)	0.085 (19)
64	0.048 (22)	0.052 (19)	0.064 (19)

Grid 2			
NP	RAS(0)	RAS(1)	RAS(2)
1	3.703 (12)	3.703 (12)	3.703 (12)
2	2.046 (13)	2.119 (13)	2.085 (12)
4	1.221 (15)	1.286 (14)	1.299 (14)
8	0.570 (16)	0.626 (15)	0.665 (15)
16	0.315 (18)	0.335 (16)	0.353 (16)
32	0.185 (19)	0.198 (17)	0.216 (17)
64	0.115 (21)	0.124 (19)	0.141 (19)

Grid 3			
NP	RAS(0)	RAS(1)	RAS(2)
1	8.808 (12)	8.808 (12)	8.808 (12)
2	6.027 (15)	6.469 (15)	6.612 (15)
4	3.230 (16)	3.463 (15)	3.519 (15)
8	1.909 (18)	2.050 (17)	2.102 (17)
16	1.076 (20)	1.114 (19)	1.188 (19)
32	0.492 (20)	0.570 (19)	0.584 (19)
64	0.266 (22)	0.296 (20)	0.332 (20)

Table 2: Solution of the pressure systems: mean execution times and iterations (in brackets) with RAS on Grid 1 (top), Grid 2 (middle) and Grid 3 (bottom).

RAS(1) and RAS(2) are the same preconditioner on one processor; however, we show the corresponding data three times, for ease of readability. The same observation applies to 2LDBJ and 2LRIF, and to 3LDBJ and 3LRIF.

We see that, among the RAS variants, RAS(0) (i.e., block-Jacobi) is the most efficient preconditioner. The slight reduction of the number of iterations obtained with RAS(1) and RAS(2) on more than two processors is not sufficient to get execution times smaller than those achieved with RAS(0). We note that a few more iterations are required for Grid 1 with respect to Grid 2 and, generally, Grid 3. This can be explained by considering the differences in the pressure matrices introduced by the discretization along the wall-normal direction. As expected, the two-level and three-level preconditioners significantly reduce the number of iterations with respect to RAS. In terms of iterations they show about the same behaviour on the three grids; a slight increase of the number of iterations can be observed as the number of processors grows, which is due to the use of an inexact solver at the coarsest level. Generally, the most efficient multilevel preconditioners are 3LDBJ and 2LDBJ, where the best tradeoff

GRID 1								
NP	C-size	2LDPJ	2LDBJ	2LRIF	C-size	3LDPJ	3LDBJ	3LRIF
1	10368	1.309 (5)	1.127 (4)	1.127 (4)	338	1.099 (4)	1.067 (4)	1.067 (4)
2	9755	0.816 (6)	0.672 (5)	0.803 (5)	388	0.609 (4)	0.599 (4)	0.605 (4)
4	10048	0.409 (6)	0.334 (5)	0.526 (5)	736	0.320 (4)	0.316 (4)	0.318 (4)
8	10128	0.215 (6)	0.186 (5)	0.426 (5)	868	0.177 (5)	0.179 (5)	0.187 (5)
16	10232	0.137 (8)	0.108 (6)	0.395 (6)	968	0.107 (5)	0.104 (5)	0.112 (5)
32	10288	0.078 (7)	0.063 (6)	0.359 (6)	1280	0.070 (5)	0.069 (5)	0.076 (5)
64	10560	0.066 (7)	0.054 (6)	0.381 (6)	1904	0.074 (5)	0.073 (5)	0.075 (5)

GRID 2								
NP	C-size	2LDPJ	2LDBJ	2LRIF	C-size	3LDPJ	3LDBJ	3LRIF
1	48108	4.253 (5)	3.771 (4)	3.771 (4)	1530	3.523 (4)	3.509 (4)	3.509 (4)
2	48325	2.237 (5)	2.111 (5)	2.479 (5)	1635	2.090 (4)	2.090 (4)	2.101 (4)
4	53138	1.337 (6)	1.187 (5)	1.847 (5)	2306	1.152 (4)	1.117 (4)	1.199 (4)
8	53632	0.738 (6)	0.630 (5)	1.474 (5)	2652	0.606 (5)	0.596 (4)	0.711 (4)
16	53820	0.402 (7)	0.362 (6)	1.384 (6)	4368	0.379 (4)	0.377 (5)	0.519 (5)
32	51640	0.236 (7)	0.202 (6)	1.351 (6)	3344	0.211 (4)	0.209 (5)	0.337 (5)
64	52752	0.148 (8)	0.127 (6)	1.383 (6)	4016	0.142 (4)	0.138 (5)	0.296 (5)

GRID 3								
NP	C-size	2LDPJ	2LDBJ	2LRIF	C-size	3LDPJ	3LDBJ	3LRIF
1	117872	9.529 (5)	8.974 (4)	8.974 (4)	2879	8.285 (4)	8.343 (4)	8.343 (4)
2	130000	6.226 (6)	5.450 (5)	6.518 (5)	4568	4.632 (4)	4.544 (4)	4.636 (4)
4	128596	3.192 (6)	2.824 (5)	4.431 (5)	4174	2.406 (4)	2.356 (4)	2.449 (4)
8	130868	1.857 (6)	1.611 (5)	3.720 (5)	4684	1.400 (4)	1.367 (4)	1.532 (4)
16	124376	1.048 (7)	0.871 (6)	3.834 (6)	4152	0.736 (4)	0.727 (4)	0.897 (4)
32	124888	0.566 (7)	0.471 (6)	3.164 (6)	4376	0.413 (5)	0.399 (4)	0.605 (4)
64	127456	0.322 (8)	0.265 (6)	3.235 (6)	4976	0.245 (5)	0.237 (5)	0.488 (5)

Table 3: Solution of the pressure systems: mean execution times and iterations (in brackets) with the multilevel preconditioners on GRID 1 (top), GRID 2 (middle) and GRID 3 (bottom).

between the effectiveness of the coarse-level correction and the cost of the preconditioner is achieved. More precisely, 3LDBJ achieves the smallest execution times in all the cases but GRID 1 on 32 and 64 processors, and GRID 2 on 16, 32 and 64 processors, where 2LDBJ is the winner. However, from the results on GRID 3 we see that using three levels becomes effective on a large number of processors when the grid size increases. The preconditioners 2LRIF and 3LRIF are not competitive, because of the cost of the coarsest-level correction; on the other hand, using a very simple coarsest-level solver does not reduce the time, because it generally produces a small increase of the number of iterations. It is interesting to note that RAS(0) outperforms the best multilevel preconditioner in some cases: on 64 processors with GRID 1 and on 16, 32 and 64 processors with GRID 2. On the other hand, the results on GRID 3 show that RAS loses its efficiency when the number of processors increases.

In Fig. 3 we compare the strong scalability of all the preconditioners for the three grids. On GRID 1 the speedups of all the preconditioners generally show close behaviours up to 32 processors, except 2LRI where the cost for replicating and factorizing a large coarsest-level system strongly reduces the scalability of the preconditioned solver. In this case, the best speedup is generally obtained with 2LDBJ; in particular, it is about 18 on 32 processors. The situation is different on 64 processors, where RAS(0) achieves the best speedup (about 24), closely followed by RAS(1) and 2LDBJ. Actually, increasing the number of processors from 32 to 64 does not yield a significant time gain, because of the relatively small size of the coarse grid, and the cost for the multilevel correction is not paid off. On the larger grids, the speedup of all the preconditioners but 2LRIF and 3LRIF shows an increasing behaviour up to 64 processors. On GRID 2 the largest speedup is achieved by RAS(0), closely followed by RAS(1), 2LDBJ and 2LDPJ. Excluding 2LRIF and 3LRIF, the speedup on 64 processors ranges from 25 (3LDPJ/3LDBJ) to 32 (RAS), which can be considered satisfactory. Conversely, on GRID 3 the best scalability is obtained by 3LDBJ, followed by 3LDPJ and 2LDBJ; in particular, 3LDBJ gets a speedup of 35. These results clearly show that the performance of the multilevel preconditioners increases with the grid size.

From the previous discussion we can conclude that 3LDBJ and 2LDBJ are generally to be preferred to the other preconditioners in our LES application, for their good behaviour in terms of both execution time and scalability.

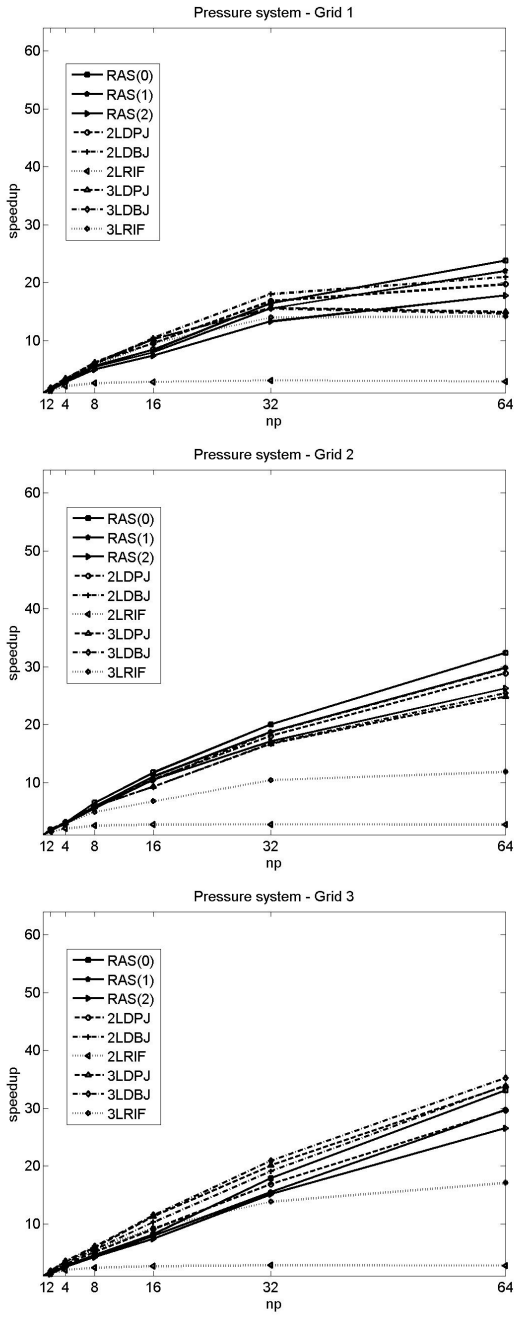

Figure 3: Solution of the pressure systems: speedups on Grid 1 (top), Grid 2 (middle) and Grid 3 (bottom).

5. Conclusions

The work presented in this paper was devoted to analysing the performance of different parallel Schwarz preconditioners in the solution of linear systems arising in a LES procedure for turbulent channel flows. The preconditioners were applied within SParC-LES, a LES code recently developed by exploiting the PSBLAS and MLD2P4 libraries, implementing parallel sparse linear algebra kernels and preconditioners. We compared one-level and multilevel Schwarz preconditioners, with the final aim of achieving the best tradeoff among convergence, time and scalability objectives on a test case widely used by the LES community. The results show that, on the pressure systems, the two-level and three-level preconditioners using a distributed block-Jacobi solver at the coarsest level provide a significant reduction of the number of iterations with respect to RAS, which generally translates into smaller execution times and good scalability, especially on the largest grid. Conversely, the features of the velocity systems make worthless the use of preconditioners on the relevant matrices.

References

[1] Y. Saad, *Iterative methods for sparse linear systems*, 2nd edition, SIAM, 2003.

[2] P. D'Ambra, D. di Serafino, and S. Filippone, *MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95*, ACM Trans. Math. Software, 37 (2010), art. 30, 23 pages.

[3] S. Filippone and M. Colajanni, *PSBLAS: a library for parallel linear algebra computation on sparse matrices* ACM Trans. Math. Software, 26 (2000), pp. 527–550.

[4] A. Aprovitola, P. D'Ambra, F. M. Denaro, D. di Serafino, and S. Filippone, *Developing parallel large eddy simulation software with libraries for sparse matrix computations*, in preparation.

[5] LESinItaly Group, *A comparative test for assessing the performances of large-eddy simulation codes*, in "Atti del XX Congresso dell'Associazione Italiana di Meccanica Teorica e Applicata", F. Ubertini, E. Viola, S. de Miranda, and G. Castellazzi, eds., 2011, ISBN 978-88-906340-1-7 (electronic), available from `http://www.lamc.ing.unibo.it/aimeta2011/`.

[6] A. Aprovitola, P. D'Ambra, F. M. Denaro, D. di Serafino, and S. Filippone, *Scalable algebraic multilevel preconditioners with application to CFD*, in "Parallel Computational Fluid Dynamics 2008", D. Tromeur-Dervout, G. Brenner, D. Emerson, J. Erhel, eds., Lecture Notes in Computational Science and Engineering, vol. 74, Springer, 2010, pp. 15–27.

[7] A. Aprovitola, P. D'Ambra, D. di Serafino, and S. Filippone, *On the use of aggregation-based parallel multilevel preconditioners in the LES*

of wall-bounded turbulent flows, in "Large-Scale Scientific Computing", I. Lirkov, S. Margenov, and J. Wasniewski, eds., Lecture Notes in Computer Science, vol. 5910, Springer, 2010, pp. 67–75.

[8] A. Aprovitola, *Sull'applicabilità degli schemi upwind multidimensionali ai volumi finiti per la simulazione numerica delle grandi scale di flussi turbolenti confinati*, Ph.D. dissertation (in Italian), Second University of Naples, 2006.

[9] A. Aprovitola and F. M. Denaro, *On the application of congruent upwind discretizations for large eddy simulations*, J. Comput. Phys., 194 (2004), pp. 329–343.

[10] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix. Anal. Appl., 18 (1997), pp. 37–51.

[11] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput. 21 (1999), pp. 792–797.

[12] E. Efstathiou and J.G. Gander, *Why restricted additive Schwarz converges faster than additive Schwarz*, BIT Numer. Math., 43 (2003), pp. 945–959.

[13] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain decomposition. Parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, 1996.

[14] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[15] A. Buttari, P. D'Ambra, D. di Serafino, S. Filippone, S. Gentile, B. Ucar, *A novel aggregation method based on graph matching for algebraic multigrid preconditioning of sparse linear systems*, International Conference on Preconditioning Techniques for Scientific and Industrial Applications, May 2011, Bordeaux, France. Available from `http://precond11.bordeaux.inria.fr/`.

[16] P. Vaněk, J. Mandel, M. Brezina, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.

[17] M. Sala, R. Tuminaro, *A New Petrov-Galerkin Smoothed Aggregation Preconditioner for Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 31 (2008), pp. 143–166.

[18] P. D'Ambra, D. di Serafino, and S. Filippone, *MLD2P4 user's and reference guide*, March 2011, available from `http://www.mld2p4.it/docs.php`.

[19] R. D. Moser, J. Kim, and N. N. Mansour, *Direct numerical simulation of turbulent channel flow up to* $Re_\tau = 590$, Physics of Fluids, 11 (1999), art. 943, 3 pages.