INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2005; **48**:367–395
Published online 14 February 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.902

Large eddy simulation of turbulent flows via domain decomposition techniques. Part 1: theory

Marcello Manna[1,*,†], C. Benocci[2] and E. Simons[2]

[1]*Dipartimento di Ingegneria Meccanica per l'Energetica, Università di Napoli 'Federico II', via Claudio 21, 80125, Naples, Italy*
[2]*Department of Environmental and Applied Fluid Dynamics, von Kármán Institute for Fluid Dynamics, Chaussée de Waterloo 72, B-1640 Rhode St. Genese, Belgium*

SUMMARY

The present paper discusses large eddy simulations of incompressible turbulent flows in complex geometries. Attention is focused on the application of the Schur complement method for the solution of the elliptic equations arising from the fractional step procedure and/or the semi-implicit discretization of the momentum equations in velocity–pressure representation. Fast direct and iterative Poisson solvers are compared and their global efficiency evaluated both in serial and parallel architecture environments for model problems of physical relevance. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: LES; DNS; domain decomposition; elliptic kernel

1. INTRODUCTION

Numerical simulation of turbulent flows can be achieved applying direct numerical simulation approach (DNS), where all the relevant scales are numerically resolved, or large eddy simulation approach (LES), where only the largest and anisotropic scales are resolved while the smallest ones are modelled by an *ad hoc* model [1]. Both procedures are being considered mature for application to flows more complex than simple *building block* flows whose simulation has justified their success and diffusion as reliable turbulence research tools. In this frame, the term *complexity* covers both more challenging physical problems (e.g. involving phase changes, combustion, shocks and acoustics to name a few) as well as flows in geometries of increased complexity (e.g. blunt and bluff body flows in composite configurations). These issues may, and usually do, occur simultaneously making it difficult, if not impossible, to consider them separately.

*Correspondence to: Marcello Manna, Dipartimento di Ingegneria Meccanica per l'Energetica, Università degli Studi di Napoli 'Federico II', via Claudio 21, 80125 Naples, Italy.
†E-mail: manna@unina.it

Received 9 January 2004
Revised 25 October 2004
Accepted 4 November 2004

The latest achievements in the field of simulation have been made possible by the rapid advancements of the ready-to-use hardware technology and by the combined progresses of the numerical algorithms and subgrid stress closures [1, 2]. In particular, LES can be considered, in some senses, mature for industrial application, as witnessed by the introduction of this technique in commercial software packages [3]. Nevertheless, we must reckon that, despite the impressive progresses in the field of LES, their application to complex separated flows remains an extremely challenging task, as shown by results and conclusions of recent workshops [4].

While not disregarding the need for further improvements of the subgrid models, one of the key issues which justifies due attention is the efficient solution of the elliptic kernel arising from the temporal discretization of the incompressible governing equations. In this frame, we consider that, for flow problems where geometry includes sharp edges of rectangular shapes, no practical alternative exists, at least in the structured grid context, to the application of multi-domain (MD) technique.

This approach allows to decouple the original problem in a set of two sub-problems, *viz.* a sub-domain problem and an interface condition. The evident advantages of this procedure lie with the easiness of the description and discretization of complex geometries, good grid control, feasibility of parallel execution on multi-node computers, and, *fundamentally* the possibility to work on simple sub-domains where fast elliptic solvers (FES) could be applied. Moreover, this approach can be easily generalized to the treatment of non-Cartesian geometry, applying the immersed boundary method [5], as shown by in-house testing [6], or re-casting the problem in curvilinear co-ordinates, while maintaining the above-quoted advantages in terms of efficient implementation and execution.

Spectral MD approaches have been widely discussed in the recent literature and in this respect we must mention the work of Patera [7] and the successive developments and applications of Henderson and Karniadakis [8]. In the finite difference and finite volume framework, the application of the MD technique is less common, and we are not aware of any such fluid dynamic oriented development (in the sense of LES or DNS). Part of the reasons may be traced back to the difficulties associated with the application to the staggered variables unknowns arrangement, which is commonly employed for the numerical solution of the incompressible Navier–Stokes equations in a finite difference framework.

In the present work we discuss the construction of a non-overlapping domain decomposition strategy for the elliptic kernel which extends the use of fast direct and iterative elliptic solvers to any geometry consisting of an arbitrary collection of rectangular sub-domains. Emphasis is laid on the interface problem formulation and to the related conceptual difficulties for those Navier–Stokes solvers based on a staggered variables arrangement. While the approach has been thought and tested in the LES frame, it remains, of course, equally promising in the DNS domain. Applications to complex flows are presented in the companion paper [9].

The article is organized as follows: in Section 2 the governing equations and the subgrid scale model are briefly recalled. Section 3 describes the numerical method, *viz.* the spatial and temporal discretization in a fractional step procedure. Section 4 addresses the multi-domain discretization of the elliptic kernel, both from theoretical and practical point of views. The following Section 5 discusses the numerical solution of the elliptic problems arising from the fractional step procedure. Finally, Section 6 reports a comparison of the performance of several iterative solution strategies for geometrical configurations which are of relevance in the field of LES and DNS. Implementation and performance on parallel architectures are discussed. Conclusions are given in Section 7.

2. GOVERNING EQUATIONS

Key of the LES approach is the separation of the different scales, which is achieved applying a filtering operator, taking the form of a space filter, expressed as a convolution integral

$$\bar{u}_i(\boldsymbol{x},t)=\iiint_V G(\boldsymbol{x}-\boldsymbol{\xi},\Delta)u_i(\boldsymbol{\xi},t)\,\mathrm{d}\boldsymbol{\xi} \tag{1}$$

where Δ is the filter width and V the computational domain. In the case of finite difference (FD) and finite volume (FV) discretization G is usually taken to be the well known *top hat* filter [1]

$$G(\boldsymbol{x}-\boldsymbol{\xi},\Delta)=\begin{cases}1/\Delta & |\boldsymbol{x}-\boldsymbol{\xi}|\leqslant\Delta/2\\ 0 & \text{otherwise}\end{cases} \tag{2}$$

The continuity and momentum equations in terms of the resolved (*grid* scales) variables (denoted with an over bar) become

$$\frac{\partial\bar{u}_i}{\partial x_i}=0 \tag{3}$$

$$\frac{\partial\bar{u}_i}{\partial t}+\frac{\partial\bar{u}_i\bar{u}_j}{\partial x_j}+\frac{1}{\rho}\frac{\partial\bar{p}}{\partial x_i}=\frac{1}{Re}\frac{\partial^2\bar{u}_i}{\partial x_j\partial x_j}-\frac{\partial\tau_{ij}}{\partial x_j}+f_i \tag{4}$$

where $Re=\upsilon\ell/\nu$ is the Reynolds number (υ and ℓ being appropriate velocity and length scales), and the contribution of the un-resolved scales (*subgrid* scales)

$$\tau_{ij}\equiv\overline{u_iu_j}-\bar{u}_i\bar{u}_j \tag{5}$$

takes the form of a stress tensor (subgrid scale stress tensor, SGS) which must be modelled to close the problem. It is well understood that the main role of subgrid scales at the level of the resolved scales is to provide the necessary amount of dissipation [10]. In consequence, the most commonly applied models rely on the concept of subgrid diffusivity. Separating subgrid diffusivity in isotropic and anisotropic contributions and adding the isotropic part of τ_{ij} to the pressure term, the anisotropic part takes the form

$$\tau_{ij}-\tfrac{1}{3}\tau_{kk}\delta_{ij}=-2\mu_T\bar{S}_{ij} \tag{6}$$

with $\bar{S}_{ij}=1/2(\partial u_i/\partial x_j+\partial u_j/\partial x_i)$ the resolved rate of strain.

Different subgrid models based on this approach can be found in literature [1], but the most widely applied model remains the Smagorinsky model

$$\tau_{ij}-\tfrac{1}{3}\tau_{kk}\delta_{ij}=-2C_s\rho\Delta|\bar{S}|\bar{S}_{ij} \tag{7}$$

where $\Delta=(\Delta x\Delta y\Delta z)^{1/3}$ is the filter width, and $|\bar{S}|=(2\bar{S}_{ij}\bar{S}_{ij})^{1/2}$ the magnitude of the resolved rate of strain tensor.

In the case of wall bounded flow, this model must be complemented by a van Driest-type damping function [1], $D=1-\exp(z^+/A^+)^3$, with $A^+=25$, applied in the wall-normal direction z, in order to impose the correct turbulence decrease towards the wall

$$\tau_{ij}-\tfrac{1}{3}\tau_{kk}\delta_{ij}=-2DC_s\rho\Delta|\bar{S}|\bar{S}_{ij} \tag{8}$$

The choice of the Smagorinsky model might appear non-optimal, in view of the current trend towards application of scale-similar or mixed subgrid scale models [11, 12], and the well known limitations of this model against the dynamic procedure [13]; however, its application would be justified in the present study, whose purpose is to demonstrate the capability of the multi-domain approach to the LES of complex flows and, ultimately, to practical applications. In this optic, the aim is the prediction of low order turbulent moments at the minimal cost. Under this point of view, it must be kept in mind that present tests [14], as well as current literature [4, 15], point out that, in case of complex flows and finite difference implementation, more advanced and expensive subgrid models offer results which are not substantially better than those obtained with the Smagorinsky model.

3. NUMERICAL METHOD

Following the standard pressure correction scheme [16], at each time level, we have decoupled the velocity and pressure in a projection and correction step. In this procedure, the projection step requires the solution of a Poisson equation for the pressure. Concerning the time-marching procedure itself, it must be considered that the choice of size of time step for DNS and LES simulations is ruled by accuracy requirements: numerical tests [17] have shown that the Courant–Friedrich–Levy number (CFL)

$$\Delta t_c \left(\frac{|\overline{u_i}|}{\Delta x_i} \right)_{\max} \tag{9}$$

must be kept smaller than 0.5–1.0 to avoid contamination of the results. Therefore, we considered that the extra complexity and extra cost introduced by an implicit time-marching algorithm would not be justified, and an explicit algorithm had to be preferred on ground of its lower number of operations per grid point, easiness of its implementation of MD procedure and its execution on parallel computers (see next paragraph). In this optic, the second-order Adams Bashforth scheme is chosen to integrate the predicted velocity field, because it offered the minimum requirements of memory storage and number of algebraic operations for grid point. A comparison of the performance of different high order schemes for the time integration of the Navier–Stokes equations in a fractional step formulation can be found in Reference [18].

Having re-defined the pressure as p/ρ, the semi-discrete form of Equation (4) takes the form

$$\frac{\bar{u}_i^* - \bar{u}_i^n}{\Delta t} = \frac{3}{2} \bar{R}_i^n - \frac{1}{2} \bar{R}_i^{n-1} - \frac{\partial \bar{p}^n}{\partial x_i} \tag{10}$$

followed by

$$\frac{\bar{u}_i^{n+1} - \bar{u}_i^*}{\Delta t} = - \frac{\partial(\delta p)}{\partial x_i} \tag{11}$$

where the term R is given by

$$R_i \equiv - \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} + \frac{1}{Re} \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} + f_i \tag{12}$$

and $\delta\bar{p} \equiv \bar{p}^{n+1} - \bar{p}^n$ is the pressure correction. Taking the divergence of (11) leads to

$$\frac{\partial^2 \delta\bar{p}}{\partial x_j \partial x_j} = \frac{1}{\Delta t} \frac{\partial \bar{u}_j^*}{\partial x_j} \tag{13}$$

which allows to compute $\bar{p}^{n+1}$ and $\bar{u}_i^{n+1}$ through (11). Equation (13), closed by the appropriate set of boundary conditions, constitutes, in the present formulation, the elliptic kernel.

The Adams Bashforth algorithms are limited to a CFL number

$$\Delta t_c \left(\frac{|\bar{u}_i|}{\Delta x_i} \right)_{\max} \leqslant 0.35 \tag{14}$$

while a more critical limit for the simulation of wall-bounded flows is the corresponding stability limit for the diffusivity operator, which for the present algorithm is

$$\nu \left(\frac{\Delta t_v}{\Delta x_i^2} \right)_{\max} \leqslant 0.1 \tag{15}$$

This second limit would appear too constraining for attached flows, but it is the experience of the present authors [19] that, in the cases of bluff body separation, the need to resolve the high velocity gradients both upstream and downstream of the corners causes a severe limit on the advective step (due to the wall-normal advection term) which becomes stricter than the diffusive one. It was then concluded that this diffusive limit was of minor importance for the applications considered herein.

It can be remarked that the present stability limits are well below the aforequoted limits [17], which ensures that the numerical errors will not degrade the theoretical accuracy of the scheme. It is evident that, when the resolution of the inner wall layer is required, as it is the case of wall-resolved LES, the stability limit for the diffusivity operator quickly becomes much stricter than the advection-based CFL; however, we considered that the above discussed advantages of the present choice made this limitation acceptable. As a matter of fact, it was found that the present algorithm remains competitive, down to grids where the first inner point is located at $z^+ \approx 1$. While this limit could appear unacceptable for low-Reynolds RANS, it can easily accommodate the minimal requirement for resolved LES, which requires, for the direction normal to the wall, three grid points to be located within the first 10 wall units [10]. A more detailed discussion of minimal grid sizes for LES will be presented in Reference [9].

All partial derivatives in Equations (10)–(13) are discretized with second-order accurate centred formulae in a staggered grid arrangement, so that velocities are defined at cell faces and pressure unknowns at cell centres (see Figure 1). Advantages and shortcomings of the staggered formulation are well known and documented in great detail in standard textbooks [20]; here, we only have to recall that its larger computational complexity is compensated by the absence of velocity pressure odd–even decoupling.

The convective terms in (12) are discretized in a skew symmetric form according to the Arakawa [21] formulation (which is energy and momentum preserving). As a matter of fact, it is well established that second-order central formulae yield numerical errors lower than those of any upwind-biased scheme of order up to 5 [22]; this conclusion has also been confirmed by a more recent study [23]. Therefore, in the case of simulation of flow at low or moderate Reynolds number, which is the case for the present investigation, it is worthwhile to

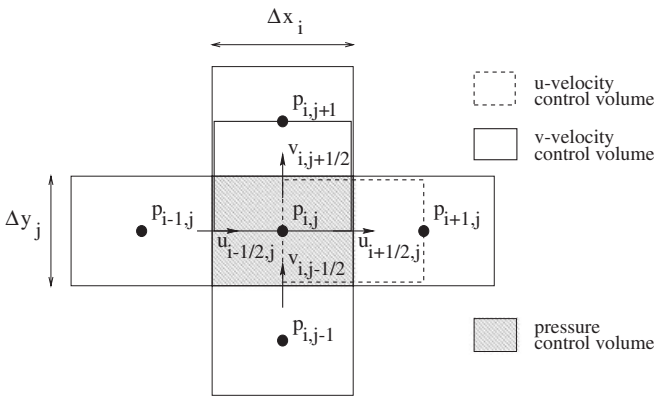

Figure 1. Velocities and pressure variables locations in a staggered grid arrangement.

apply the second-order central scheme and to refine the grid to avoid the presence of aliasing effects [14].

The discretization of (12) in the staggered grid approach is trivial and will be omitted for sake of brevity; conversely, the discretization of Equation (13) in a complex geometry environment, where the complexity refers to the presence of one or more bluff bodies of rectangular shape, will be addressed in quite some detail in the following section.

4. THE MULTI-DOMAIN DISCRETIZATION OF THE ELLIPTIC KERNEL

As already mentioned in Section 3, and independent of the temporal discretization of the convective term, all LES and DNS codes adopting a pressure correction scheme, require the solution of one or more three-dimensional elliptic (Poisson or Helmholtz) equations with a time varying right-hand side for a large number of time steps (typically $O(10.000)$) to adequately collect the statistics. It is therefore self evident that the efficient solution of these systems, both in terms of speed and storage requirements, is one of the most pacing items for the application of LES/DNS to turbulent flows in more complex geometries.

For simple box-like geometries all research groups have traditionally resorted to the so-called FES [24, 25], which are a family of direct methods employing fast Fourier transformation, cyclic reduction and Gaussian elimination. They can be shown to be optimal in terms of speed as well as storage requirements, for those class of problems to which they can be applied. Their drawback is precisely their limited range of application. The alternative is represented by the multi-domain technique.

The basic idea of any MD approach is to decompose the physical domain Ω into a set of sub-domains Ω_i according to some criteria; following Reference [26] (for a more complete and exhaustive discussion on the topic the reader is referred to the book of Quarteroni and Valli [27]) we loosely divide them into overlapping or *Schwarz domain decomposition methods* as opposed to non-overlapping domain decomposition methods, also commonly referred to as *sub-structuring* or *Schur complement methods*.

In Schwarz methods the overall problem is typically solved by obtaining a solution in an alternating way on each of the overlapping sub-domains using as boundary condition on the

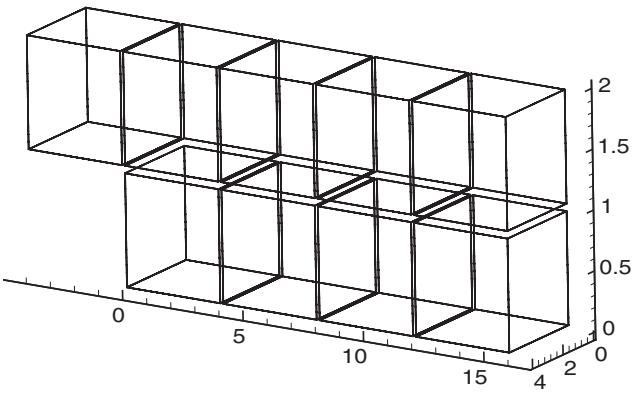

Figure 2. Multi-domain breakup of a backward facing step geometry.

internal overlapping boundaries the value obtained from a previous intermediary solution of the neighbouring sub-domains problem until some overall convergence criterion is satisfied. The amount of overlap between the sub-domains guarantees the exchange of information between the sub-domains and affects therefore strongly the convergence speed of the overall iterative process (see Reference [27] for additional details). This is in contrast with the non-overlapping methods where the exchange of information between the sub-domains solves is established through the definition and resolution of an explicit interface problem. Thus, in the *sub-structuring* approach the sub-domains are *patched* with each other (see Figure 2).

It can be shown that the two methods are more similar than it might be thought at first glance; details about the convergence properties of the two formulations for elliptic kernels can be found in Reference [27] and will be omitted. Here we simply wish to mention that, on the basis of computational cost estimates provided in References [27–29], we have preferred the Schur method over the Schwarz one.

4.1. The Schur complement method

In this work, and as far as we are aware for the first time in the field of DNS/LES, we propose a non-overlapping domain decomposition technique based on the Schur complement method, which extends the use of these solvers to any geometry that can be decomposed into a collection of non-overlapping rectangular sub-domains [14, 19, 30]. The cost of the resulting algorithm will be shown to be twice an elliptic solve on each of the sub-domains separately (here and after referred to as problem P_I), supplemented with one solve for the interface unknowns (problem P_Γ). Problem P_I can be efficiently solved on any parallel architecture machine with the aforementioned FES.

Unlike the more widely adopted *capacitance matrix method* [31, 32], which also requires two elliptic solves, but now on an enlarged box-like domain embedding the complex geometry, the method we propose does not introduce any additional unknown and is naturally suited for a parallel implementation.

For reasons of simplicity we shall restrict ourselves to the class of problems presenting at least one homogeneous spatial direction; by doing so we reduce the complexity of the original

three-dimensional problem to that pertaining to a set of decoupled two-dimensional problems. The extension to fully three-dimensional problems is trivial.

4.2. Theoretical aspects

The discretization of (13) in a multi-domain formulation is straightforward. Let us begin with some nomenclature. Let $\mathbf{x}_\Omega$ and $\mathbf{x}_\Gamma$ denote the solution vectors corresponding to the sub-domains and to the interfaces; the former represents the numerical approximation of the original problem defined in any of the rectangles whose collection adds up to the original domain. In a collocated formulation the interface Γ is not well defined at discrete level (*viz.* there are no unknowns on the boundaries separating two sub-domains). We thus define Γ as one (or more) layer of unknowns next to the boundary lines, which are therefore eliminated from the sub-domain count. Obviously $\mathbf{x} = \mathbf{x}_\Omega \cup \mathbf{x}_\Gamma$ (see Figure 3).

For reasons of efficiency we have used Fourier expansion of the unknowns and the right-hand sides of (13), rewritten as $\Delta_2 \phi = f$, in the homogeneous spanwise direction y, i.e.

$$\phi_{i,j,k} = \sum_{m=0}^{N_y-1} \hat{\phi}_{i,m,k} \mathrm{e}^{2\pi I j m/N_y}, \qquad f_{i,j,k} = \sum_{m=0}^{N_y-1} \hat{f}_{i,m,k} \mathrm{e}^{2\pi I j m/N_y} \tag{16}$$

where m and N_y are the wave number and the amount of grid points in the homogeneous direction, respectively. By doing so, the original three-dimensional Poisson problem, whose discretized form evaluated at point (i,j,k) reads

$$\left(\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2} + \frac{\delta^2}{\delta z^2} \right) \phi_{i,j,k} = f_{i,j,k} \tag{17}$$

is reduced to a set of $N_y/2$ two-dimensional decoupled Helmholtz problems

$$\left(\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta z^2} - \lambda_m \right) \hat{\phi}_{i,m,k} = \hat{f}_{i,m,k}, \quad m = 0, \ldots, N_y - 1 \tag{18}$$

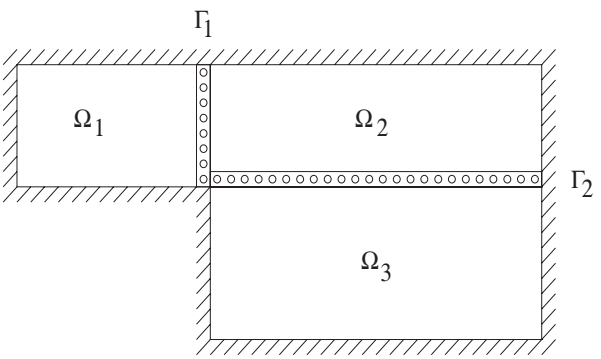

Figure 3. Example of domain decomposition: 3 sub-domains Ω_1, Ω_2 and Ω_3 separated by 2 interfaces Γ_1 and Γ_2.

which differ from each other exclusively for the parameters

$$\lambda_m = \frac{2}{\Delta_y^2}\left[1 - \cos\left(\frac{\pi(m+1)}{N_y}\right)\right] \tag{19}$$

usually referred to as the Helmholtz entries. In Equations (17) and (18) δ denotes a discrete difference operator whose exact form is irrelevant to the present discussion. Note that because of the symmetry of the Fourier transformed equations, we only need to solve $N_y/2$ problems.

If we order the unknowns in a domainwise fashion, followed by the interface unknowns, and we apply block Gaussian elimination to the original system of equations arising from the discretization, we find

$$A\mathbf{x} = \mathbf{b} \tag{20}$$

that is

$$\begin{pmatrix} A_{1,1} & \dots & 0 & A_{1,N+1} & \dots & A_{1,N+M} \\ \vdots & \ddots & \vdots & \dots & \dots & \dots \\ 0 & \dots & A_{N,N} & A_{N,N+1} & \dots & A_{N,N+M} \\ 0 & 0 & 0 & S_{N+1,N+1} & \dots & S_{N+1,N+M} \\ 0 & 0 & 0 & \dots & \dots & \dots \\ 0 & 0 & 0 & S_{N+M,N+1} & \dots & S_{N+M,N+M} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{\Omega_1} \\ \dots \\ \mathbf{x}_{\Omega_N} \\ \mathbf{x}_{\Gamma_1} \\ \dots \\ \mathbf{x}_{\Gamma_M} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \dots \\ \mathbf{b}_N \\ \tilde{\mathbf{b}}_{N+1} \\ \dots \\ \tilde{\mathbf{b}}_{N+M} \end{pmatrix} \tag{21}$$

where $A_{i,i}$ is the matrix representing the discretization of the LHS of (13) in Ω_i, N and M the number of sub-domains and interfaces, and

$$S_{ij} \equiv A_{ij} - \sum_{k=1}^{N} A_{ik} A_{kk}^{-1} A_{kj} \quad \forall i, j = N+1, N+M \tag{22}$$

$$\tilde{\mathbf{b}}_j \equiv \mathbf{b}_j - \sum_{k=1}^{N} A_{jk} A_{kk}^{-1} \mathbf{b}_k \quad \forall j = N+1, N+M \tag{23}$$

The interface unknowns can be determined from the lower right sub-block of equations

$$S\mathbf{x}_\Gamma = \tilde{\mathbf{b}} \tag{24}$$

that is,

$$\begin{pmatrix} S_{N+1,N+1} & \dots & S_{N+1,N+M} \\ \dots & \dots & \dots \\ S_{N+M,N+1} & \dots & S_{N+M,N+M} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{\Gamma_1} \\ \dots \\ \mathbf{x}_{\Gamma_M} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{b}}_{N+1} \\ \dots \\ \tilde{\mathbf{b}}_{N+M} \end{pmatrix} \tag{25}$$

The submatrix S is generally referred to as the *Schur complement matrix*. Note that the connection between the sub-domain and interface unknowns, which is apparently lost, is embedded

instead in the RHSs of (25). Effectively, to construct these RHSs we have to determine the factors $A_{kk}^{-1}\mathbf{b}_k$, $\forall k = 1, \ldots, N$ which means solving one elliptic equation per sub-domain, since

$$\mathbf{x}_k = A_{kk}^{-1}\mathbf{b}_k \iff A_{kk}\mathbf{x}_k = \mathbf{b}_k \tag{26}$$

where A_{kk} represents the restriction of the original discretized elliptic operator A to sub-domain Ω_k, applying homogeneous Dirichlet boundary conditions at its internal interface boundaries.

Once the solution on the interfaces $\mathbf{x}_\Gamma$ is computed, we can eliminate those unknowns from system (21) simply bringing them to the RHSs.

To summarize the solution algorithm reduces to the following three step procedure:

1. *Problem* P_I (*part* 1): solve an elliptic problem in each of the sub-domains Ω_i, $i = 1, N$ separately, using the original boundary conditions at the external boundaries, homogeneous Dirichlet boundary conditions at the internal boundaries and its original RHSs $\mathbf{b}_i$.
2. *Problem* P_Γ: assemble the RHSs of the interface problem using the step 1 values, and solve the Schur complement problem.
3. *Problem* P_I (*part* 2): solve an additional elliptic problem in each of the sub-domains Ω_i separately, using the original boundary conditions at the external boundaries, imposing now $\mathbf{x}_\Gamma$ as Dirichlet boundary conditions at the interfaces and using the original RHSs $\mathbf{b}_i$.

Although at a first sight the overall cost of the elliptic kernel appears more than doubled (two elliptic solves plus an interface problem of smaller size), this overhead is largely compensated by the fact that this approach not only allows to recover the use of FES on rectangular domains (Problem P_I, i.e. steps 1 and 3), but also opens the way to parallel executions. Additionally, in an iterative solution context, it provides the opportunity of relaxing the stiffness of the full scale original problem (which is reduced to a set of smaller size, better conditioned sub-problems) and is intrinsically suited for zonal approaches and heterogeneous problems.

4.3. Practical aspects

Before describing the particular solution method we employ for the Schur complement system (24), we wish to recall that our direct interest is its application in the field of LES and DNS. Characteristic to these problems is the demand for an accurate solution of a (number of) fixed elliptic equation(s) with varying RHSs for a large amount of time steps.

This suggests that it could be convenient to compute and store the inverse of the Schur matrix S^{-1} through an LU-factorization procedure once and for all in a preprocessing step. The overhead of the costly factorization process will indeed be easily amortized over the large number of time steps for which we can re-use it through simple forward–backward substitution. This involves however the explicit generation and storage of the Schur complement matrices S_{ij}, which are generally block dense matrices due to the fill-in generated by the inverse operators A_{kk}^{-1} appearing in its definition (22).

For this reason it is often preferred to avoid the need for generating the Schur matrix altogether in step 2 of the above solution algorithm. Indeed, with an iterative method the explicit construction of S is replaced by a series of matrix vector multiplications at every time step. From definition (22) we find that a matrix–vector multiplication $S_{ij}x$ involves determining terms of the form $z \equiv A_{kk}^{-1}y$ for $y \equiv A_{kj}x$, which is equivalent to finding a solution of $A_{kk}z = y$. The last operation is nothing else than a decoupled elliptic solve on sub-domain Ω_k.

This however would greatly reduce the attractiveness of the overall Schur complement algorithm since, instead of simply one decoupled elliptic solve on each of the sub-domains in steps 1 and 3, an iterative solution of step 2 would additionally involve an extra amount of elliptic solves on each of the sub-domains equal to the amount of iterations needed to converge the Schur complement system. Even though fancy preconditioning methods are abundant in the literature for the current situation (see f.e. References [28, 33, 34]), it is obvious that they will never outperform the *one-shot* direct LU backward and forward substitution process as proposed above.

We have therefore decided to explicitly generate and LU-factorize the Schur matrices, not before having convincingly shown that the memory requirements they impose are indeed realizable for their current application to LES and DNS of flows with one statistically homogeneous direction.

We next discuss the Schur complement construction procedure. For reasons of convenience, flexibility and the control it gives over accuracy, we have decided to construct the Schur matrix S using an existing parallel iterative solver package Aztec [35] (the package is freely available from the authors for research purposes), instead of using fast direct elliptic solvers on regular geometries. This is a highly performant parallel iterative solver package for the solution of linear systems that typically arise from a finite difference or element discretization of partial differential equations.

We recall that the two-dimensional Helmholtz problems to be solved on the complex domain given in (18) differ only in their value of the Helmholtz entry λ_m. Also from Equation (19) it follows that $\lambda_{2m-1} = \lambda_{2m}$, $\forall m = 1, \ldots, (N_y - 1)/2$, implying that the diagonal contribution to the discretized matrix due to the Helmholtz entry, and as such the discretized matrix corresponding to (20), only change for every second Fourier mode. As a consequence also the corresponding Schur complement matrices $S(\lambda_m)$ will only differ every second Fourier mode so that only $N_y/2$ of them need to be constructed, LU-factorized and stored as part of a preprocessing step.

If we merge all N_Γ interface unknowns together in one block we can symbolically rewrite system matrix (21) in the following block form:

$$\begin{pmatrix} A_{11}(\lambda_m) & \ldots & 0 & A_{1\Gamma} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & A_{NN}(\lambda_m) & A_{N\Gamma} \\ 0 & \ldots & 0 & S(\lambda_m) \end{pmatrix} \begin{pmatrix} \mathbf{x}_{\Omega_1} \\ \vdots \\ \mathbf{x}_{\Omega_N} \\ \mathbf{x}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \\ \tilde{\mathbf{b}}_\Gamma \end{pmatrix} \tag{27}$$

where

$$S(\lambda_m) = A_{\Gamma\Gamma}(\lambda_m) - \sum_{k=1}^{N} A_{\Gamma k} A_{kk}^{-1}(\lambda_m) A_{k\Gamma} \tag{28}$$

$$\tilde{\mathbf{b}}_\Gamma = \mathbf{b}_\Gamma - \sum_{k=1}^{N} A_{\Gamma k} A_{kk}^{-1}(\lambda_m) \mathbf{b}_k \tag{29}$$

Here, the off-diagonal block entries $A_{\Gamma k}$ and $A_{k\Gamma}$ are independent of the Fourier mode and are very sparse matrices.

We then construct each of the $S(\lambda_m)$s Schur complement matrices in a columnwise fashion as follows using expression (28):

$$\text{column } j \text{ of } S(\lambda_m) = S(\lambda_m).\mathbf{e}_j = A_{\Gamma\Gamma}(\lambda_m).\mathbf{e}_j - \sum_{k=1}^{N} A_{\Gamma k} A_{kk}^{-1}(\lambda_m) A_{k\Gamma}.\mathbf{e}_j \tag{30}$$

where $\mathbf{e}_j$ is a $(N_\Gamma \times 1)$ unit column vector with only zero entries except for unit entry at its jth row. The $A_{k\Gamma}.\mathbf{e}_j = \mathbf{f}_j$ part is computed using a sparse parallel matrix–vector product. Then, since $A_{kk}^{-1}(\lambda_m)\mathbf{f}_j = \mathbf{x}_j \iff A_{kk}(\lambda_m)\mathbf{x}_j = \mathbf{f}_j$ we can obtain $\mathbf{x}_j$ through a parallel iterative solve with RHS $\mathbf{f}_j$. This is followed by an additional sparse parallel matrix–vector product $A_{\Gamma k}\mathbf{x}_j$ before adding the result of another sparse parallel matrix–vector product $A_{\Gamma\Gamma}(\lambda_m).\mathbf{e}_j$ to obtain finally the jth column of $S(\lambda_m)$. This is repeated for each of the N_Γ columns of $S(\lambda_m)$.

Using a parallel iterative solver which employs double precision reals in this preparation phase allows the control over the accuracy of the Schur matrix construction and at the same time is a convenient way to distribute in parallel the cost over all processors, even though the latter factor is not of primary importance. In fact previous attempts using single precision fast direct elliptic solvers for the $A_{kk}(\lambda_m)\mathbf{x}_j = \mathbf{f}_j$ solves in the Schur complement matrix construction phase were found to suffer from a lack of accuracy in the overall Schur complement solution algorithm for certain grids and sub-domain configurations and were therefore dismissed for reasons of inconsistency. ILU-preconditioned restarted GMRES in combination with a very strict stopping criterion is invariably used in this Schur construction phase for reasons of stability and accuracy.

Subsequently, the Schur matrices are LU-factorized once and for all using the dedicated Lapack subroutine DGETRF and stored in memory. Here a problem arises however w.r.t. the Schur complement matrix S corresponding to the first Fourier mode $\lambda_0 = 0$ which is singular as we show next.

We know that the discretized two-dimensional Poisson matrix $A(\lambda_0)$, which is accompanied by a combination of Neumann and periodic boundary conditions, is singular. Furthermore, from the equivalence of the system matrices (21) and (27) it follows that:

$$|A(\lambda_0)| = |A_{11}(\lambda_0)| \;.\; |A_{22}(\lambda_0)| \;.\; \ldots \;.\; |A_{NN}(\lambda_0)| \;.\; |S(\lambda_0)| \tag{31}$$

where $|.|$ denotes the matrix determinant. Now, each of the A_{ii} block matrices represents the discretization of a Poisson matrix on sub-domain Ω_i with Dirichlet boundary conditions on its internal boundaries and is therefore non-singular. As a consequence from relation (31) we therefore find that $S(\lambda_0)$ inherits the singularity. Therefore, any LU-factorization attempt of the original $S(\lambda_0)$ matrix will break down due to a zero pivot in the last elimination step.

Recall that this singularity is linked to the indeterminacy of the pressure solution, i.e. if $\bar{p}$ is a solution then for every constant C also $\bar{p} + C$ will be a valid solution. And the existence of a solution is guaranteed by satisfaction of the compatibility condition by the RHSs. We have cured for the singularity in the present study by increasing the diagonal dominance of the $A_{\Gamma\Gamma}(\lambda_0)$ submatrix in Equation (28) by replacing its upper-left component $(A_{\Gamma\Gamma})_{1,1}$ with $3.(A_{\Gamma\Gamma})_{1,1}$. This increases the diagonal dominance of the system matrix and therefore avoids the singularity and breakdown of the LU-factorization described above. Note that we are allowed to apply this modification since it can be shown to be equivalent with choosing the solution of the one-parameter family $\bar{p} + C$ for which $(\bar{p}_{\Gamma\Gamma})_1 = (x_\Gamma)_1 = 0$.

4.4. Feasibility

We next discuss the storage requirements of the LU-factorization to check whether the approach is feasible in practice.

From its definition (25) we see that the resulting Schur complement matrices S are $(N_\Gamma \times N_\Gamma)$ matrices, where N_Γ is the total amount of unknowns that are located on the complete set of interfaces. To have an idea of the size of the interface problem let us take the straining example of a DNS calculation of the flow over a backward-facing step as described in Reference [36]. If we break up the domain into 3 sub-domains as schematically shown in Figure 3, separated by 2 interfaces, we find that for the most expensive grid they used we obtain a total of $N_\Gamma = 672$ interface unknowns. The storage for the Schur complement matrix then adds up to $N_\Gamma^2 = 0.45$ Mega words per two-dimensional problem. For the full three-dimensional problem we have to store $N_y/2 = \frac{64}{2} = 32$ of those LU-decompositions which lead us to a total of 14.5 Mega words.

Note that we did not exploit possible zero-block entries within the Schur complement matrices associated with the above example through a sparse storage scheme. Also, one could further reduce the storage cost by a factor of two by applying a co-ordinate transformation which symmetrizes the discretized pressure-Poisson equations (18) and therefore would only necessitate storing its upper triangular factor U instead of both its L and U factor. Additionally, one can also choose to only treat some of the worst conditioned Fourier modes with the Schur complement strategy and all others with a full iterative parallel solver as we will elaborate further in Section 5. Finally, in a distributed memory parallel environment we can distribute the storage load over the different available processors.

The above clearly demonstrates the feasibility of the approach, even for DNS purposes, and validates therefore our proposal for a direct solution of the Schur complement interface problems P_Γ.

5. LINEAR SYSTEM SOLUTION

The previous section has outlined, in detail, the essence of the multi-domain discretization of the elliptic kernel, in terms of a pair of decoupled problems: the interior unknowns problem P_I and the interface one P_Γ. Several issues related to the latter problem have already been discussed in Section 4.3; this section concentrates instead on problem P_I.

As anticipated, problem P_I consists of a set of decoupled, on a sub-domain basis, elliptic problems which can be solved by either direct or iterative methods, as discussed in the following sections.

5.1. Direct methods

With the above technology we have reduced the elliptic kernel to a set of elliptic equations in simple rectangular geometries. For those geometries, in the context of low order methods, we may profit from the developments carried out in the 1970s concerning the so-called FES [24, 25, 37, 38].

These are a family of direct solvers which employ various combinations of the fast Fourier transform (FFT), the cyclic reduction (CR) algorithm and one-dimensional Gaussian

elimination to achieve a close-to-optimal operation count of order $O(N \log N)$, where N is the total amount of grid points (in 3D sense). Also from a memory point of view they are highly optimal requiring the storage of less than N words. These solvers come in slightly different flavours in terms of the choice of the underlying grid (staggered vs non-staggered), boundary condition implementation, flexibility of the grid size and grid non-uniformity and the details of the actual underlying algorithm. For a good introduction to the various combinations of the FFT and CR algorithms typically employed we refer the reader to a study by Wilhelmson and Ericksen [38], which also presents a detailed operation count analysis of these solvers. The high speed that these solvers attain intrinsically relies on the exploitation of the regularity of the underlying problem. This limits however their applicability to the solution of the Poisson equation in box-like geometries. Uniform grid spacing is required in all spatial directions where the FFT algorithm is applied in contrast to the CR algorithm, which allows for non-uniformity of the grid.

For problems where grid non-uniformity is required in two directions (say the streamwise as well as the wall-normal directions) the above set of solvers no longer applies and we resort to the fast solver for general separable elliptic equations BLKTRI as developed by Swarztrauber [37].

5.2. *Iterative methods*

In order to be able to assess the performance of the Schur complement solution strategy against other existing methods, we have constructed an iterative solver which does not use any multi-domain strategy, i.e. it solves the elliptic problem as such, on the whole computational domain. Starting from the geometric configuration, we first assemble the matrix coefficients stemming from the spatial discretization, using a sparse distributed storage format. The resulting system is then solved with the state-of-the-art parallel library Aztec [35], which is specifically designed for the solution of large-scale linear systems. The library contains a wide sample of commonly applied Krylov subspace iterative solvers such as the conjugate gradients (CG) and the restarted generalized minimum residual method (GMRES) together with a large variety of preconditioning methods such as incomplete LU (ILU) or Cholesky (IC) factorization methods amongst other choices. The package is specifically designed for use with parallel distributed memory architectures and uses the message-passing interface (MPI) [39] to handle interprocessor communication. The different (sparse) BLAS libraries are embedded for reasons of portability and performance.

For a good introduction to the theory of preconditioned iterative methods with an emphasis on Krylov sub-space methods we refer the reader to Reference [40], whereas a more complete reference including issues related to its parallel implementation can be found in the book of Saad [41]. The major computational kernels of Krylov sub-space based iterative methods consist of

- preconditioner application $Pu = v$,
- sparse matrix–vector products $Ax = y$,
- vector updates $x = y + \alpha z$,
- inner products $x = y^t z$.

We are not concerned with the cost related to the generation of the preconditioner itself since it can be amortized over its repeated use for different RHSs. The effect of the preconditioning

operation is to improve the condition number of the discretized matrix system such as to reduce the amount of iterations needed for convergence. On the other hand, the application of the preconditioning increases the cost of the procedure per iteration, such that a good balance between quality and complexity has to be found.

In the current work we shall exclusively employ Saad's ILUT(fill,α) preconditioner [42] which uses two criteria for determining the fill-in in the resulting approximate factorizations. Fill-in relates to the generation of non-zero entries in locations which were originally zero and were therefore not stored in the sparse storage matrix representation. For e.g. fill $= 1.5$ requires that the ILUT factors contain no more than approximately 1.5 times the number of non-zeros of the original matrix such that for fill $= 0$ no additional fill-in beyond the non-zero structure of the original matrix is allowed. Additionally ILUT drops all elements in the resulting factors whose absolute value is less than α. This is a class of generally robust preconditioners which are commonly used in combination with GMRES for the solution of non-symmetric linear systems.

In a parallel context, the sparse linear system matrix is logically distributed over the processors. This is done on a row-wise basis. The accompanying vector components are allocated to the same processor with the same ordering. In the current implementation this distribution is induced by the multi-domain decomposition of the computational geometry. It is straightforward to define a sub-domain to processor mapping, so that all matrix rows corresponding to unknowns within a sub-domain are retained in the local submatrix of the processor to which this sub-domain is mapped. A further local reordering of the row indices may be performed in order to optimize MPI communication.

As a consequence of the distribution of the system matrix and accordingly all vectors over the processors, most of the major computational kernels of Krylov iterative solvers mentioned above, *viz.* matrix–vector products, preconditioning operators and inner products require MPI communication across processors and will therefore influence the parallel performance of the procedure. Vector updates on the other hand can be fully locally computed.

A parallel extension of the ILUT set of preconditioners introduced above is achieved by using them in a domain decomposition framework. Either the ILUT(fill, α) strategy is applied strictly to the submatrix attributed to the local processor (zero overlap) or alternatively some overlap between sub-domains can be allowed by applying the ILUT(fill, α) strategy to the local submatrix which is recursively augmented by the rows of external unknowns which appear in the discretization stencil of the previous version of the augmented matrix system. This localization of the preconditioner will negatively influence the ability of the preconditioning to reduce the condition number of the global preconditioned system, but it allows some control over the amount of MPI communication involved in the application of the preconditioning operator which might be more crucial from a cost point of view than limiting the amount of iterations for convergence.

As in the case of the Schur complement method we will apply this parallel iterative solver on each of the two-dimensional decoupled Helmholtz equations (18) that resulted from the Fourier expansion of the originally three-dimensional pressure-Poisson equation (17).

5.3. *Hybrid methods*

As already anticipated in Section 4.2 the Schur complement method can be an effective manner for reducing the stiffness of a multi-domain problem to the one which pertains to the

decoupled sub-domain problems. Thus, to check the effectiveness of this opportunity we have also considered the possibility of solving problem P_I with the previously introduced Aztec iterative solver.

Although quantitative cost estimates could be given both for the direct and iterative approaches, we have preferred to assess their relative performance *in situ*, that is exactly on the same multiple grid system for which LES results will be provided in the companion paper [9].

6. PERFORMANCE EVALUATION

In this section we compare the performance of the different pressure-Poisson solvers that were implemented as part of the current study. They are summarized in Table I.

First of all, we have the fast direct Poisson solver discussed in Section 5.1 and applicable to grids which can be stretched in both streamwise and wall-normal directions (DIR2STR). We have seen in Section 4 that the above solver can be exploited in the framework of the Schur complement method to perform the decoupled sub-domain solves that occur in steps 1 and 3 of the algorithm. We shall refer to the implementation of this solver combination as SCMDIR. As already mentioned, the Schur complement method could also be an effective manner for reducing the stiffness of a multi-domain problem to that of the decoupled sub-domain problems. To test this hypothesis a second Schur complement solver option (SCMITER) was implemented using the previously introduced Aztec iterative solver in steps 1 and 3 for the resolution of the decoupled sub-domain problems instead of the fast direct solver. The implementation of the parallel iterative Aztec solver previously introduced in Section 5.2 will be referred to as ITERPAR. Recall that this solves the multi-domain Poisson linear system without any resort to Schur complement strategies. As a result the block matrices corresponding to the respective sub-domains remain fully coupled.

Finally a solver option combining SCMDIR and ITERPAR was implemented. Recalling the Fourier expansion in the spanwise direction of the originally three-dimensional pressure-Poisson problem into a set of two-dimensional Helmholtz problems which depend on a Helmholtz factor λ_m, $\forall m = 0, \ldots, ny - 1$ as described by Equations (18) and (19), this solver option allows to select a Fourier mode m^* such that all modes with $m \leqslant m^*$ will be solved with SCMDIR, and the remaining ones with ITERPAR. This is based on the observation that a straightforward iterative solver procedure will have a tough job converging the Helmholtz problems corresponding to the ill-conditioned lowest wave number Fourier modes, whereas the increasing diagonal dominance of the discretized matrix generated by the Helmholtz factors λ_m associated with higher wave number Fourier modes will quickly improve the condition number of the underlying matrix and therefore render an iterative procedure more opportune. This solver combination will be referred to as SCMFLEX.

Table I. Summary of implemented Poisson solver options.

DIR2STR	Fast direct single domain (2D stretching)
SCMDIR	Schur compl. multi-domain + DIR2STR sub-domain solves
SCMITER	Schur compl. multi-domain + iterative sub-domain solves
ITERPAR	Plain parallel preconditioned iterative multi-domain
SCMFLEX	Flexible combination of SCMITER and ITERPAR

The performance tests will be done both in a serial and a parallel context. All parallel tests were done on an 8 processor symmetric multi-processor Sun HPC 3500 shared memory machine consisting of eight 400 MHz UltraSPARC II processors with 8 Gb of shared memory and a so-called 2.6 Gb/s Gigaplane system interconnect running Solaris 2.7 and Sun MPI. Primary cache consists of 16 Kb instruction and 16 Kb data on chip, whereas the secondary external cache is 8Mb. The nominal peak performance is 6.4Gflops (400MHz $\times 8$ processors $\times 2$ flops per cycle). We are well aware that this machine is, presently, being out-moded by the current trend towards PC-based clusters, but we consider the number of processors employed sufficient to assess the behaviour of present algorithms in parallel environments, while quoted CPU values remain a valid basis of comparison for performance of different algorithms tested herein. To avoid drawing conclusions from a set of well-defined theoretical test cases which are not necessarily relevant for our current interest in LES of complex turbulent flows we have opted to do the performance measurements on *real-life* large-eddy simulations for which we will present the actual statistical results in the companion paper [9].

6.1. Single domain channel grid

Before determining the performance of the respective Poisson solution methods in a multi-domain context we first compare some of them on a grid used for single domain wall-resolved fully developed turbulent channel calculations. The regularity of the box-like geometry and uniformity of the grid in the streamwise and spanwise directions allows to apply directly, without resorting to any Schur complement strategy, the family of so-called *fast direct Poisson solvers* here represented by DIR2STR. As mentioned, the solver allows for grid non-uniformity in two directions (but one would be enough in this particular case) and sequentially applies fast Fourier transformation, cyclic reduction and Gaussian elimination, with an optimal operation count. The run time comparisons are obtained on $(nx, ny, nz) = (64, 128, 64)$ grid, amounting to 524 288 points in the three-dimensional sense, whereas each of the 128 two-dimensional Helmholtz problems corresponds to an elliptic problem of size $64 \times 64 = 4096$. The timings were obtained by advancing the code over 100 time steps also making sure that the flow field was in the regime where statistical sampling could be started to guarantee the physical correctness of the flow.

In Table II, we present two separate timings for this case. First of all the timings of the Poisson solver part, referred to hereafter as *Poisson*, which includes all contributions due to the pre- and post-processing involved in the application of the FFT and IFFT algorithms, naturally the solution of the Poisson equation itself, as well as the final update of the pressure correction defined by Equation (13) and the imposition of boundary conditions on the newly computed pressure field. The second timer, referred to hereafter as *parallel*, presents the timings of the remaining part of the LES solver around the pressure-Poisson solution part,

Table II. Single domain $(64, 128, 64)$ grid channel timings.

	Poisson	Parallel	Total LES	Poisson (%)
DIR2STR	2 min 14 s	2 min 36 s	4 min 50 s	46
SCMDIR	4 min 22 s	2 min 36 s	6 min 58 s	63
ITERPAR	3 min 52 s	2 min 36 s	6 min 28 s	60

i.e. the explicit time-advancement part, including the dynamic evaluation of the time step, the repeated application of boundary conditions, as well as the run-time evaluation of various statistical quantities. All these tasks are independent on a sub-domain basis except, of course, the boundary condition part, and are therefore inherently parallel. For reasons of convenience the SGS model was switched off altogether during all the timing measurements even though this task could be included in the *parallel* timer part since it is computed on a sub-domain basis. This should be taken into account when relating the *Poisson* timer to the timings of the overall LES solver (as a guideline, the Smagorinsky SGS model will typically increase the cost of the complete LES solver by 20–30%). The sum of both timers is included as well and corresponds to the full run-time performance of the LES code (minus the SGS contribution), excluding only the cost related to pre-processing steps, such as input related to the grid definition and initial conditions, as well as occasional output of intermediary result and restart files. Finally, in the last column the cost of the Poisson solve is related to the total cost of the LES procedure. To test solver option SCMDIR for the current single domain grid we artificially defined a Schur interface Γ at the connecting periodic streamwise outflow boundary. The corresponding timings have been included in Table II. They nicely demonstrate the doubling of the Poisson solution time as suggested by the Schur complement algorithm. It was found that for the current size of the Schur interface corresponding to $n_\Gamma = 64$ unknowns the cost of the LU back-substitution phase remains invisible. Finally, Table II includes the timings of solver option ITERPAR for the current single domain grid. The iterative solver selected is the restarted GMRES in combination with Saad's ILUT(α, fill) preconditioner with a dual dropping strategy, where $\alpha = 1.0e-5$ and fill $= 4.0$. The restart value was set to 40. Lowering the initial residual by a factor $5.0e-3$ was found to be sufficient as stopping criterion to obtain good accuracy. It was observed that for the current choice of the preconditioner typically every Fourier mode m converged in 1 iteration except for the lowest wave number which needed 2. This indicates the excellent quality of the preconditioner for the current case. Allowing no fill-in nor dropping of small entries in the ILUT preconditioner, the Poisson run-time was increased by 13%.

We conclude that, for the current single domain channel grid, the fast direct solver clearly outperforms the preconditioned iterative solver, despite the excellent convergence characteristics obtained with the ILUT preconditioner. This further motivates our search to extend the use of fast direct Poisson solvers to a multi-domain context.

6.2. *Multi-domain channel grid*

In order to study the relative performance of the different Poisson solver methods in a parallel context the previous single domain resolved channel grid was divided into 8 equally sized sub-domains as shown in the (x,z)-plane representation of Figure 4. On the same figure the definition of all Schur interfaces, which allow the decoupling along the sub-domains, is indicated, as well as the numbering of the sub-domains which decides their distribution over the processors. This multi-domain breakup leads to a grid size of $(nx, ny, nz) = (16, 128, 32)$ per sub-domain and allows a perfect load balancing in the case of a 1, 2, 4 or 8 processor parallel calculation. This amounts to 65 536 grid points in a three-dimensional sense, whereas each of the 128 two-dimensional Helmholtz problems corresponds to an elliptic problem of relatively small size $16 \times 32 = 512$ per sub-domain. The total amount of interface unknowns n_Γ introduced this way amounts to 316 for each of the two-dimensional Helmholtz problems

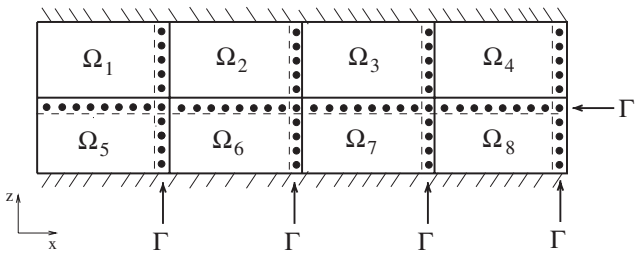

Figure 4. Multi-domain channel breakup into 8 equally sized sub-domains including definition of Schur interfaces.

Table III. Multi-domain $8 \times (16, 128, 32)$ grid channel timings for ITERPAR.

No. of processors	Poisson	Parallel
1	3 min 56 s	3 min 6 s
2	2 min 41 s	1 min 30 s
4	1 min 36 s	43 s
8	1 min 14 s	22 s
8 processor parallel speedup	3.2	8.4
8 processor parallel efficiency	40%	105%

that result from Fourier decomposing the three-dimensional pressure-Poisson problem in the periodic spanwise direction. The extremely low ratio of inner unknowns to Schur interface unknowns equaling 12.96 for the single processor and barely 1.6 for the 8 processor calculation will clearly have a significant negative effect on the ratio of interprocessor data communication to true computation inside each of the sub-domains, as well as on the cost of the sequentially implemented LU back-substitution in step 2 compared to the parallelized steps 1 and 3 of the Schur complement algorithm. As such, this case represents a very demanding test in terms of achieving a good parallel speedup and efficiency. The parallel performance measurements were obtained on 1, 2, 4 and 8 processors of an 8 processor Sun E3500 shared memory architecture. Again, it could not be guaranteed that all other processors were free of other tasks during the CPU timings. The effect of this could potentially be bigger than for the serial runs in light of the increased data-transfer between the processors; nevertheless, it was not judged as too important. Again, all timings were typically repeated to filter out possible discrepancies linked to a variable processor occupation. The timings were obtained by advancing the code over 100 time steps, also making sure the flow field was in the regime where statistical sampling could be started to guarantee the physical correctness of the flow.

We first present in Table III the results obtained for the ITERPAR solver. As in the single domain case, the iterative solver selected is restarted GMRES in combination with an ILUT(α, fill) type of preconditioner with a dual dropping strategy, where $\alpha = 1.0E - 5$ and fill $= 4.0$. No overlapping between sub-domains was allowed in the construction of the preconditioner, since it was found not to bring any improvements in run-time. The restart value was set to 40. Lowering the initial residual by a factor $5.0e - 3$ was again found to be sufficient as stopping criterion to obtain good accuracy. The parallel speedup and efficiency of the

Table IV. Further investigation into ITERPAR solver parallel behaviour.

No. of procs	Total no. of iters	Total time	No. of iters/s	No. of iters/s/No. of procs
1	12 878	205	62	62
2	23 528	146	161	80
4	23 528	88	266	66
8	23 828	63	378	47

LES code excluding the Poisson solution part is perfect, indicating that the cost of the MPI interprocessor data communication due to boundary exchanges is comparable to the cost of the boundary exchanges in a single processor calculation. Here, we adopted the usual notions of parallel speedup $S(n,P)$ and parallel efficiency $E(n,P)$

$$S(n,P) \equiv \frac{T(n,1)}{T(n,P)} \tag{32}$$

$$E(n,P) \equiv \frac{S(n,P)}{P} = \frac{T(n,1)}{PT(n,P)} \tag{33}$$

where n denotes the problem size, P the amount of processors involved, $T(n,1)$ and $T(n,P)$ the execution times on 1 and P processors respectively.

The speedup of the ITERPAR solver shows a much less favourable picture. This can be, first of all, attributed to the effect the parallelization has on the quality of the preconditioner. It was observed that for the current choice of preconditioner for the single processor run, typically every two-dimensional problem converged in 1 iteration except for the lowest wave number which needed 2. This again confirms the excellent quality of the current preconditioner. This was no longer the case for the multiple processor runs which typically took 6 iterations for the lowest frequency two-dimensional problem to converge, gradually going down to 1 iteration for the highest frequency modes. This deterioration in the preconditioning effectiveness is linked to the localized character of the constructed preconditioner in which no overlap between sub-domains was allowed. Another factor decreasing the parallel efficiency is of course the MPI interprocessor data communication associated with the non-local computational kernels appearing in the GMRES iterations, *viz.* the matrix–vector multiplications, inner products and the application of the preconditioner.

To investigate both of the above arguments more accurately we monitored the total amount of iterations over the 100 time steps and over all two-dimensional Helmholtz problems for each of the runs $(1,2,4,8$ processors), as well as the raw timings of the Aztec GMRES solver call. They follow below, in the second and third column of Table IV. From the second column we can clearly observe the deterioration in the preconditioner effectiveness as soon as more than 1 processor is involved. Also, since the total amount of iterations for 2 and 4 processors is the same, the timings from column 3 should in principle differ by a factor of 2 in the absence of any MPI communication. This is not the case, putting into evidence the cost related to increased MPI communications. It is remarked that due to the sub-domain ordering given in Figure 4, in conjunction with the current non-overlapping preconditioner, the preconditioners constructed in the case of 2 and 4 processors should indeed be exactly the same and consequently also the amount of iterations for convergence for 2 and 4 processors.

Table V. Multi-domain $8 \times (16, 128, 32)$ grid channel timings for SCMDIR.

No. of procs	Poisson	LU backs	Net Poisson	Parallel
1	5 min 8 s	1 min 8 s	3 min 50 s	3 min 6 s
2	3 min 17 s	1 min 8 s	1 min 59 s	1 min 30 s
4	2 min 17 s	1 min 8 s	1 min 1 s	44 s
8	1 min 54 s	1 min 10 s	38 s	22 s
$S(n,8)$	2.7		6.1	8.4
$E(n,8)$	34%		76%	105%
No. of procs	Schur step 1		Schur step 2	Schur step 3
1	1 min 46 s		1 min 18 s	1 min 54 s
2	53 s		1 min 18 s	1 min 1 s
4	26 s		1 min 16 s	32 s
8	14 s		1 min 16 s	20 s
$S(n,8)$	7.6			5.7
$E(n,8)$	95%			71%

The combined influence of the MPI cost and the worsening of the preconditioner is nicely seen in the gradual decrease in the amount of iterations per second per processor for $2, 4$ and 8 processors. Somehow strange however is the low value 62 obtained for the single processor calculation. Most likely this is due to other GMRES related cost factors. The almost immediate convergence (1–2 iterations) may well prevent a good sustained flop rate for the underlying GMRES algorithm.

We proceed presenting the timing results for the SCMDIR solver for $1, 2, 4$ and 8 processors on the current multi-domain arrangement in Table V above. Note that in Table V the *Schur step* 1, 2 and 3 timers correspond to the complete implementation of the different steps inside the Schur complement algorithm, so including all pre- and post-processing steps, such as e.g. the generation of the RHSs, which might be involved in the actual implementation. This is in contrast to the *LU backs* timer which times only the *raw* call to the machine-optimized Lapack LU back-substitution routine DGETRS. As a consequence, we logically find that *Schur step* $2 > LU\ backs$ and $Poisson > Schur\ step\ 1{+}2{+}3$. This has to be kept in mind when observing the timings. As explained before, the second step of the Schur complement algorithm has not been parallelized since it was expected that the LU back-substitution cost would be negligible w.r.t. the cost of the rest of the algorithm and as such would not be worth parallelizing. So, in the current implementation every processor performs the full LU back-substitution task for each of the two-dimensional Fourier modes. The above timings reveal, however, that for the current problem the overall cost of the LU back-substitution in step 2 of the algorithm is of the same order of magnitude as the cost of the direct decoupled sub-domain solves in steps 1 and 3 of the algorithm for the calculation on a single processor. Therefore, the serial fraction of the algorithm is certainly non-negligible for this case. As a result of Amdahl's law which defines the following maximal limit for the parallel speedup:

$$S(n,P) \leqslant \frac{1}{\alpha + (1-\alpha)/P} \leqslant \frac{1}{\alpha} \tag{34}$$

Table VI. Multi-domain $8 \times (16, 128, 32)$ grid channel timing comparisons.

No. of procs	ITERPAR Poisson	SCMDIR Poisson	SCMDIR theor.
1	3 min 56 s	5 min 8 s	5 min 8 s
2	2 min 41 s	3 min 17 s	2 min 38 s
4	1 min 36 s	2 min 17 s	1 min 21 s
8	1 min 14 s	1 min 54 s	48 s
$S(n, 8)$	3.2	2.7	6.4
$E(n, 8)$	40%	34%	80%

where $0 \leqslant \alpha \leqslant 1$ refers to the fraction of the computation which is performed sequentially, this inhibits dramatically an efficient parallelization of the overall algorithm as is evidenced by the low parallel efficiency result of the Poisson timer. This situation could however be effectively remedied for by parallelizing w.r.t. the spanwise direction this step of the algorithm. Instead of solving the Schur interface problem for all Fourier modes bluntly on each processor one could distribute the N_y decoupled Schur interface problems evenly over all processors at the small extra MPI communication cost linked to distributing the Schur interface solution back to all processors.

Based on these comments, we concentrate on the parallel performance of the algorithm excluding the LU back-substitution step as represented by the *net Poisson* timer in the above table. This is simply defined as the Poisson timer minus the Schur step 2 timer. This shows a much more favourable picture nevertheless not attaining completely satisfactory values, mainly because of a reduced parallel efficiency in step 3 of the algorithm. The main MPI communication overhead in step 3 is linked to the generation of its RHS, which requires a matrix–vector product of a sparse distributed matrix with a dense interface solution vector. The reduced efficiency may be linked to the current extremely low ratio of inner unknowns relative to Schur interface unknowns.

We can also derive ideal theoretical timings of the complete Poisson solution step in case the LU back-substitution phase would have been parallelized as described above. For this we simply add the (*serial cost of Schur step* 2)/(*amount of processors*) to the *net Poisson* part. The comparison of this theoretical result with the previously obtained ITERPAR and SCMDIR timings is presented in Table VI. We conclude that, for the current grid, the ITERPAR solver outperforms the Schur solvers in case of a single processor calculation. Recall, however, that for a single processor calculation for the current rectangular geometry, one can simply apply the fast direct solvers on a single domain without resorting to Schur complement multi-domain breakup strategies. So the interest in a multi-domain breakup is directly coupled to the intention for a parallel computation for the current geometry. From the multiple processor timings we conclude that, for the current range of processors, the ITERPAR and SCMDIR outperform the other depending on whether the LU back-substitution step is parallelized or not. The much higher parallel efficiency obtained with the theoretical SCMDIR suggests that its better performance than ITERPAR will only increase with the amount of processors.

Finally, we present in Table VII the performance results for the SCMITER solver for 1, 2, 4 and 8 processors on the current multi-domain arrangement. The parallel efficiency of Schur step 1 and 3 attains much lower values than the ones for the SCMDIR case. This is quite surprising, in view of the explicit decoupling achieved by the Schur complement algorithm.

Table VII. Multi-domain $8 \times (16, 128, 32)$ grid channel timings for SCMITER.

No. of procs	Poisson	LU backs	Net Poisson	Parallel
1	8 min 26 s	1 min 15 s	6 min 58 s	3 min 7 s
2	5 min 21 s	1 min 12 s	3 min 56 s	1 min 29 s
4	3 min 42 s	1 min 11 s	2 min 23 s	44 s
8	3 min 10 s	1 min 11 s	1 min 52 s	23 s
$S(n, 8)$	2.7		3.7	8.1
$E(n, 8)$	33%		47%	102%
No. of procs	Schur step 1	Schur step 2		Schur step 3
1	3 min 26 s	1 min 28 s		3 min 22 s
2	1 min 54 s	1 min 25 s		1 min 57 s
4	1 min 7 s	1 min 19 s		1 min 12 s
8	51 s	1 min 18 s		58 s
$S(n, 8)$	4.0			3.5
$E(n, 8)$	50%			44%

However, in its present implementation the decoupled sub-domain problems are gathered in one big system matrix in Aztec distributed sparse storage format which is solved via parallel GMRES. This effectively means that steps like checking the current residual (globally over all processors) and the Householder orthogonalization process will still involve a fair deal of MPI communication, despite the decoupling.

Besides the rather poor parallel efficiency also the anticipated positive effect an explicit decoupling would have on the quality of the preconditioner is not very pronounced. Comparing the Schur step 1 column of Table VII with the ITERPAR Poisson timer of Table III, one observes only a modest improvement. This is however influenced greatly by the more restrictive stopping criterion that was applied in the current case, *viz.* $1.0e-5$ instead of $5.0e-3$ used in the ITERPAR case. This increased tolerance was found necessary for accuracy in the current 3-stage Schur complement solution procedure where the accuracy of each step also depends on the input it receives from the previous step. Then again, as we have discussed before, the preconditioning operator in the ITERPAR case was already very performant such that there was no room for major improvement in the first place.

6.3. Multi-domain backward-facing step grid

In the current section once more we use a test case for which the LES statistical results will be presented in full detail in the companion paper [9]. The details on the grid design will also be given elsewhere [9], but for the current purposes it is sufficient to refer to Table VIII in combination with Figure 5 to summarize its multi-domain composition. From that table it can be seen that the total amount of grid points sum up to 405 696 in a three-dimensional sense, whereas each of the 32 two-dimensional Helmholtz problems correspond to an elliptic problem over 12 678 unknowns. The total amount of interface unknowns n_Γ introduced this way amounts to 207 for each of the two-dimensional Helmholtz problems that result from Fourier decomposing the three-dimensional pressure-Poisson problem in the periodic spanwise direction. In the current case the ratio of inner unknowns per sub-domain to Schur interface unknowns varies from 4.0 for the smallest sub-domain Ω_3 to 18.8 for the largest Ω_5, which is

Table VIII. Computational domain sizes and grid definition of 5 sub-domain BFS geometry.

Sub-domain	L_x	L_y	L_z	N_x	N_y	N_z
Ω_1	9	4	1.13	90	32	21
Ω_2	9	4	5	90	32	32
Ω_3	2	4	5	26	32	32
Ω_4	20	4	5	122	32	32
Ω_5	20	4	1	122	32	26

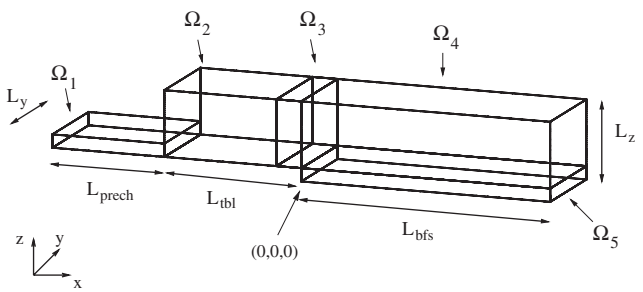

Figure 5. Multi-domain BFS computational box composition, axis not to scale.

already substantially better than the value 1.6 found for the previously discussed multi-domain channel case. Consequently the ratio of the cost of the sequentially implemented LU back substitution in step 2 to the cost of the parallelized steps 1 and 3 of the Schur complement algorithm should be much more reduced compared to the multi-domain channel case. Besides the above grid summary it has to be realized that the grid is highly stretched in both the streamwise and the wall-normal direction to accommodate a variation of the turbulent length scales throughout the computational domain. The stretching will have a negative effect on the stiffness of the original non-preconditioned pressure-Poisson discretized matrix rendering an iterative solution procedure more difficult. Also important to note is that the current BFS grid is composed out of 5 sub-domains with quite varying grid sizes so that, in contrast to the 8 sub-domain channel grid, the load-balancing for computation on a parallel architecture will be far from ideal. As such, parallel speedup and efficiency numbers will be seriously hampered from the start. Nevertheless, we present in the following a performance comparison between the different available Poisson solution options which are applicable to this case, for serial as well as parallel calculations. The serial calculation performance measurements were obtained on a single processor of the Sun E3500 shared memory architecture, whereas the parallel timings were done using 5 of its processors such that each sub-domain is allocated to a different processor. Again, it could not be guaranteed that all other processors were free of other tasks during the timings, which might affect to some degree the run-time performance. All timings were, typically, repeated, to filter out possible discrepancies related to a variable processor occupation. The timings were obtained by advancing the code over 100 time steps also making sure the flow field was in the regime where statistical sampling could be started to guarantee the physical correctness of the flow. In the following we only compare solver

Table IX. Multi-domain backward-facing step timings for ITERPAR.

No. of processors	Poisson	Parallel
1	4 min 9 s	2 min 22 s
(5)	(6 min 50 s)	(44 s)
5	3 min 44 s	44 s
$S(n,5)$	1.1	3.2
$E(n,5)$	22%	65%

options SCMDIR, SCMITER and ITERPAR, since the single domain direct solvers are not applicable to the current geometry. The iterative solver selected in case of SCMITER or ITERPAR is restarted GMRES in combination with an ILUT(α, fill) type of preconditioner with a dual dropping strategy where $\alpha = 1.0e-5$ and fill $= 4.0$. The restart value was set to 40. Lowering the initial residual by a factor $5.0e-3$ was found to be sufficient as stopping criterion to give good accuracy in case of ITERPAR. However, in the case of SCMITER the criterion had to be lowered as before to a value of $1.0e-5$ in order to achieve the same level of accuracy.

We first discuss the timings obtained with the ITERPAR solver which are summarized in Table IX above. It was observed, that in case of a serial run with solver ITERPAR, typically every Fourier mode converged in 1–2 iterations except for the lowest wave number, which needs roughly 10. This indicates again that the preconditioner is quite performant. In case of a 5-processor run initially, as it was the case for the channel simulation, no overlapping between sub-domains was used for the construction of the ILUT preconditioner. For this however it was observed that the GMRES restart size had to be increased to 80 to obtain convergence for all Fourier modes within 300 iterations and that the lowest frequency mode typically takes more than 200 iterations to converge, whereas all other modes took from 20 for the lower to typically 6 iterations for the highest frequency modes. The result was a very poor *Poisson* timer result, reported in Table IX between brackets, even exceeding by far its serial counterpart.

The quality of the non-overlapping ILUT-preconditioner constructed, therefore, proves itself to be largely inadequate for parallel purposes. Overlapping will improve the quality of the preconditioner and as such reduce the amount of iterations needed for convergence, but at the same time it will augment the cost per iteration step due to the increased communication between processors. It was found that an overlap of 2 cells between the sub-domains for the construction of the ILUT preconditioner resulted in the best *Poisson* timings which are retained in Table IX. The overlapping has, clearly, a very favourable effect on the Poisson solution time roughly halving it w.r.t. the non-overlapped case. Nevertheless, the speedup w.r.t. the serial case remains only marginal. Observe that, unlike the resolved channel case, the *parallel* timer part does not show a perfect parallel speedup. This can be fully attributed to the poor load-balancing of the current multi-domain decomposition, however.

We proceed with the timing results obtained with the SCMDIR solver for a single and 5-processor calculation. The global results, as well as the details of computational load for different steps are summarized in Table X above. We observe that the Schur complement method in combination with a direct sub-domain solver clearly outperforms the ITERPAR

Table X. Multi-domain backward-facing step timings for SCMDIR.

No. of procs	Poisson		Parallel
1	2 min 48 s		2 min 18 s
5	59 s		44 s
$S(n,5)$	2.8		3.1
$E(n,5)$	57%		63%
No. of procs	Schur step 1	Schur step 2	Schur step 3
1	1 min 11 s	13 s	1 min 17 s
5	21 s	10 s	24 s
$S(n,5)$	3.4		3.2
$E(n,5)$	67%		64%

Table XI. Multi-domain backward-facing step timings for SCMITER.

No. of procs	Poisson		Parallel
1	9 min 47 s		2 min 18 s
5	2 min 41 s		44 s
$S(n,5)$	3.6		3.1
$E(n,5)$	73%		63%
No. of procs	Schur step 1	Schur step 2	Schur step 3
1	4 min 56 s	17 s	4 min 27 s
5	1 min 21 s	9 s	1 min 9 s
$S(n,5)$	3.7		3.9
$E(n,5)$	73%		77%

options, both in the serial and 5-processor calculation. Also, the parallel speedup and efficiency numbers of the net Poisson part compares very favourably with the ones of the *parallel* timer. The observation that in a serial calculation ITERPAR converges very fast for all Fourier modes except for the lowest wave number, coupled to the out-performance of the SCMDIR solution method suggests to combine both methods as implemented in solver SCMFLEX with $m^* = 1$, i.e. use SCMDIR only for the lowest Fourier mode and ITERPAR for all others. This was also tried for a single processor calculation giving 3 min 30 s for the *Poisson* timer part. So this flexible combination does not outperform SCMDIR proving that the Schur complement method coupled to a direct solution of the decoupled sub-domain problems is a highly optimized algorithm even for the highest Fourier mode.

Finally, we present the results obtained with the SCMITER solver for the current backward-facing step case in Table XI. Despite its surprisingly good parallel speedup and efficiency numbers for the *Poisson* timer, the SCMITER solver is still 3 times slower than the SCMDIR solver for the 5-processor calculation. The combined results of Tables IX, X and XI show convincingly that for the current flow geometry the decoupling of the sub-domain problems achieved via the Schur complement method is highly desirable, especially in, but not limited to, a parallel context.

7. CONCLUSIONS

A non-overlapping domain decomposition strategy for the solution of the elliptic kernel associated with the fractional step procedure and/or the semi-implicit discretization of the incompressible Navier–Stokes equations in complex geometries has been presented. The power of the Schur complement method has been demonstrated, especially when applied in conjunction with fast direct elliptic solvers, both in a serial and parallel computing context. The run-time results obtained with the iterative (on a sub-domain basis) solver were not as convincing as anticipated, mainly because of the relatively strict convergence criterion which was found to be necessary in problem P_I (steps 1 and 3) of the Schur algorithm for a good overall accuracy. Besides that, its current implementation combining the inherently parallel decoupled sub-domain solves into one big distributed parallel Aztec solve negatively influences the speedup numbers because of unnecessary MPI communications.

The more straightforward approach to solve elliptic problems on the current class of complex domains which use parallel iterative preconditioned Krylov sub-space methods, here implemented using the Aztec package, in general, shows a fairly good performance for serial calculations, whereas, in a parallel context, the issue of inadequate preconditioning can quickly degrade its performance. This can be directly linked to the trade-off, which seems unavoidable between the extra MPI communication cost involved to allow for a sufficient amount of overlapping in the construction of the preconditioning operators to preserve some of its global convergence characteristics, as opposed to domain decomposition based preconditioners with relatively little overlap which need more iterations to reach convergence.

In the literature, several methods have been proposed for restoring the global convergence characteristics of parallel preconditioned iterative solution methods via the introduction of a low cost coarse-level globally acting reduced version of the elliptic operator in some stages of the iterative procedure, much in line with the philosophy behind multigrid methods. Most of these methods can however be interpreted as variants of *iterative* Schur complement methods as they could be found in References [26, 29], where step 2 of the Schur algorithm is solved iteratively instead of with direct methods as in the current approach.

The present results seem to confirm that none of the currently available parallel preconditioned iterative solvers (not even belonging to the so-called two level class [28, 33, 34, 43]) will ever outperform the currently proposed *direct* Schur complement algorithm at least for the class of problems we are concerned with.

ACKNOWLEDGEMENTS

The authors are indebted with the *Edinburgh parallel computing centre* (*EPCC*), who made generously available its Sun HPC 3500 super-computer, to Dr Simons during his TRACS research visit as PhD student, a training program on high-performance computing funded by the European Union's transnational access to research infrastructures programme. Professor Giulio Iannello of the *Universitá Campus Biomedico* of Rome provided valuable support when discussing several issues related to the parallel architecture.

REFERENCES

1. Piomelli U. DNS and LES, basic principles, current trends, expectations. *Introduction to the Modelling of Turbulence*, von Kármán Institute Lecture Series, 2000-02, 2002.

2. Sagaut P. *Large Eddy Simulation for Incompressible Flows*. Springer Scientific Computation, 2001.
3. Malan P. Turbulence modelling for industrial computational fluid dynamics. *Industrial Computational Fluid Dynamics*, von Kármán Institute Lecture Series, 1999-06, 1999.
4. Rodi W, Ferziger JH, Breuer M, Pourquié M. Status of large eddy simulation: results of a workshop. *Journal of Fluids Engineering* 1997; **119**:248–262.
5. Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J. Combined immersed boundary finite difference methods for three dimensional complex flow simulations. *Journal of Computational Physics* 2000; **161**(1):35–60.
6. Grosges T. Simulation on general bodies on Cartesian grids: application to VKI-LES code. *Manual 57*, von Kármán Institute for Fluid Dynamics, May 2002.
7. Patera AT. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *Journal of Computational Physics* 1984; **54:**468–488.
8. Henderson R, Karniadakis G. Unstructured spectral element methods for simulation of turbulent flows. *Journal of Computational Physics* 1995; **122**:191–217.
9. Simons E, Giammanco R, Benocci C, Manna M. Large eddy simulation of turbulent flows via domain decomposition techniques. Part 2: applications. *International Journal for Numerical Methods in Fluids*, 2005, in press.
10. Hartel C, Kleiser L, Unger F, Friedrich R. Subgrid-scale energy transfer in the near-wall region of turbulent flows. *Physics of Fluids* 1994; **6**(9):3130–3143.
11. Sarghini F, Piomelli U, Balaras E. Scale-similar models for large-eddy simulations. *Physics of Fluids* 1999; **11**(6):1484–1490.
12. Kerr RM, Tucker PG. A new mixed non linear LES model for boundary layers. *21st International Congress of Theoretical and Applied Mechanics*, *Paper FM24*, 2004.
13. Piomelli U. High Reynolds number calculations using the dynamic subgrid scale stress model. *Physics of Fluids* 1993; **5**(6):1484–1490.
14. Simons E. An efficient multi-domain approach to large eddy simulation of incompressible turbulent flows in complex geometries. *Ph.D. Thesis*, Catholic University of Louvain, Belgium, 2000.
15. Akselvoll K, Moin P. Large eddy simulation of turbulent confined coannular jets and turbulent flow over a backward facing step. *Report TF-63*, Stanford University, February 1995.
16. Van Kan J. A second order accurate pressure correction scheme for viscous incompressible flow. *Journal on Scientific and Statistical Computing* 1986; **7**:870–891.
17. Choi H, Moin P. Effects of the computational time step on the numerical solutions of turbulent flows. *Journal of Computational Physics* 1994; **113**(1):1–4.
18. Kirkpatrick MP, Armfield SW, Kent JH, Dixon T. Simulation of vortex shedding flows using high order fractional step methods. *ANZIAM Journal* 2000; **42**:C856–C876.
19. Simons E, Manna M, Benocci C. *Parallel Multi-Domain Large Eddy Simulation of the Flow over a Backward Facing Step at Re* = 5100, vol. 65, Chapter 5. Kluwer Academic Press: New York, 2002.
20. Hirsch C. *Numerical Computation of Internal and External Flow*, vols. 1, 2, Series in Numerical Methods in Engineering. Wiley-Interscience: New York, 1989.
21. Horiuti K. Comparison of conservative and rotational forms in large-eddy simulation of turbulent channel flow. *Journal of Computational Physics* 1987; **71**:343–370.
22. Tafti D. Comparison of some upwind-biased high order formulations with a second order central difference scheme for time integration of the incompressible Navier–Stokes equations. *Computers and Fluids* 1996; **25**(7):647–665.
23. Tucker PG. Hybrid MILES-RANS method for more dissipative solvers and the use of non-linear LES. *AIAA Paper 2004-0071*, 2004.
24. Schmidt H, Schumann U, Volkert H, Ulrich W. Three-dimensional, direct and vectorized elliptic solvers for various boundary conditions. *DFVLR-Mitt. 84-15*, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, July 1984.
25. Swarztrauber PN. A direct method for the discrete solution of separable elliptic equations. *SIAM Journal on Numerical Analysis* 1974; **11**(6):1136–1150.
26. Smith B, Bjorstad P, Gropp W. *Domain Decomposition*, *Parallel Multi-Level Methods for Elliptic Partial Differential Equations*. Cambridge University Press: Cambridge, 1996.
27. Quarteroni A, Valli A. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press: Oxford, 2000.
28. Chan TF, Mathew TP. Domain decomposition algorithms. *Acta Numerica* 1994; 61–143.
29. Goovaerts D. Domain decomposition methods for elliptic partial differential equations. *Ph.D. Thesis*, Catholic University of Louvain, Belgium, 1990.
30. Simons E, Manna M. A multi-domain approach to large simulation of complex turbulent flows. In *Numerical Methods for Fluid Dynamics VI*, Baines MJ (ed.), 1998; 499–506.
31. Schmitt L, Friedrich R. Large-eddy simulation of turbulent backward facing step flow. In *Notes on Numerical Fluid Mechanics*, Deville M (ed.), vol. 20, Vieweg, 1988; 355–362.

32. Le H, Moin P. An improvement of fractional step methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1991; **92**:369–379.
33. Bramble JH, Pasciak JE, Schatz AH. The construction of preconditioners for elliptic problems by substructuring I. *Mathematical Computations* 1986; **47**(175):103–134.
34. Carvalho LM, Giraud L, Le Tallec P. Algebraic two-level preconditioners for the Schur complement method. *SIAM Journal on Scientific Computing* 2001; **22**(6):1987–2005.
35. Hutchinson SA, Prevost LV, Shadid JN, Tuminaro RS. Aztec user's guide, Version 2.0 Beta. *Report SAND95-1559*, Massively Parallel Computing Research Laboratory, Sandia National Laboratories, July 1998.
36. Le H, Moin P. Direct numerical simulation of turbulent flow over a backward-facing step. *Report TF-58*, Stanford University, December 1994.
37. Swarztrauber PN, Sweet RA. Efficient FORTRAN subprograms for the solution of separable elliptic partial differential equations. *ACM Transactions on Mathematical Software* 1979; **5**:352–364.
38. Wilhelmson RB, Ericksen JH. Direct solutions of Poisson equation in three dimensions. *Journal of Computational Physics* 1977; **25**:319–331.
39. Message Passing Interface Forum. MPI: a message passing interface standard. *The International Journal of Supercomputer Applications and High Performance Computing* 1994; 8.
40. Gropp W, Lusk E, Skjellum S. *Using MPI*, *Portable Parallel Programming with the Message Passing Interface*. The MIT Press: Cambridge, MA, 1997.
41. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company: Massachusetts, 1996.
42. Saad Y. A dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications* 1994; **1**:387–402.
43. Smith BF. An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM Journal on Scientific and Statistical Computing* 1992; **13**(1):364–378.

