ELSEVIER

Comput. Methods Appl. Mech. Engrg. 174 (1999) 371–391

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Multi-domain parallel computation of wake flows

Y. Osawa, V. Kalro, T. Tezduyar*

Aerospace Engineering and Mechanics, Army High Performance Computing Research Center, University of Minnesota, 1100 Washington Avenue South, Minneapolis, MN 55415, USA

Received 17 April 1998; revised 21 May 1998

Abstract

We present a new, multi-domain parallel computational method for simulation of unsteady flows involving a primary object, a long wake region and, possibly, a secondary object affected by the wake flow. The method is based on the stabilized finite element formulation of the time-dependent Navier–Stokes equations of incompressible flows. In the multi-domain computational method the entire simulation domain is divided into an ordered sequence of overlapping subdomains. The flow data computed over the leading subdomain is used for specifying the inflow boundary conditions for the next subdomain. The subdomain corresponding to the wake would not involve any objects, hence the mesh constructed over this domain would be structured. A special-purpose finite element implementation for structured meshes is used for the wake domain to achieve much higher computational speeds compared to a general-purpose implementation. We present verification studies for the multi-domain method and special-purpose implementation, followed by two numerical examples. The first example is the wake behavior behind a circular cylinder. The second one is the aerodynamic effect of tip vortices released from a leading wing on a trailing wing placed in the far wake.

1. Introduction

Recent advances in methods for flow simulation and modeling, together with advances in supercomputing technology, have enabled us to develop powerful computational tools to simulate complex fluid mechanics applications. Methods designed for a general class of challenging flow problems sometimes need to be re-designed and/or optimized for more specific classes of problems which pose their own unique challenges. One of these specific classes is 3D simulation of unsteady wake flow generated by a primary (leading) object and its effect on a secondary (trailing) object placed in its wake. Examples of this class of applications are flow around a small aircraft in the wake of a larger aircraft and flow around a parachute traversing the wake flow of an aircraft. In many cases, because the two objects are separated by a large distance compared to the length scales of the objects, these wake regions are rather long and we need to take this situation into account in developing computational methods which will be effective for this particular class of problems.

In this paper, we present a multi-domain parallel computational method for simulation of unsteady flow past the primary object, unsteady wake flow in the long region between the two objects and the influence of this wake flow on the secondary object. The base method is a finite element formulation with the streamline-upwind/Petrov–Galerkin (SUPG) [1] and pressure-stabilizing/Petrov–Galerkin (PSPG) [2] stabilizations. With these stabilization techniques, simulations can be carried out for flows with high Reynolds numbers and thin boundary layers, without generating numerical oscillations but also without introducing excessive numerical dissipation to the computations. Furthermore, these stabilization techniques allow us to use equal-order interpolation functions (such as trilinear-trilinear and linear-linear) for velocity and pressure without generating any numerical oscillations that would normally be caused by such combinations of interpolation functions.

* Corresponding author. Mechanical Engineering and Materials Science, Rice University – MS 321, 6100 Main Street, Houston, TX 77005-1892.

0045-7825/99/$ – see front matter
PII: S0045-7825(98)00305-3

The spatial discretization of the finite element formulation leads to a coupled, nonlinear, ordinary differential equation system which is solved with a generalized trapezoidal time-marching scheme. At each time step, a coupled, nonlinear equation system is solved with Newton–Raphson iterations. At each Newton–Raphson step, a coupled, linear equation system is solved iteratively with the GMRES [3] update technique. In computation of vector–matrix products involved in these innermost iterations, we use element-matrix-based, element-vector-based (also called matrix-free) [4] and sparse-matrix-based [5] methods. These methods, particularly the last two, can be very effectively used for simulation of problems with millions of equations. The entire formulation and the solution techniques have been implemented on parallel computing platforms by using the MPI programming environment. The parallel platforms used include CRAY T3D, CRAY T3E and the SGI POWER CHALLENGE with the R8000 processor.

The multi-domain computational approach is based on dividing the entire simulation domain into an ordered sequence of overlapping subdomains. In cases involving objects in tandem, Subdomain-1 (SD-1) and Subdomain-3 (SD-3) would be used for computation of the unsteady flow around the primary and secondary objects, respectively. Because these objects typically would have complex geometries, the subdomains are discretized with unstructured meshes and a general-purpose implementation (GPI) of the finite element flow solver. Subdomain-2 (SD-2), which connects SD-1 and SD-3, would be used for computation of the unsteady wake flow generated by the primary object. It is discretized by highly-refined structured meshes to capture the wake flow behavior accurately. A special-purpose implementation (SPI) for structured meshes can be optimized to yield much higher computational speeds compared to GPI. Thus SD-2 serves as a hi-fidelity, low-cost channel for transferring flow information from SD-1 to SD-3.

To demonstrate this multi-domain computational approach, we present two numerical examples. The first example is the unsteady wake flow behind a circular cylinder. Several researchers have reported [6–10] 3D behavior behind a circular cylinder. 3D simulations in the near wake region of a circular cylinder were presented in a paper by Kalro and Tezduyar [11]. We use a highly-refined mesh to capture in detail the 3D effects and vortex shedding in the far wake. The second example is flow past a large, leading wing and two small, trailing wings placed in the far wake of the larger wing. This simulation allows us to study the aerodynamic response of secondary objects affected by the wake flows. It is a step towards developing tools which address the dynamics of parachute systems which often encounter the wake of the leading aircraft.

The governing equations for incompressible flows are reviewed in Section 2, followed by a brief description of the finite element formulation in Section 3. The multi-domain computational method and a parallel data projection scheme are discussed in Section 4. A special-purpose implementation of the finite element formulation for structured meshes in wake flow computations is described in Section 5. The verification studies for the multi-domain method are presented in Section 6, and the numerical examples are presented in Section 7. Concluding remarks are given in Section 8.

2. Governing equations

For a fixed 3D spatial domain Ω with boundary Γ, the Navier–Stokes equations of incompressible flows are written as follows:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \boldsymbol{f}\right) - \nabla\cdot\boldsymbol{\sigma} = 0 \quad \text{on } \Omega\,, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{on } \Omega\,. \tag{2}$$

Here, $\boldsymbol{u}$ is the velocity vector, ρ is (constant) density and $\boldsymbol{f}$ is the external force. The stress tensor $\boldsymbol{\sigma}$ is defined as

$$\boldsymbol{\sigma}(\boldsymbol{u}, p) = -p\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})\,, \tag{3}$$

where p is the pressure, $\boldsymbol{I}$ is the identity tensor, μ is the constant viscosity and $\boldsymbol{\varepsilon}(\boldsymbol{u})$ is the strain rate tensor:

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left((\nabla\boldsymbol{u}) + (\nabla\boldsymbol{u})^{\mathrm{T}}\right). \tag{4}$$

Both Dirichlet- and Neumann-type boundary conditions are considered and are expressed as

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{g} \quad \text{on } \Gamma_g, \\ &\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{h} \quad \text{on } \Gamma_h, \end{aligned} \tag{5}$$

where Γ_g and Γ_h are complementary subsets of the boundary Γ, $\boldsymbol{n}$ is the unit normal vector at the boundary and $\boldsymbol{g}$ and $\boldsymbol{h}$ are given functions. A divergence-free velocity field is specified as the initial condition.

3. Stabilized formulation for incompressible flow simulations

The domain Ω is discretized into sub-domains Ω_e, $e = 1, 2, \ldots, n_{\text{el}}$, where n_{el} is the number of elements. For this discretization, we define the finite element interpolation function spaces $\mathcal{S}_u^h$ for velocity and $\mathcal{S}_p^h$ for pressure and the corresponding test function spaces $\mathcal{V}_u^h$ and $\mathcal{V}_p^h$:

$$\mathcal{S}_u^h = \{u^h | u^h \in [H^{1h}(\Omega)]^{n_{\text{sd}}}, u^h = g^h \text{ on } \Gamma_g\}, \tag{6}$$

$$\mathcal{V}_u^h = \{w^h | w^h \in [H^{1h}(\Omega)]^{n_{\text{sd}}}, w^h = 0 \text{ on } \Gamma_g\}, \tag{7}$$

$$\mathcal{S}_p^h = \mathcal{V}_p^h = \{q^h | q^h \in H^{1h}(\Omega)\}, \tag{8}$$

where $H^{1h}(\Omega)$ is the finite-dimensional function space over Ω. The stabilized finite element formulation is written as follows: find $\boldsymbol{u}^h \in \mathcal{S}_{\boldsymbol{u}}^h$ and $p^h \in \mathcal{S}_p^h$ such that $\forall\, \boldsymbol{w}^h \in \mathcal{V}_{\boldsymbol{u}}^h$ and $q^h \in \mathcal{V}_p^h$:

$$\begin{aligned} &\int_\Omega \boldsymbol{w}^h \cdot \rho\left(\frac{\partial \boldsymbol{u}^h}{\partial t} + \boldsymbol{u}^h \cdot \nabla \boldsymbol{u}^h - \boldsymbol{f}\right) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{w}^h) : \boldsymbol{\sigma}(p^h, \boldsymbol{u}^h)\, \mathrm{d}Q + \int_\Omega q^h \nabla \cdot \boldsymbol{u}^h \, \mathrm{d}Q \\ &\quad + \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\text{SUPG}} \rho u^h \cdot \nabla w^h + \tau_{\text{PSPG}} \nabla q^h] \cdot [\boldsymbol{L}(\boldsymbol{u}, p) - \rho \boldsymbol{f}] \, \mathrm{d}\Omega^e + \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} \delta \nabla \cdot \boldsymbol{w}^h \rho \nabla \cdot \boldsymbol{u}^h \, \mathrm{d}\Omega^e \\ &= \int_{\Gamma_h} \boldsymbol{w}^h \cdot \boldsymbol{h}^h \, \mathrm{d}\Gamma. \end{aligned} \tag{9}$$

The following notation is used in the finite element formulation given by Eq. (9):

$$\boldsymbol{L}(\boldsymbol{w}, q) = \left[\rho\left(\frac{\partial \boldsymbol{w}\mathrm{h}}{\partial t} + \boldsymbol{u}^h \cdot \nabla \boldsymbol{w}^h\right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \boldsymbol{w}^h)\right]. \tag{10}$$

For further details on the formulation the reader is referred to Tezduyar [2].

4. The multi-domain computational method

First, the entire simulation domain is divided into an ordered sequence of overlapping subdomains. The number of subdomains depends on the problem and the power of the computational resources used.

SD-1 is used for computation of the unsteady flow around the primary object. The inflow conditions for SD-1 are obtained from free-stream conditions. SD-1 typically contains a complex geometry and is therefore discretized using unstructured meshes. A GPI is used for computations in this domain. An appropriately positioned layer is selected in SD-1, downstream of the primary object. We shall refer to this layer as Outflow Layer-1 (OL-1). The flow data in OL-1 is extracted at every time step.

SD-2 is used for computation of the unsteady wake flow generated by the primary object and this subdomain would typically be much longer than SD-1. Since SD-2 contains only the wake and no other objects, it is discretized by highly-refined structured meshes, perhaps even a uniform one. The computation over SD-2 can be accomplished by methods other than the finite element method, such as the spectral method and that might be more desirable for computations with very regular geometries. The time-dependent inflow boundary conditions for SD-2 are obtained by projection between OL-1 and the inflow layer of SD-2. We call this inflow layer IL-2 which overlaps with OL-1. As is the case for SD-1, a suitably located outflow layer OL-2 is selected for SD-2.

Similar to SD-1, the computation over SD-3, which would contain the secondary object, would typically require an unstructured mesh and consequently a GPI. Inflow conditions for SD-3 at IL-3 are obtained in a similar fashion as it is done for IL-2. These different subdomain computations can be performed on different computational platforms and almost in parallel, provided a leading subdomain is at least one time step ahead of the following subdomain.

The key points of this method are described below.

4.1. Location of the interfaces

We need to place IL-2 sufficiently ahead of the outflow boundary of SD-1, so that the input to SD-2 is sufficiently clear of any truncation effects at the downstream boundary of SD-1. Furthermore, this input needs to be captured early enough before the solution in SD-1 enters the coarser regions of the mesh towards the downstream boundary. Similarly, IL-3 needs to be placed sufficiently ahead of the outflow boundary of SD-2, which in turn needs to be sufficiently distant from the secondary object.

4.2. Variables specified at the inflow layer

For SD-2, SD-3, ..., normally we would specify the velocity at the inflow boundary plane. Alternately, we can specify the velocity and pressure at the inflow boundary plane and the velocity at the second plane of the inflow layer. If these inflow layer conditions are extracted from a single-domain computation where the mesh downstream of this inflow layer is identical to the mesh used for SD-2, SD-3, ..., then the solution obtained with the SD-2, SD-3, ..., will be consistent with the solution obtained with the single-domain computation. This is the reason why we sometimes use this two-plane approach rather than just specifying the velocity at the inflow boundary plane. We will show some comparisons between the two approaches in Section 6.

4.3. Data projection between layers

Because in general OL-n and IL-(n + 1) might have different meshes, we have developed a parallel version of the data projection method presented in [12] to transfer flow information between these outflow and inflow layers.

The data projection procedure consists of an interpolation step and a least-squares step. The flow data needs to be interpolated from the elements in the outflow layer to the quadrature points (QPs) of the elements in the inflow layer. Hence, we are required to search the outflow layer for elements which contain the QPs of elements in the inflow layer. This search need to be conducted only once for a given set of meshes involved. Therefore it does not contribute significantly to the computational cost. However, to be prepared for more general cases, such as meshes that deform or move relative to each other, we developed an efficient parallel implementation of this search process. The amount of communication could become large on a distributed memory computing platform since all elements could possibly be searched for each QP. We developed a two-level search method to obtain better performance to reduce this communication. An enclosing volume (EV) is constructed on each processor such that it encloses all locally-residing elements in the outflow layer (see Fig. 1). Each processor sends the information of the EVs boundary to all other processors. In Fig. 1, two diagonally located nodal coordinates (●

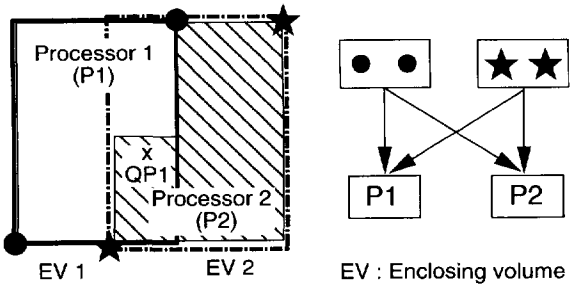

Fig. 1. Example of the EVs of the outflow layer and a QP of the inflow layer for two processors.

for EV1 and ★ for EV2) are sent to all other processors. The first level search is then locally carried out over all EVs. Subsequently, nodal coordinates of the QPs are sent to the processors which hold the EV found in the search. Some nodal coordinates (e.g. QP1 in Fig. 1) would be sent to more than one processor since EVs may overlap. Once the processor(s) expected to contain a QP is identified, a second-level search for the QP is locally carried out over elements of the outflow layer residing on that(those) processor(s). For the second-level search, we use the 1D search algorithm described in [12]. After this 1D search, in the processor(s) which were expected but not confirmed to contain a QP, we perform an all-elements search for that QP. At this point, an all-elements search becomes more efficient, because in confirmation that a processor does not contain a QP, the 1D search algorithm with random restarts would end up having as many restarts as the number of outflow layer elements in that processor. The all-elements search requires $O(n)$ checks, but the 1D search with random restart requires $O(n \log n)$ checks. Here, n is the number of elements in the search domain. After finding the elements which contain the QPs, the flow data is interpolated to each QP in the inflow layer from the elements in the outflow layer.

Once the flow data is interpolated to each of the QPs, a least-squares step is carried out on each processor:

$$\int_{\Omega_{\text{interface}}} \boldsymbol{w}^{\text{IL}} \cdot (\boldsymbol{d}^{\text{IL}} - \boldsymbol{d}^{\text{OL}})\, \mathrm{d}\Omega = 0\,. \tag{11}$$

The solution to Eq. (11) is obtained using Jacobi iterations because of its low memory requirements.

5. A special-purpose implementation of the finite element formulation for wake computations

A general-purpose implementation (GPI) of the finite element formulation employs quadrature rules for numerical integration. When the meshes are structured (regular), computational performance can be improved by a special-purpose implementation (SPI) of the finite element formulation described below.

Some researchers [13–15] have proposed a one-point integration with hourglass control. Some others [16,17] have developed an analytical integration scheme. Between the two, the former is more general but requires more computational efforts; the latter requires less computation but is difficult to apply to arbitrary element shapes. From these characteristics, the analytical integration is more suitable for our SPI goal since the meshes we plan to use are structured.

We describe here the approximations to the terms containing the advection velocity. The terms affected by this approximation are the nonlinear advective term $\boldsymbol{w} \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})$ and the SUPG term $(\boldsymbol{u} \cdot \nabla \boldsymbol{w}) \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})$. For example, the advective term is expressed in the weak form as follows:

$$\int_{\Omega_e} \boldsymbol{w} \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})\, \mathrm{d}\Omega_e = \int_{\Omega_e} \left(\sum_{a=1}^{nen} N_a c_a \right) \cdot \left[\left(\sum_{b=1}^{nen} N_b \boldsymbol{u}_b \right) \cdot \nabla \left(\sum_{c=1}^{nen} N_c \boldsymbol{u}_c \right) \right] \mathrm{d}\Omega_e\,. \tag{12}$$

Integrations involving products of velocities create the difficulty of using analytical integration. To overcome this, the advective velocity is approximated by its value $\boldsymbol{u}_0$ at the element center:

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} \cong \boldsymbol{u}_0 \cdot \nabla \boldsymbol{u}\,, \tag{13}$$

$$\boldsymbol{u} \cdot \nabla \boldsymbol{w} \cong \boldsymbol{u}_0 \cdot \nabla \boldsymbol{w}\,. \tag{14}$$

Hence, for example, the advective term in Eq. (12) is expressed as

$$\int_{\Omega_e} \boldsymbol{w} \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})\, \mathrm{d}\Omega_e \cong \boldsymbol{u}_0 \int_{\Omega_e} \left(\sum_{a=1}^{nen} N_a c_a \right) \cdot \nabla \left(\sum_{c=1}^{nen} N_c \boldsymbol{u}_c \right) \mathrm{d}\Omega_e\,. \tag{15}$$

With this approximation, now we can easily do an analytical integration. When highly-refined meshes are used, very accurate solutions can be obtained (as indicated by our results). This approximation is also employed by Christon [15].

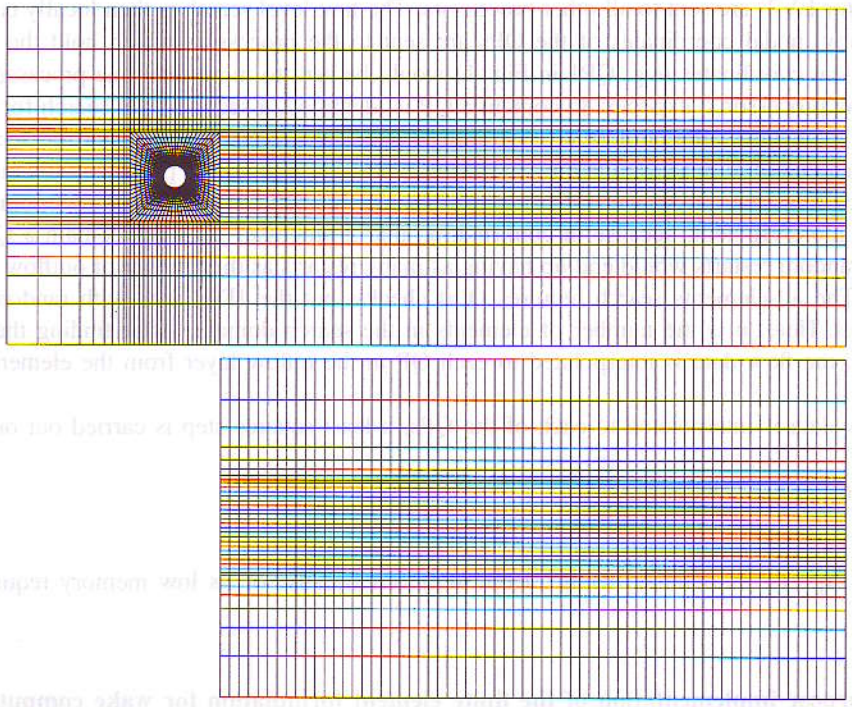

Fig. 2. Verification of the multi-domain method. Mesh for the single-domain computation (upper) and the multi-domain computation (lower).

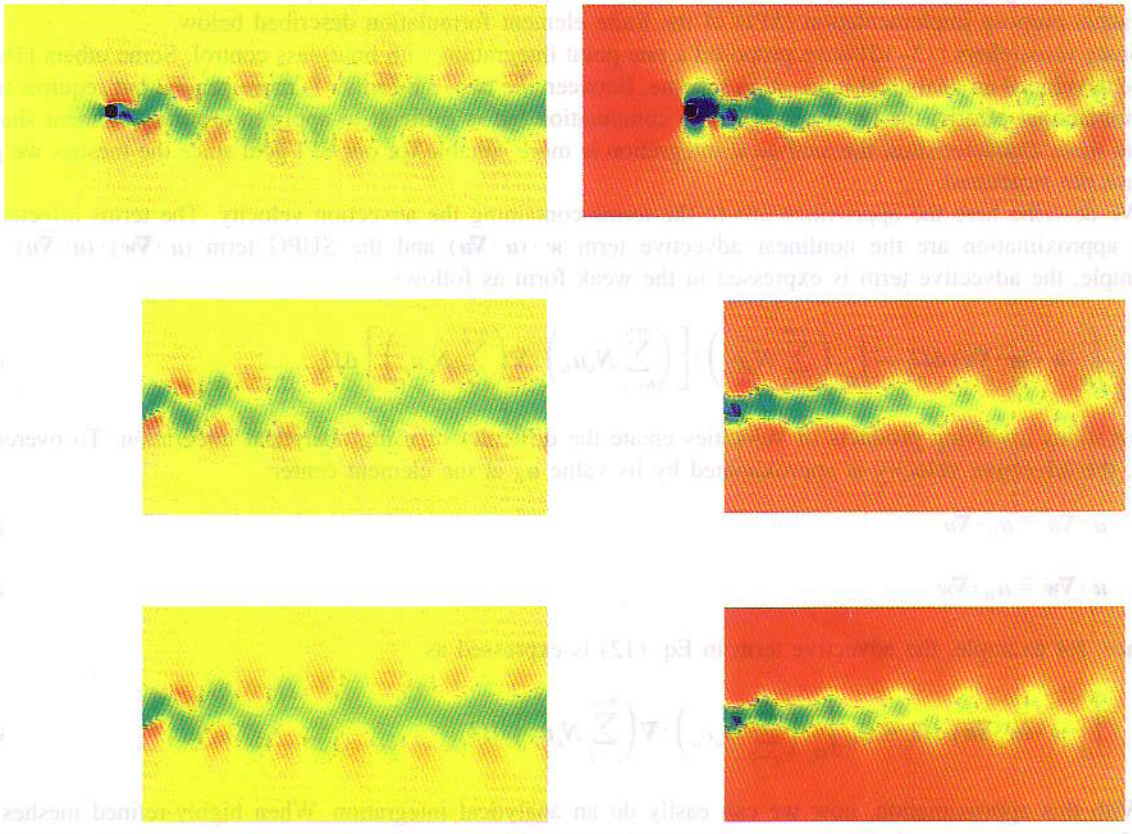

Fig. 3. Verification of the multi-domain method. Streamwise component of the velocity (left) and pressure (right); obtained with the single-domain computation (upper), multi-domain computation with two-plane inflow conditions (middle) and multi-domain computation with single-plane inflow conditions (lower).

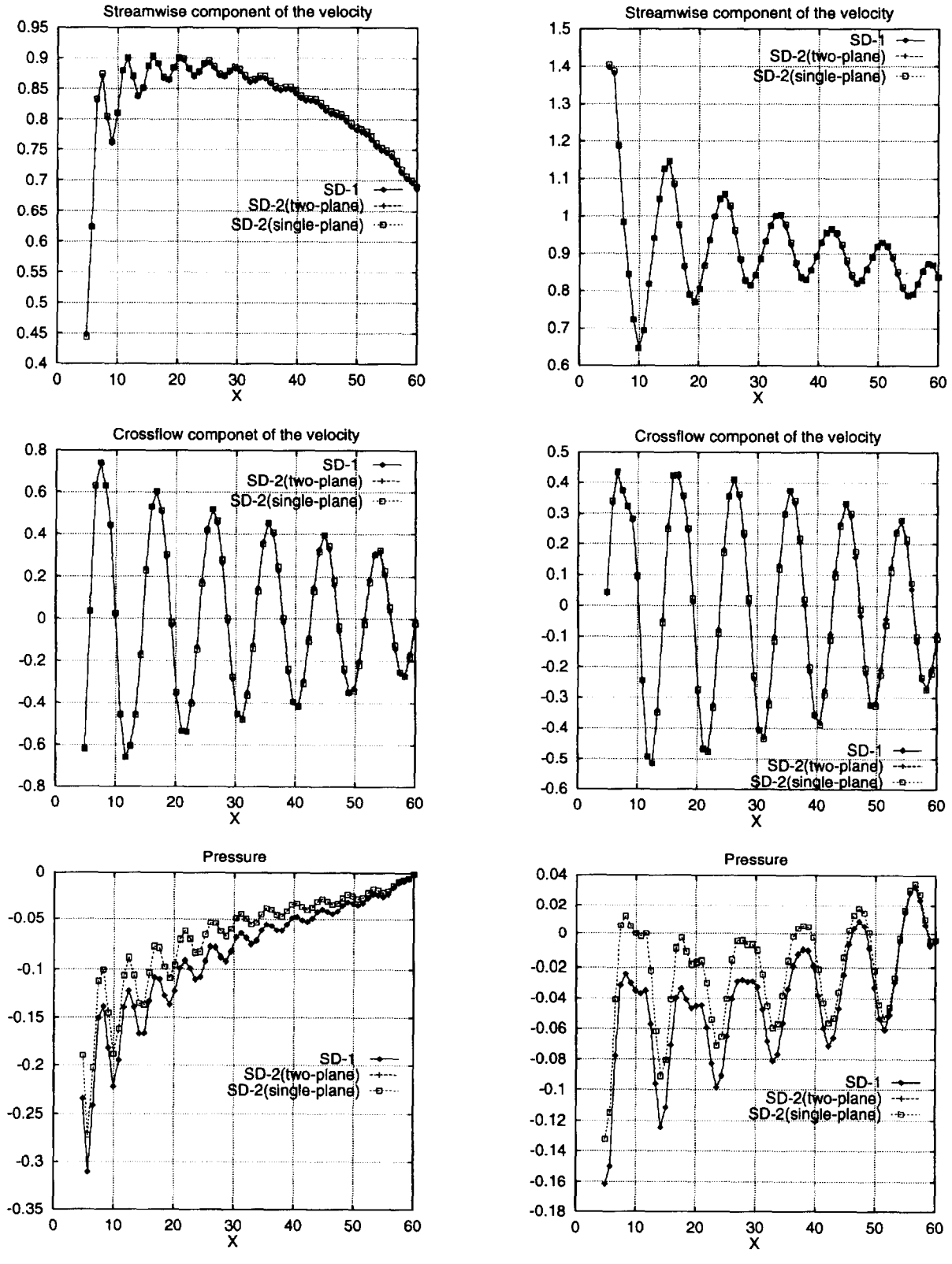

Fig. 4. Verification of the multi-domain method. Streamwise component of the velocity (upper), crossflow component of the velocity (middle) and pressure (lower); along the center line of the domain (left) and the line passing through the lower row of vortices (right). The results were obtained with the single-domain computation, multi-domain computation with two-plane inflow conditions and multi-domain computation with single-plane inflow conditions.

6. Verification studies

6.1. Verification of the multi-domain method

Simulation of flow past a circular cylinder at Re = 300 is used to verify our multi-domain method. SD-1 is selected as the mesh for a single-domain model. The cylinder, with both the radius and span length one unit, is located at the origin of the computational domain. The upstream, downstream and crossflow (i.e. top and bottom lateral) boundaries are located, respectively, at 15, 60 and 15 units from the origin (see Fig. 2). The mesh consists of 9656 nodes and 4656 hexahedral elements (with one element along the spanwise direction) and results in 28 116 equations. The boundary conditions consist of uniform inflow velocity, zero-shear stress and zero-normal velocity at lateral boundaries, traction-free condition at the outflow boundary and no-slip condition on the cylinder. For the multi-domain computation, we chose the nodes of the wake mesh (SD-2) to coincide with those of SD-1. For SD-2 we carried out two sets of computations; one with the inflow conditions specified at both planes of IL-2 and the other with the inflow conditions specified only at the inflow boundary plane. The interface (OL-1 and IL-2) is selected in such a way that its upstream plane is located 4 units downstream of the origin. SD-2 consists of 4958 nodes and 2376 hexahedral elements and results in 14 244 equations. The inflow

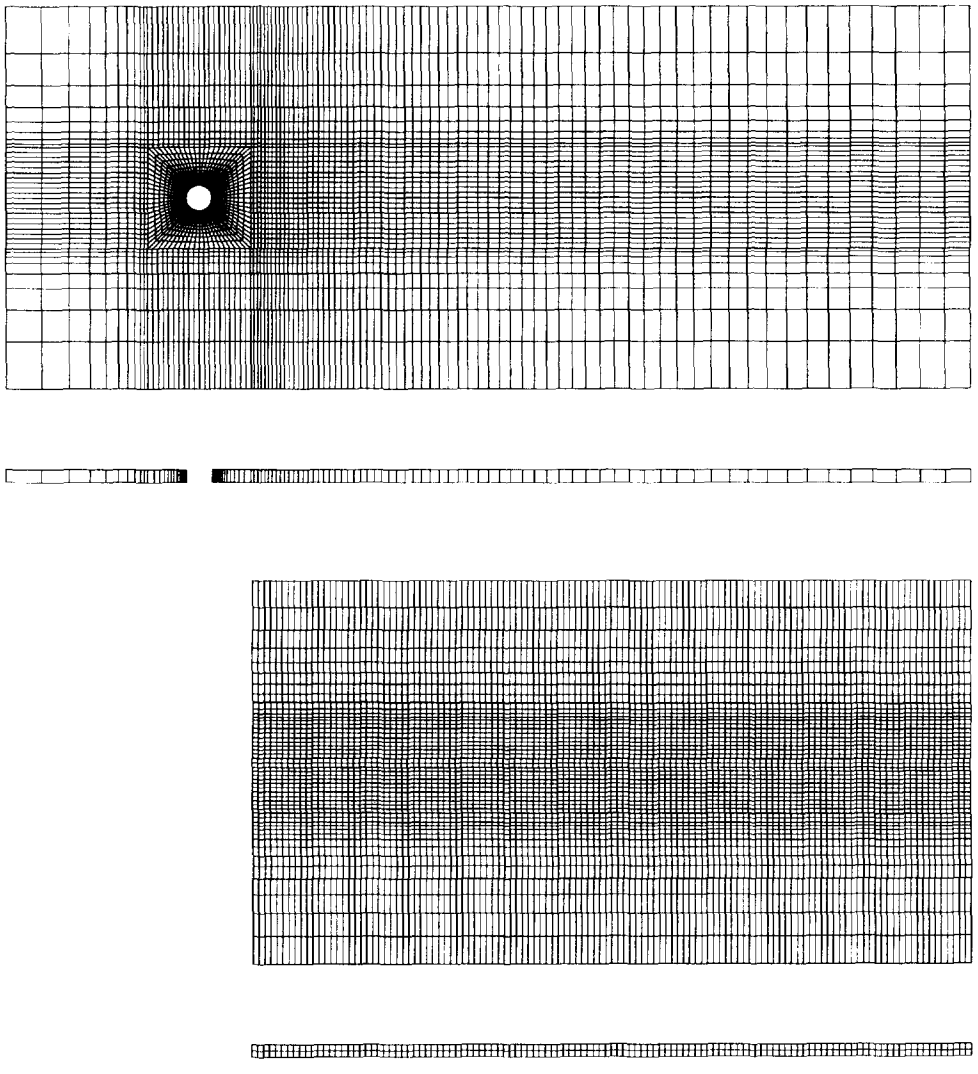

Fig. 5. Verification of SPI compared to GPI. Mesh for the first subdomain (upper) and the second subdomain (lower).

conditions are extracted from the solution on SD-1. Boundary conditions similar to those for SD-1 are imposed at the crossflow and downstream boundaries.

For these computations, the free-stream velocity is set to 1 and the time step size to 0.1. Two nonlinear iterations are employed at every time step and the size of the Krylov space within the GMRES update is 80, without restart. The computation over SD-2 requires 7.85 s for every time step on 16 processors of a CRAY T3E-900. The projection requires 0.13 s for every time step and this is 1.7% of the total time.

In Fig. 3, we compare, at the horizontal plane, the velocity and pressure obtained with the three computations. In Fig. 4, we compare the plots for the streamwise component of the velocity and pressure, along the center line of the domain and the line passing through the lower row of vortices. We note the exact agreement between the SD-1 solution and the SD-2 solution based on two-plane inflow conditions. For the SD-2 computation with single-plane inflow conditions, the velocities match closely with SD-1, however, for pressure, we see a small

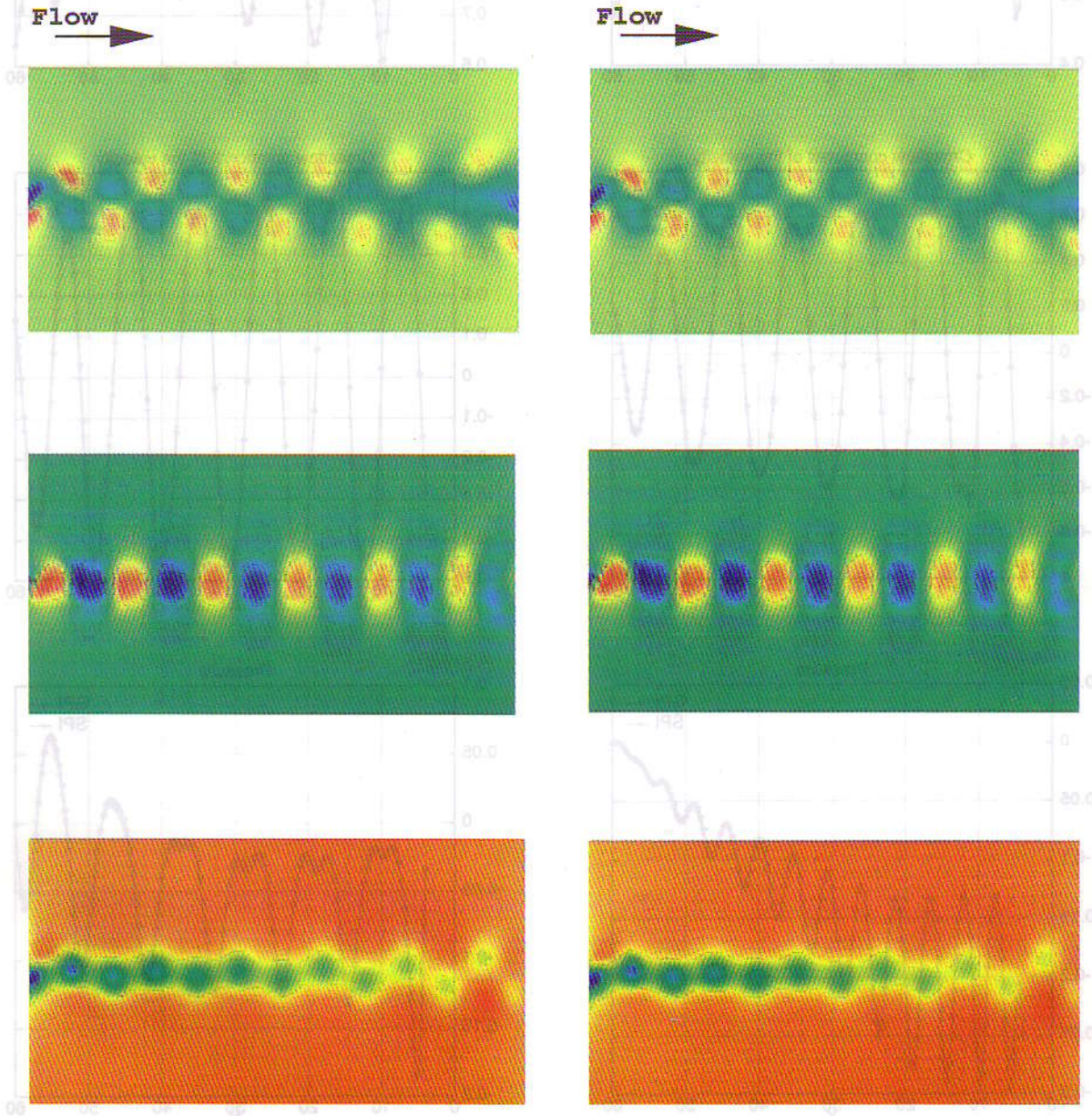

Fig. 6. Verification of SPI compared to GPI. Streamwise component of the velocity (upper), crossflow component of the velocity (middle) and pressure (lower); obtained with GPI (left) and SPI (right).

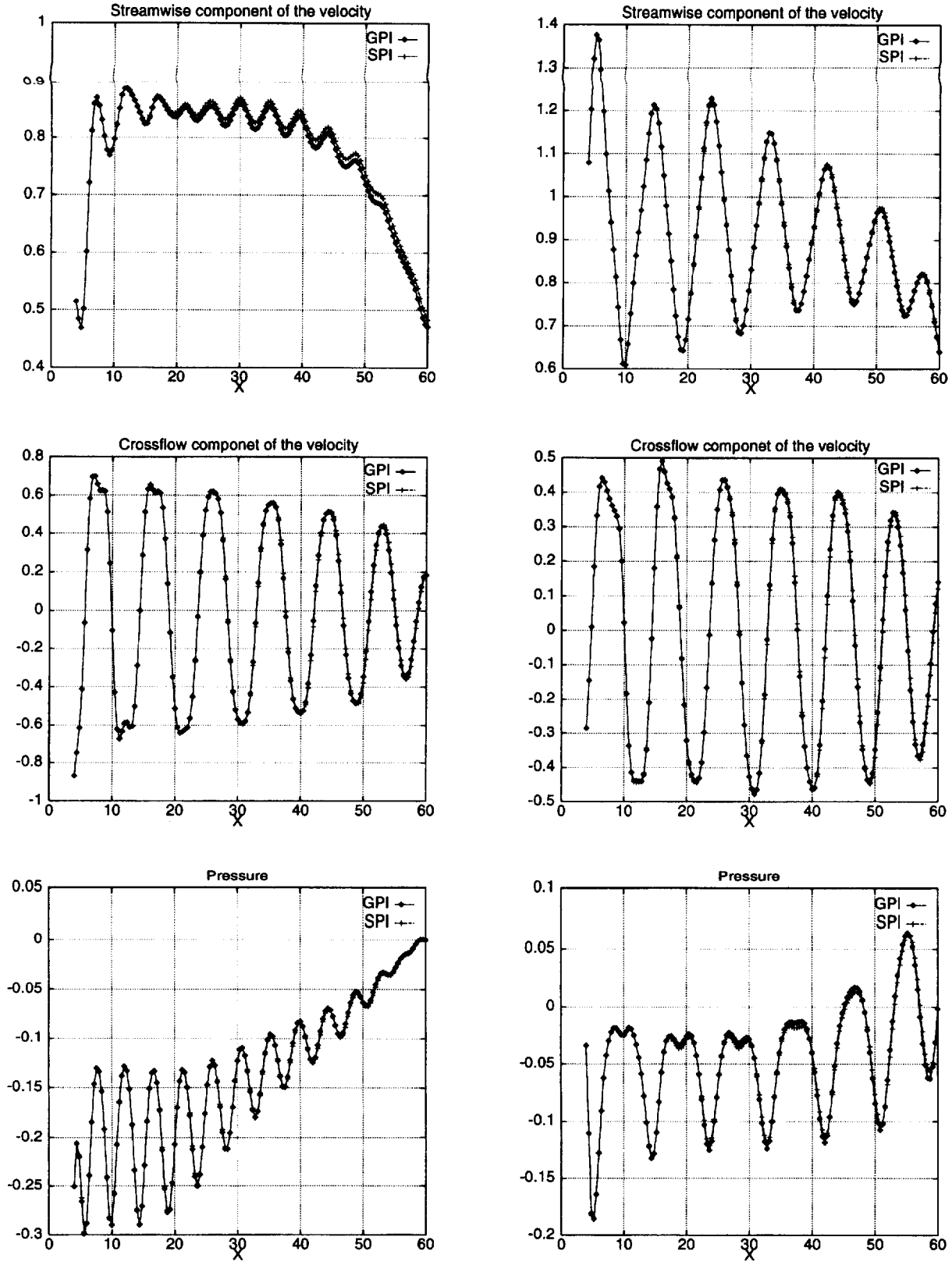

Fig. 7. Verification of SPI compared to GPI. Streamwise component of the velocity (upper), crossflow component of the velocity (middle) and pressure (lower); along the center line of the domain (left) and the line passing through the lower row of vortices (right); obtained with GPI and SPI.

shift which gradually diminishes away from the inflow boundary. All simulations described in the remaining part of this paper were based on the two-plane inflow conditions.

6.2. Verification of SPI compared to GPI

Flow past a circular cylinder at Re = 300 is utilized here to verify the accuracy of SPI and compare its performance to GPI. Fig. 5 shows the meshes for these computations. SD-1 has the same domain size, boundary conditions and number of nodes and elements as that of SD-1 in previous computations. The only difference is that the wake mesh is non-uniform. SD-2 has the same domain size and boundary conditions as that of SD-2 in previous computations. The difference is that the mesh is more refined in the horizontal plane and has 2

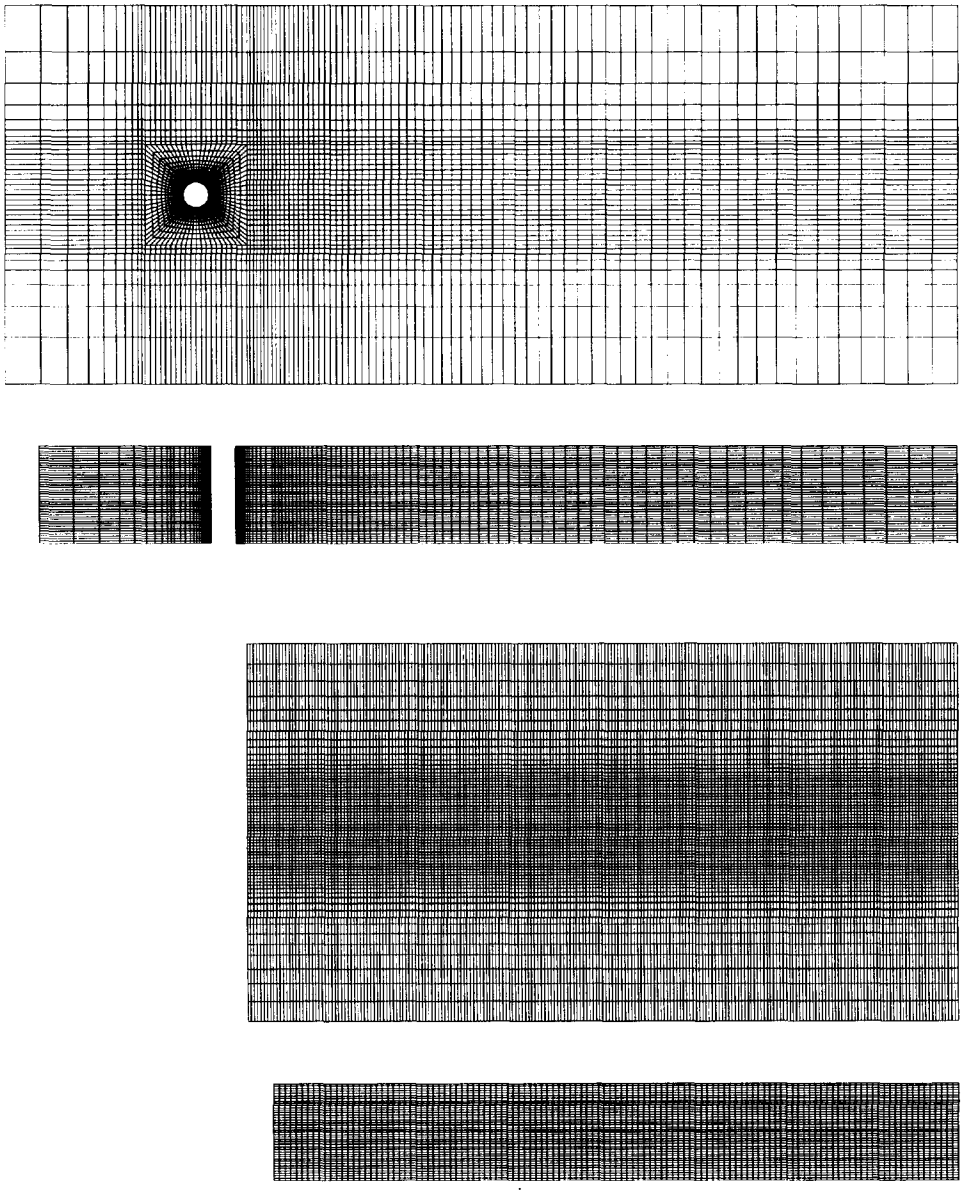

Fig. 8. Cylinder wake computation at Re = 300 with a highly-refined wake mesh. Mesh for the single-domain computation (upper) and multi-domain computation (lower).

elements in the spanwise direction. The interface (OL-1 and IL-2) is selected the same way as before. This mesh consists of 17 787 nodes and 11 520 hexahedral elements and results in 57 743 equations. For these computations, the size of the Krylov space is 10, without restart and there are two nonlinear iterations at each time step.

In Fig. 6, we compare, at the centered horizontal plane, the velocity and pressure obtained with GPI and SPI. In Fig. 7, we compare the plots for the streamwise and crossflow (i.e. direction perpendicular to streamwise and spanwise direction) components of the velocity and pressure, along the center line of the domain and the line passing through the lower row of vortices. The results from SPI match those from GPI, but with a slight difference (maximum 5%) in the streamwise component of the velocity along the centerline of the domain. On 8 processors of a CRAY T3E-900, per time step, GPI requires 4.36 s, while SPI requires only 1.44 s, resulting in a three-fold speedup.

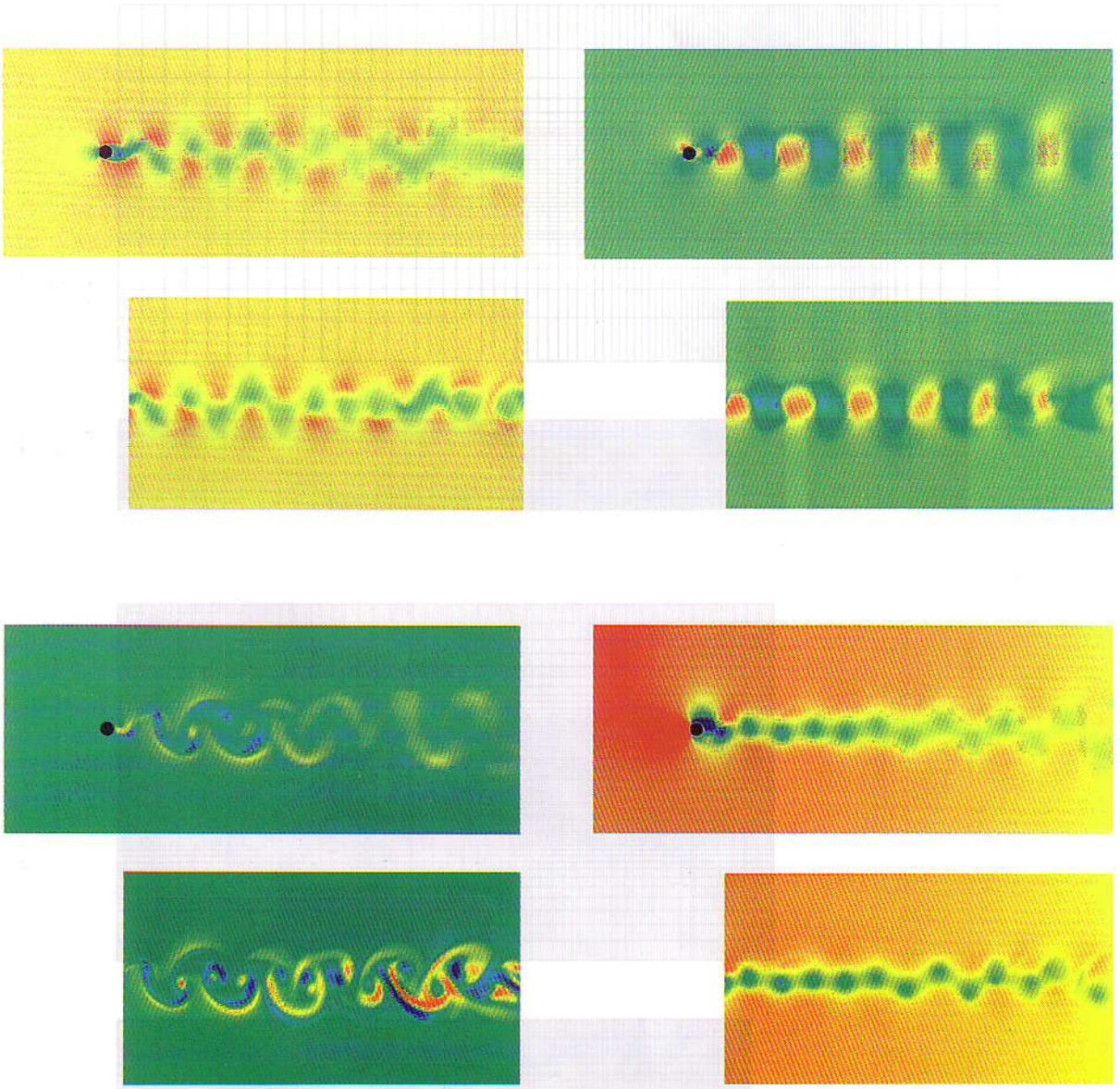

Fig. 9. Cylinder wake computation at Re = 300 with a highly-refined wake mesh. Comparison of the solutions obtained with the single-domain and multi-domain computations; streamwise component of the velocity (upper-left), crossflow component of the velocity (upper-right), spanwise component of the velocity (lower-left) and pressure (lower-right).

7. Numerical examples

7.1. Cylinder wake computation at Re = 300 with a highly-refined wake mesh

In this section, we compute the unsteady flow past a circular cylinder with the single- and multi-domain methods. The purpose is to capture the 3D wake behavior with a highly-refined mesh downstream of the cylinder.

We carried out two sets of computations. One is the single-domain computation using the same mesh as the one used by Kalro and Tezduyar [11] and the other one is the multi-domain computation with a highly-refined structured mesh (see Fig. 8). The in-plane mesh is the same as the one used in the previous section, with the span length of the cylinder set to 8 units. SD-1 consists of 197 948 nodes and 186 240 hexahedral elements and results in 760 107 equations.

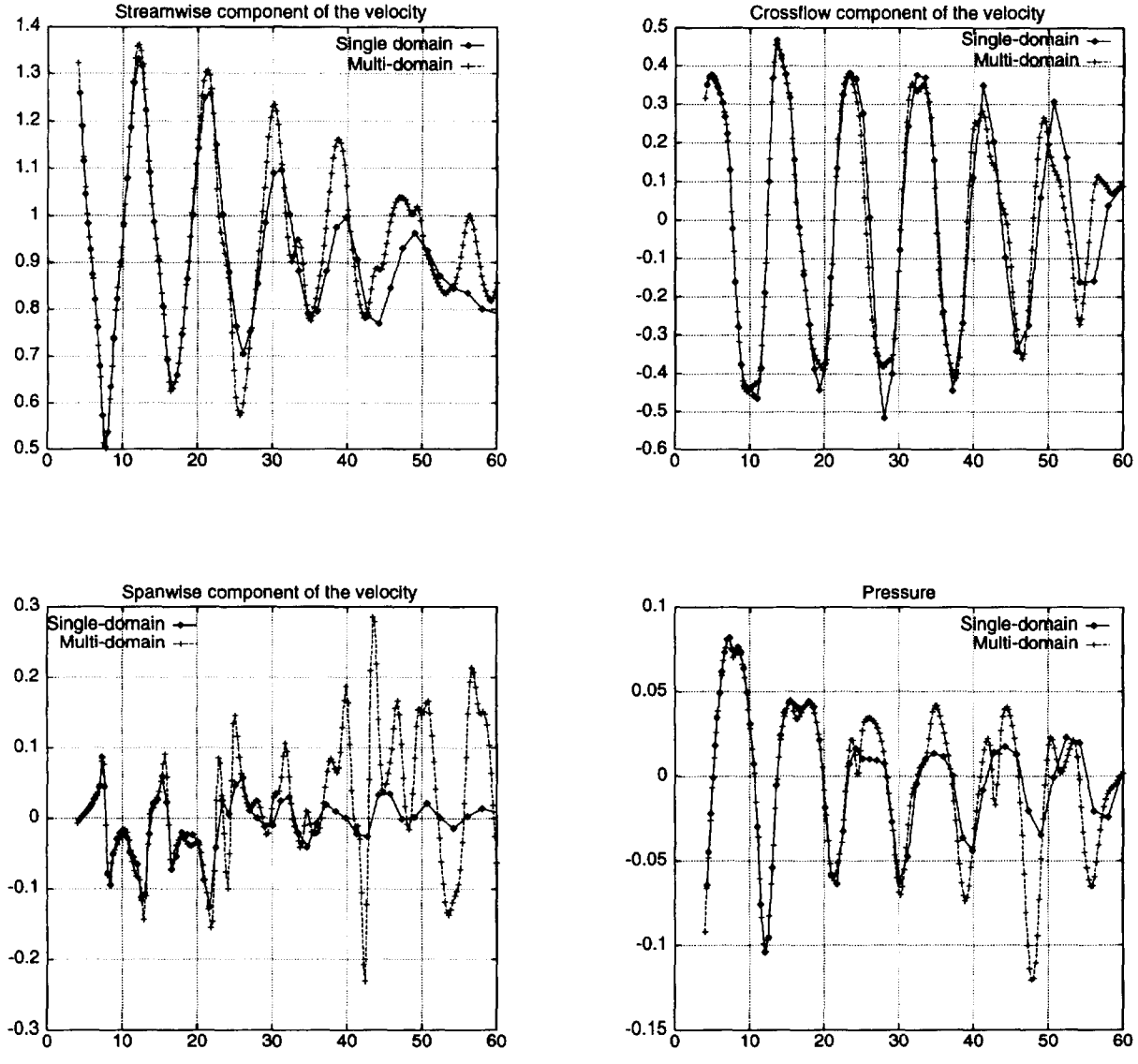

Fig. 10. Cylinder wake computation at Re = 300 with a highly-refined wake mesh. Comparison of the solutions obtained with the single-domain and multi-domain computations; streamwise component of the velocity (upper-left), crossflow component of the velocity (upper-right), spanwise component of the velocity (lower-left) and pressure (lower-right); along the line passing through the lower row of vortices.

The wake domain (SD-2) stretches 56, 30 and 8 units along the streamwise, crossflow and spanwise directions, respectively. IL-2 is located, as before, 4 units downstream of the origin. SD-2 mesh consists of 1 163 565 nodes and 1 126 400 hexahedral elements and results in 4 553 457 equations. Compared to the wake region of the single-domain model, the number of elements in SD-2 is around 3.3 times larger in the streamwise direction, around 1.8 times larger in the crossflow direction and 2.0 times larger in the spanwise direction. For sake of clarity, the bottom image in Fig. 8 shows a mesh twice coarsened in the spanwise direction.

For these computations, the free-stream velocity is set to 1 and the time step size to 0.1. The size of the Krylov space is 10, without restart; and the number of nonlinear iterations is two per time step. These computations were carried out on a CRAY T3E-900. The computation, per time step, requires 26 s on 32 processors for SD-1 and 26 s on 48 processors for SD-2.

Fig. 9 shows the velocity and pressure for both computations. These figures clearly show that the solution on SD-2 captures flow features in greater detail and maintains these features all the way to the outflow boundary. The spanwise component of the velocity grows downstream in SD-2, whereas it fades in SD-1 due to dissipation caused by coarse elements. The difference between these two computations becomes more clear in Fig. 10 which shows the plots for velocity components along the line passing through the lower row of vortices.

7.2. Flow past wings in tandem

In this section, we compute the unsteady flow past a large wing at Re = 1000 and the effect of its wingtip vortices on two smaller wings placed downstream (see Fig. 11). We assume symmetry with respect to the plane passing through the middle of the leading wing. SD-1 contains half of the primary wing and is handled by GPI (see Fig. 11). SD-2 is the wake region and is handled by SPI. SD-3 contains one of the trailing wings and is handled by GPI and an unstructured mesh[1].

The leading wing has a unit chord length with an aspect ratio of 8. Its cross-section is a NACA 0012 and is placed at 8.0° angle of attack (see Fig. 12). The center of the wing is located at the origin of the solution domain.

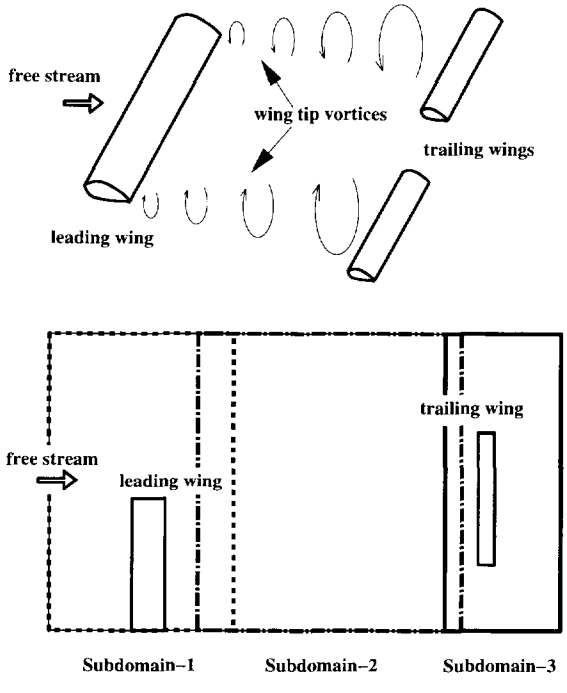

Fig. 11. Arrangement of the wings and the subdomains.

[1] The mesh generator was developed by the Team for Advanced Flow Simulation and Modeling (T*AFSM) at the Army HPC Research Center.

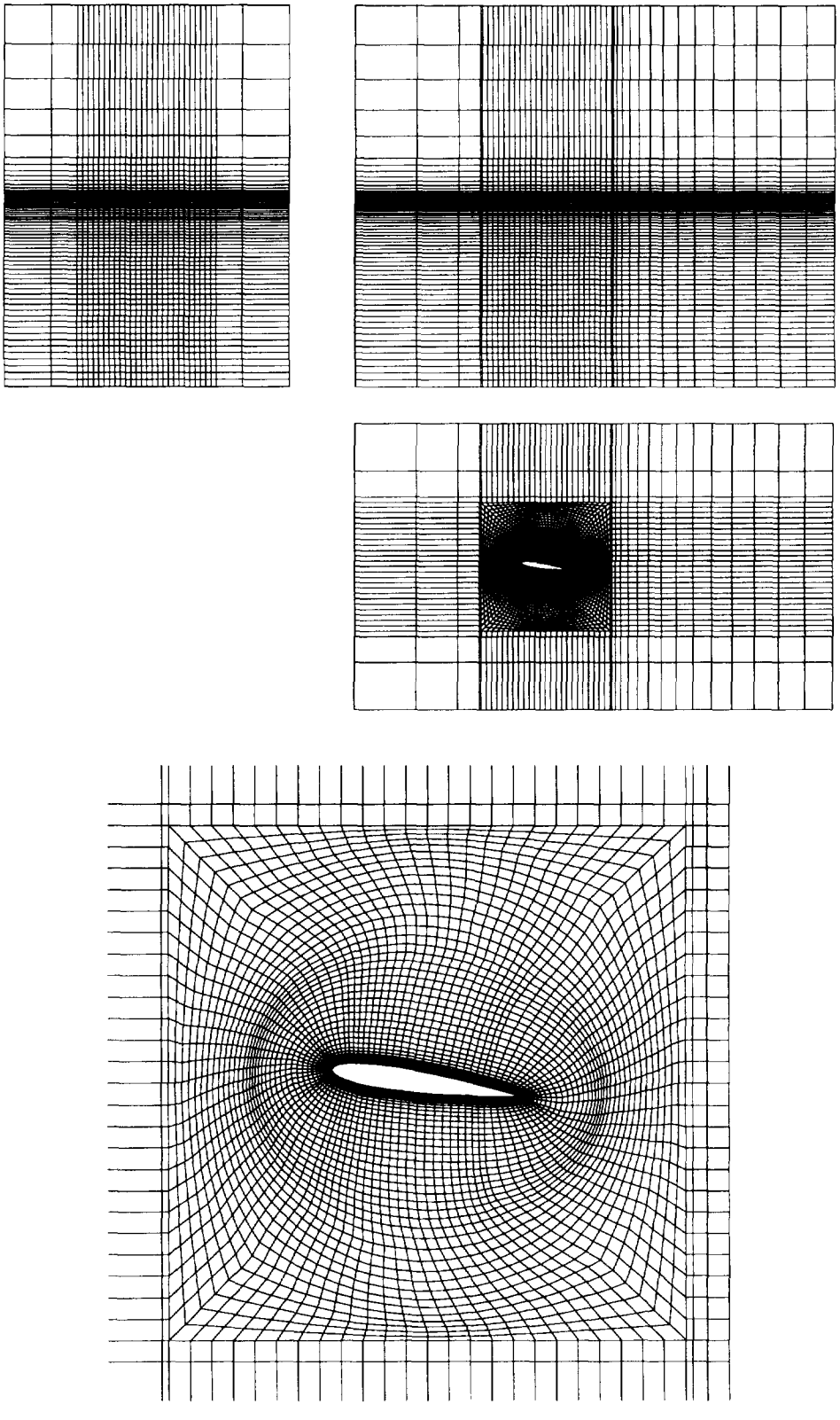

Fig. 12. Mesh for Subdomain-1.

The upstream, downstream and crossflow boundaries are located at 4.0, 6.0 and 3.0 units from the origin and the spanwise length of the domain is 8 units. SD-1 consists of 383 594 nodes and 370 300 hexahedral elements and results in 1 494 400 equations. The boundary conditions are uniform inflow velocity, zero-shear stress and zero-normal velocity at crossflow boundaries, traction-free condition at the outflow boundary and no-slip condition on the wing. The interface (OL-1 and IL-2) is selected in such a way that its upstream plane is at 1.39 unit length downstream of the origin.

SD-2 stretches 11.925, 6.0 and 8.0 units in the streamwise, crossflow and spanwise directions, respectively (see Fig. 13). SD-2 consists of 725 382 nodes and 700 000 hexahedral elements and results in 2 815 020 equations. The interface (OL-2 and IL-3) is selected in such a way that its upstream plane is at 12.315 units downstream of the origin (1.0 unit upstream of outflow boundary).

The trailing wing has a 0.5 unit chord length with an aspect ratio of 8. Its cross-section is a NACA 0012 and is also placed at 8.0° angle of attack (see Fig. 14). The center of this wing is located at 13.375 units downstream of the leading wing and 4 units in spanwise direction from the origin. In crossflow direction it is in the same position as the leading wing. SD-3 consists of 558 321 nodes, 3 208 013 tetrahedral elements and 1692 prismatic elements for the interface layer and results in 1 929 412 equations.

For these computations, the free-stream velocity is set to 1.0 and the time step size to 0.1. The size of the Krylov space is 20, without restart, and there are two nonlinear iterations at every time step.

Fig. 15 shows the streamwise and vertical components of the velocity for SD-1. Fig. 16 shows iso-surfaces of the streamwise and spanwise components of the vorticity for SD-1. These figures clearly depict the wing tip

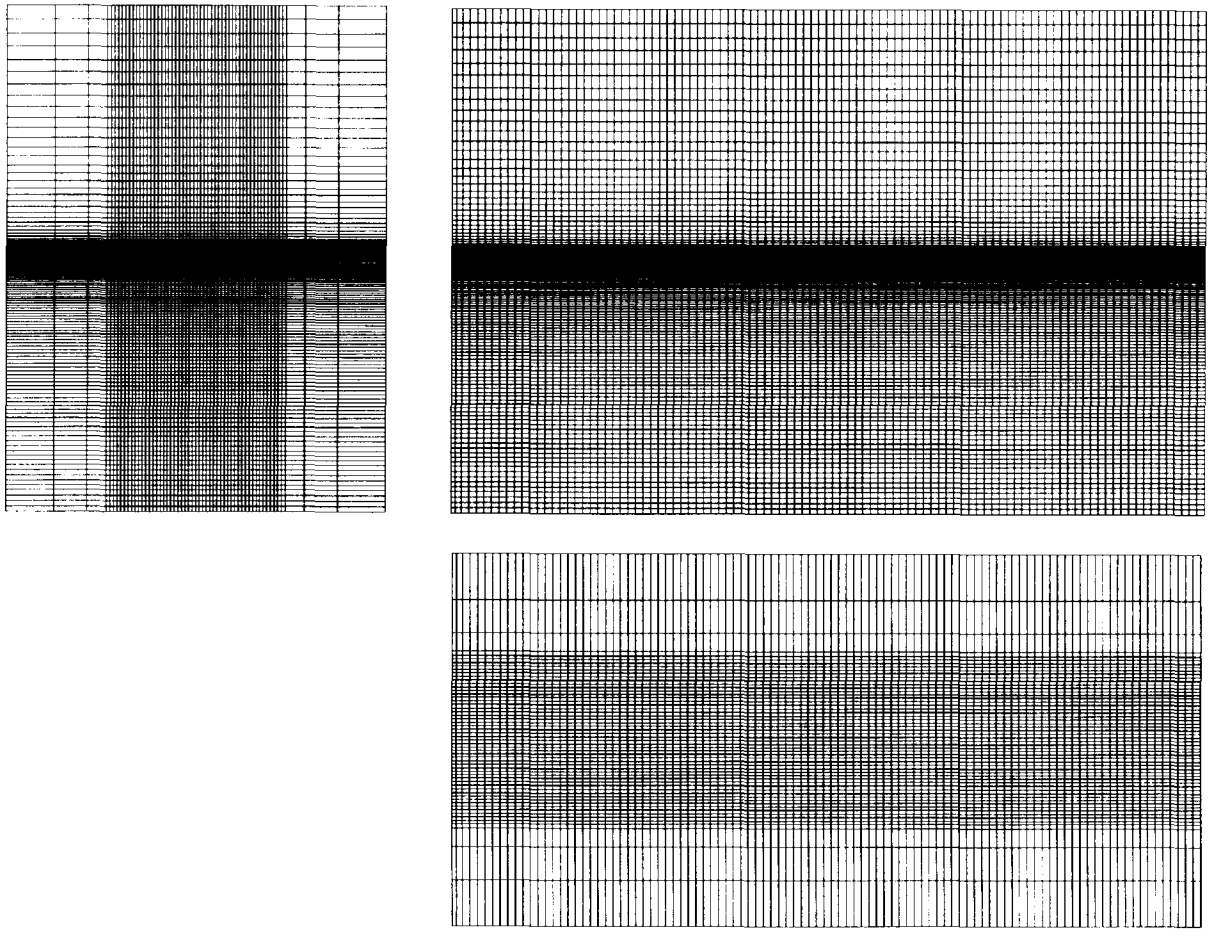

Fig. 13. Mesh for Subdomain-2.

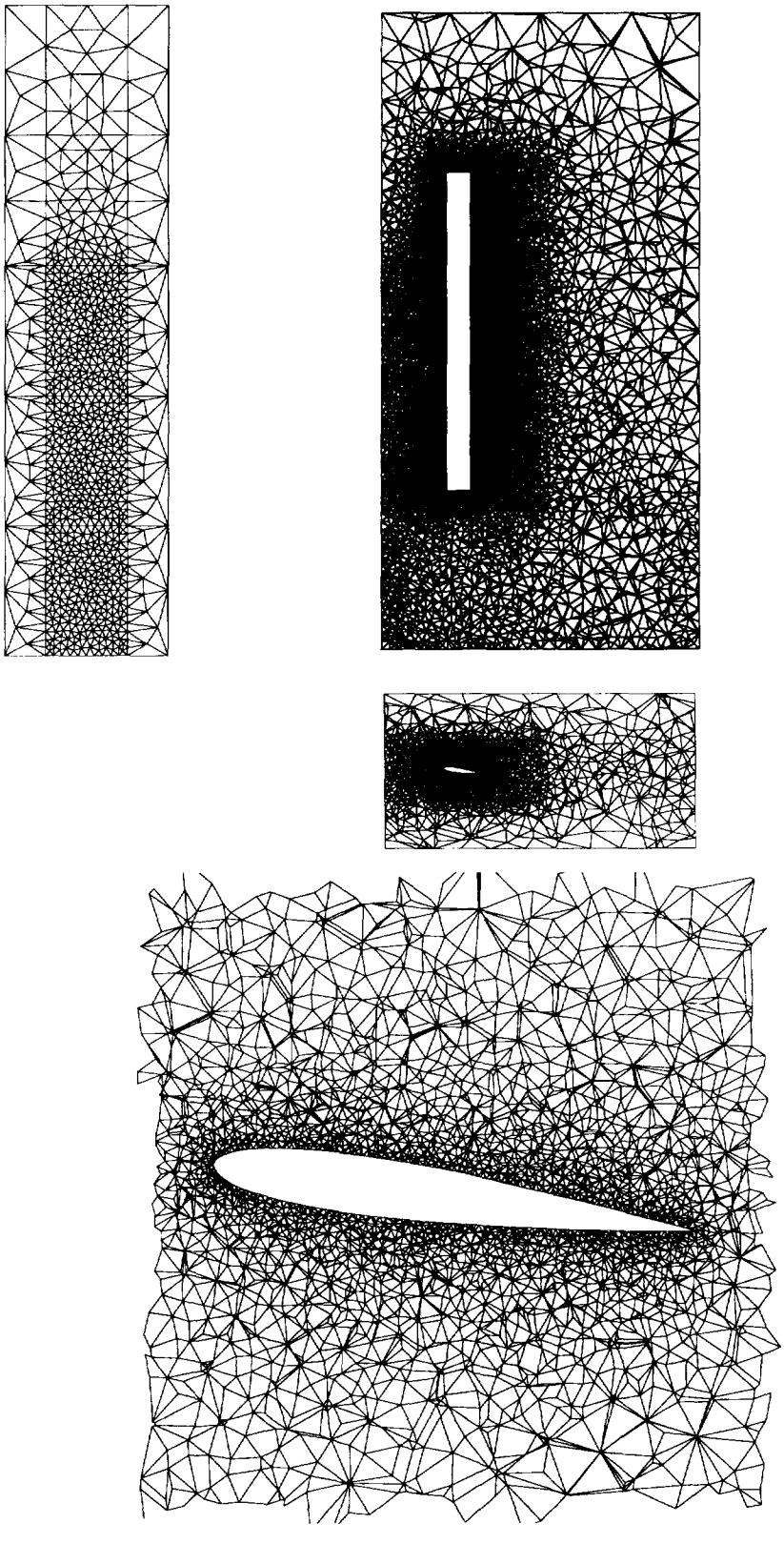

Fig. 14. Mesh for Subdomain-3.

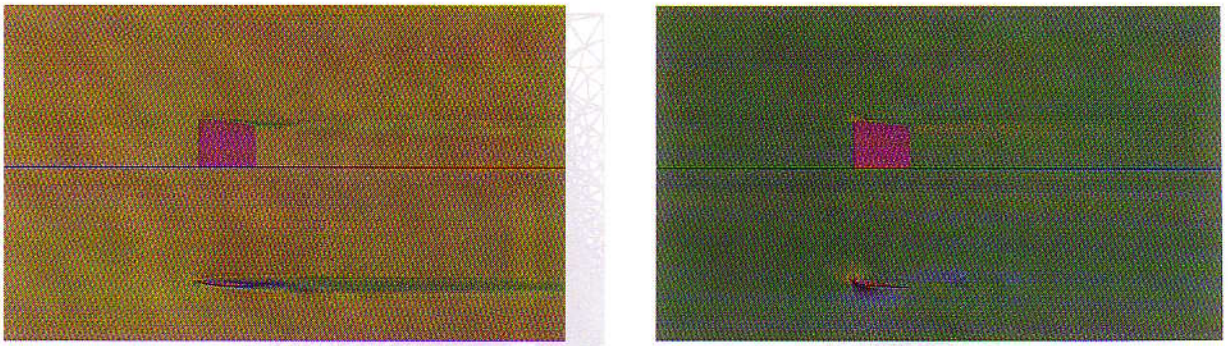

Fig. 15. Left: streamwise component of the velocity at two vertical planes; the right wing tip and the center. Right: vertical component of the velocity at the same planes.

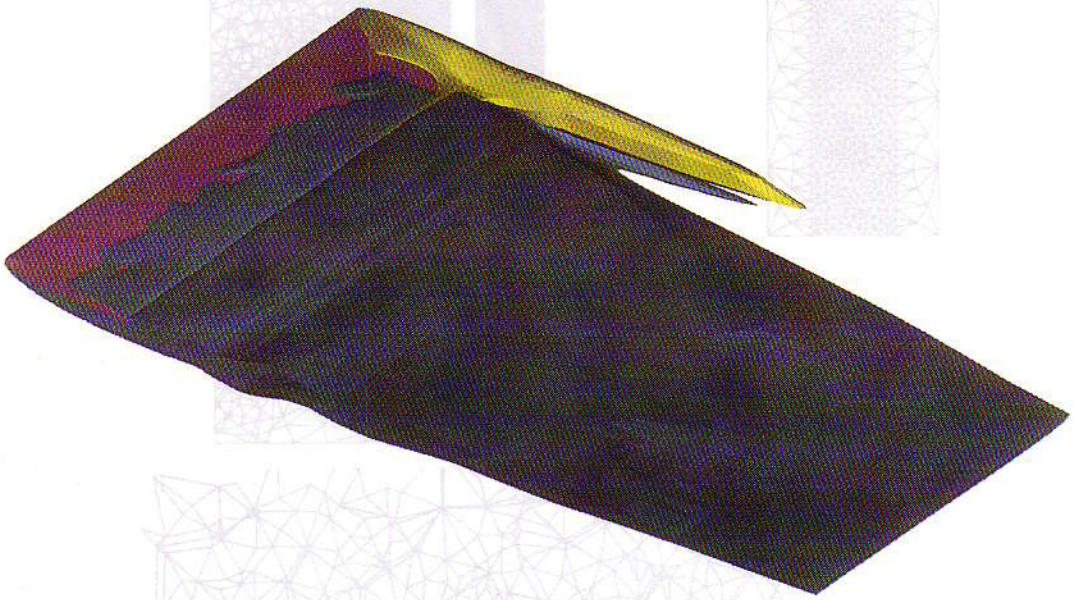

Fig. 16. Iso-surfaces of streamwise component of the vorticity corresponding to +1.2 value (light) and spanwise component of the vorticity corresponding to −1.2 value (dark).

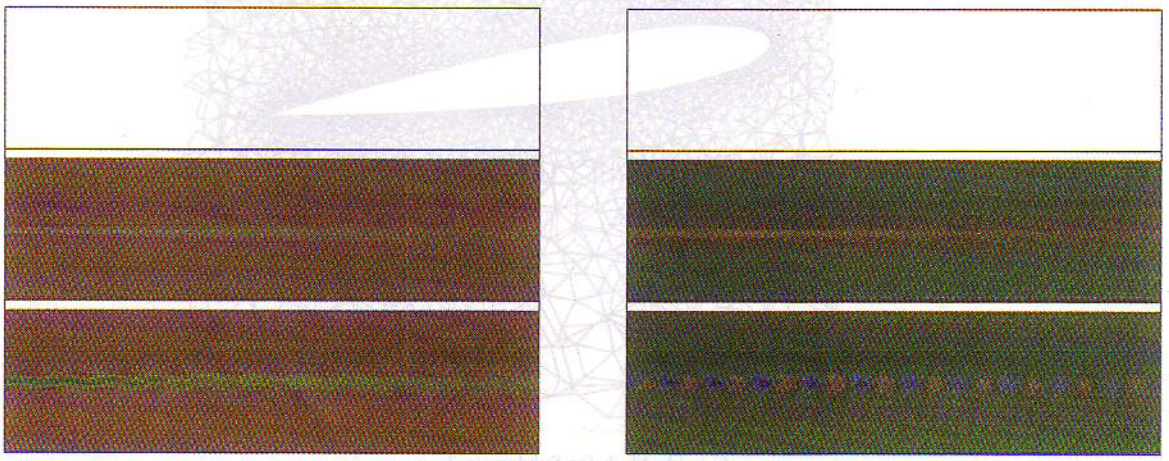

Fig. 17. Left: streamwise component of the velocity at two vertical planes; behind the right wing tip and the center of primary wing. Right: vertical component of the velocity at the same planes.

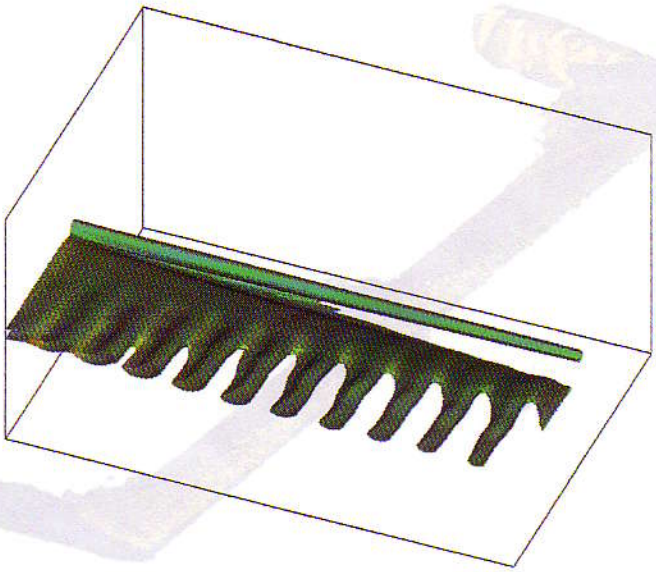

Fig. 18. Iso-surfaces of streamwise component of vorticity corresponding to 0.5 value (upper, light) and spanwise component of vorticity corresponding to −0.5 value (lower, dark).

vortex. We also observe vortex shedding from the center of the wing, which quickly dissipates downstream due to the coarseness of the mesh.

Figs. 17 and 18 show the same components of the velocity and vorticity for SD-2. The vortex structures are clearly captured in SD-2 though they were not captured in SD-1.

Fig. 19 shows the same components of the velocity for SD-3. Fig. 20 shows iso-surfaces of the streamwise component of the vorticity for SD-3. The left wing tip has more exposure to the wing tip vortices from the leading wing. Hence, the left wing tip vortex of this wing is weaker than the right one. The center part of the trailing wing interacts with the upper part of the tip vortex from the leading wing, since the tip vortex is displaced downwards as it propagates downstream of the leading wing. Fig. 21 shows the aerodynamic forces and moments acting on the trailing wing, compared to the case when that wing is in uniform flow. Of interest is the strong rolling moments created due to the influence of the wing tip vortices.

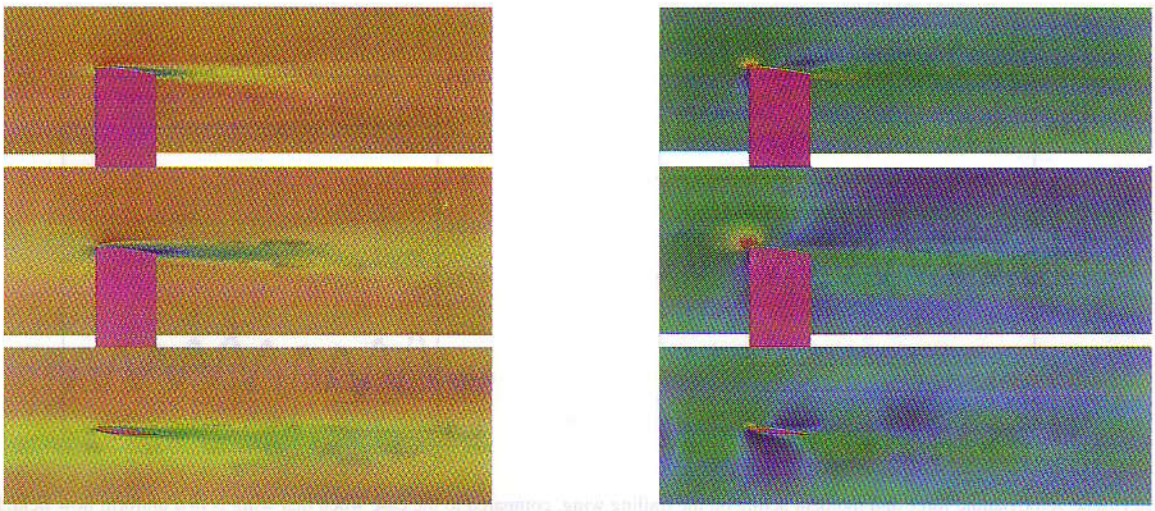

Fig. 19. Left: streamwise component of the velocity at three vertical planes: the right wing tip, the center and the left wing tip. Right: vertical component of the velocity at the same planes.

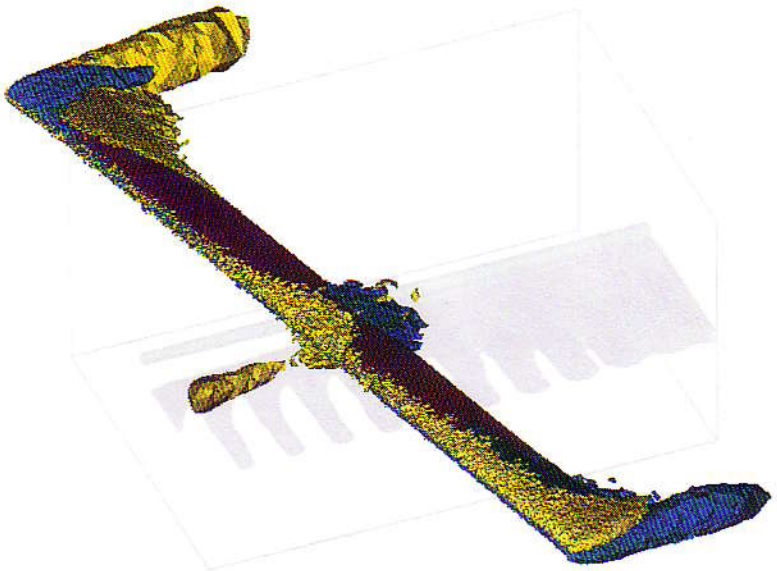

Fig. 20. Iso-surfaces of streamwise component of the vorticity corresponding to +0.5 value (light) and −0.5 value (dark).

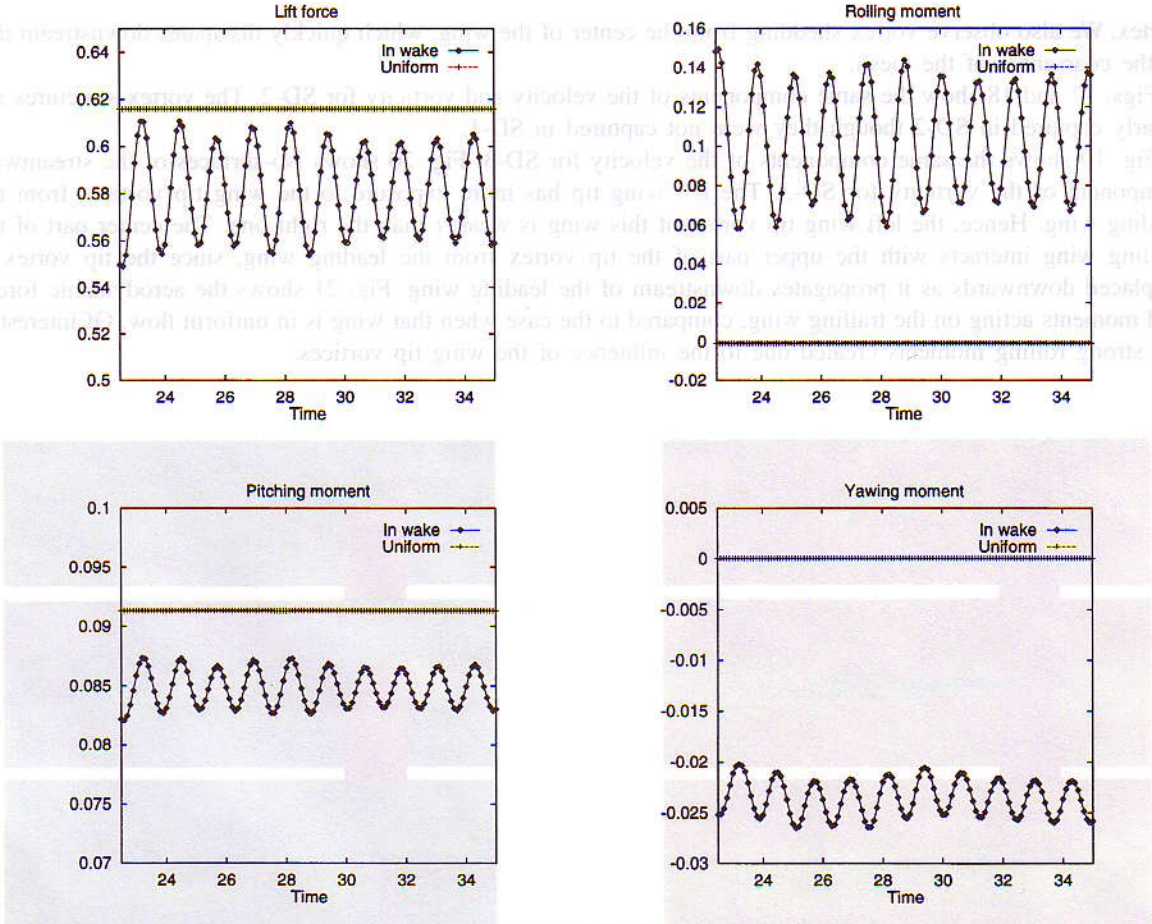

Fig. 21. The aerodynamic force and moment acting on the trailing wing, compared to the case when that wing is in a uniform flow field. Lift force (upper left), rolling moment (upper right), pitching moment (lower left) and yawing moment (lower right).

8. Concluding remarks

We presented a parallel multi-domain computational method for computation of wake flows. For structured meshes we designed a special-purpose implementation of the finite element formulation which is three-fold faster and still maintains accuracy. With the multi-domain method, we were able to capture the detailed wake behavior behind a circular cylinder, as well as the aerodynamic effect of the tip vortices and vortex shedding released by a wing on a second wing placed downstream in the far wake.

Acknowledgments

This work was supported by the ARO (grant DAAH04-93-G-0514), NASA (grant NAG9-919) and by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008. The content does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. CRAY time was provided in part by the Minnesota Supercomputer Institute. The first author has been supported by the Bridgestone Corp.

References

[1] A.N. Brooks and T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259. 1992
[2] T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Adv. Appl. Mech. 28 (1991) 1–44.
[3] Y. Saad and M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[4] Z. Johan, T.J.R. Hughes and F. Shakib, A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids, Comput. Methods Appl. Mech. Engrg. 87 (1991) 281–304.
[5] V. Kalro and T. Tezduyar, Parallel iterative computational methods for 3D finite element flow simulations, Comput. Assist. Mech. Engrg. Sci. 5 (1998) 173–183.
[6] C.H.K. Williamson, The existence of two stages in the transition to three-dimensionality of a circular cylinder wake, Phys. Fluids 31 (1988) 3165–3167.
[7] C.H.K. Williamson, Vortex dynamics in the cylinder wake, Ann. Rev. Fluid Mech. 28 (1996) 477–539.
[8] A. Roshko, On the development of turbulent wakes from vortex street, NACA report 1191, NACA, 1954.
[9] M. Coutanceau and J.R. Defaye, Circular cylinder wake configurations: a flow visualization survey, Appl. Mech. Rev. 44 (1991) 255–305.
[10] George E. Karniadakis and George S. Triantafyllou, Three-dimensional dynamics and transition to turbulence in the wake of bluff bodies, J. Fluid Mech. 238 (1992) 1–30.
[11] V. Kalro and T.E. Tezduyar, Parallel 3D computation of unsteady flows around circular cylinder, Parallel Comput. 23 (1997) 1235–1248.
[12] A.A. Johnson and T.E. Tezduyar, 3D simulation of fluid-particle interactions with the number of particles reaching 100, Comput. Methods Appl. Mech. Engrg. 145 (1997) 301–321.
[13] P.M. Gresho, S.T. Chan, R.L. Lee and C.D. Upson, A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 1: Theory, Int. J. Numer. Methods Fluids, 4 (1984) 557–598.
[14] M. Mallet, C. Poirier and F. Shakib, A new finite element formulation for computational fluid dynamics: Development of an hourglass control operator for multidimensional advective–diffusive systems, Comput. Methods Appl. Mech. Engrg. 94 (1992) 429–449.
[15] M.A. Christon, A domain-decomposition message-passing approach to transient viscous incompressible flow using explicit time integration, Comput. Methods Appl. Mech. Engrg. 48 (1997) 329–352.
[16] A. Mizukami, Some integration formulas for a four-node isoparametric element, Comput. Methods Appl. Mech. Engrg. 59 (1986) 111–121.
[17] S. Ohta, A. Maruoka, H. Hirano and M. Kawahara, 3-dimensional analysis of incompressible flow around circular cylinder, Proc. 1996 ASME Fluid Engineering Division Summer Meeting, 238 (1996) 445–450.