Journal of Computational Physics 229 (2010) 5541–5563

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A new domain decomposition method with overlapping patches for ultrascale simulations: Application to biological flows

L. Grinberg, G.E. Karniadakis [*]

Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

A R T I C L E   I N F O

Article history:
Received 12 November 2009
Received in revised form 12 February 2010
Accepted 8 April 2010
Available online 18 April 2010

Keywords:
CFD
Petaflop computing
Spectral element method
Scalability
Multi-domain decomposition
Arterial network

A B S T R A C T

We address the failure in scalability of large-scale parallel simulations that are based on (semi-)implicit time-stepping and hence on the solution of linear systems on thousands of processors. We develop a general algorithmic framework based on domain decomposition that removes the scalability limitations and leads to optimal allocation of available computational resources. It is a non-intrusive approach as it does not require modification of existing codes. Specifically, we present here a two-stage domain decomposition method for the Navier–Stokes equations that combines features of discontinuous and continuous Galerkin formulations. At the first stage the domain is subdivided into overlapping patches and within each patch a C^0 spectral element discretization (second stage) is employed. Solution within each patch is obtained separately by applying an efficient parallel solver. Proper inter-patch boundary conditions are developed to provide solution continuity, while a Multilevel Communicating Interface (MCI) is developed to provide efficient communication between the non-overlapping groups of processors of each patch. The overall *strong* scaling of the method depends on the number of patches and on the scalability of the standard solver within each patch. This dual path to scalability provides great flexibility in balancing accuracy with parallel efficiency. The accuracy of the method has been evaluated in solutions of steady and unsteady 3D flow problems including blood flow in the human intracranial arterial tree. Benchmarks on BlueGene/P, CRAY XT5 and Sun Constellation Linux Cluster have demonstrated good performance on up to 96,000 cores, solving up to 8.21B degrees of freedom in unsteady flow problem. The proposed method is general and can be potentially used with other discretization methods or in other applications.

1. Introduction

The scalability bottleneck: The main obstacle to realizing petaflop speeds on multi-petaflop-scale systems and beyond is scalability of algorithms for large-scale problems [1,2]. For time-dependent problems in particular governed by PDEs, implicit or semi-implicit temporal discretization leads to solution of very large linear systems, e.g., the solution of Poisson equation. Current solvers typically fail to scale beyond 1000 processors whereas the new petaflop systems are based on more than 100,000 processors. Even a solver with $\mathcal{O}(N^{3/2})$ scaling takes more than 90% of the total computation time whereas the remaining 10% is spent on the $\mathcal{O}(N)$ complexity part of the code. Hence, the entire computation time is almost "all-solver" when the size of the problem N increases. Moreover, the condition number of the corresponding matrix is extremely large and while effective preconditioners can reduce it substantially, such preconditioners are typically not scalable. Progress can be made by re-formulating existing parallel algorithms to first avoid solving one monolithic big system, and second to

[*] Corresponding author. Tel.: +1 401 863 1217; fax: +1 401 863 3369.
E-mail address: gk@dam.brown.edu (G.E. Karniadakis).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.014

provide flexibility on how to assign subdomains onto subsets of nodes/processors in existing computer configurations. However, it is not always possible to perform significant modifications to existing well-tested parallel PDE solvers and it is preferable to follow a "non-intrusive" upgrade, i.e., to re-use existing codes.

Domain decomposition: The approach we follow here is to break up the domain into "patches" and to couple an existing solver (applied to compute a solution within each patch) to itself through a set of proper interface boundary conditions between patches. We have tested this approach in preliminary work in [3], where we discussed such multi-patch domain decomposition method for spectral/*hp* element discretization (SEM). The method presented in [3] considered decomposition of the large computational domains into several non-overlapping patches. However, this approach is not always stable and it cannot preserve the so-called "spectral-accuracy" of SEM. In the current study we introduce *overlapping* 3D subdomains that lead to stable simulations and also improve the spatio-temporal accuracy, essentially recovering full spectral accuracy. The new method based on domain-decomposition techniques and overlapping patches is more robust, however, the use of overlapping patches increases somewhat the computational complexity.

Literature review: There have been several approaches for solving PDEs using different domain-decomposition techniques, mostly with the objective of constructing effective preconditioners. Here we review some works that share similar features with our approach.

Chen and Lazarov [4] studied domain splitting method for solving mixed finite element approximations to parabolic initial boundary problems. Their method was designed to produce a non-iterative time-stepping scheme, i.e., solution in each subdomain was performed separately and no sub-iterations were required for convergence of the numerical solution at the subdomain interfaces. Instead, local averaging was applied to enhance the accuracy of the solution at the subdomain interfaces. This paper also provides a proof of stability of the algorithm in the L_2-norm and analyzes its accuracy.

Application of overlapping domain decomposition methods for solving scalar convection–diffusion problems in 2D domains in conjunction with a finite difference scheme was analyzed by Hebeker and Kuznetsov [5]. The authors emphasized that the accuracy and stability of the solver depends on the size of the overlapping region. It was also suggested that in unsteady problems the inter-patch boundary conditions can be extrapolated from the data computed in previous time steps, however, no numerical results based on the extrapolation were provided.

Hebeker [6] considered decomposition of the domain into overlapping patches for solving unsteady two-dimensional heat conduction problems. Inside each patch a finite element discretization was implemented. The author formulated a non-iterative overlapping domain splitting scheme to solve the linear system while the boundary data at the subdomain interfaces were obtained by extrapolation from the previous time steps. It was recommended that the width of the overlapping region should be three to four elements to allow the numerical perturbations present due to the inter-patch conditions to decay.

Wu and Zou [7,8] proposed a grid-overlapping parallel method for implicit schemes. The authors employed finite difference scheme, with one-sided derivatives based on upwinding considerations and time-lagging treatment for inter-patch conditions. Their method was applied to simulate compressible Euler flow, and similar convergence rate to steady state was observed with and without multi-patch discretizations.

One of the main arguments for justification of the stability and accuracy of the methods discussed in [4–8] was the exponential decay (in the normal inward direction to the boundary) of perturbations present at the boundaries of the overlapping regions. For sufficiently large overlap, the magnitude of the perturbations at certain distance from the interface boundary is comparable or smaller than the time–space-discretization parameters ($(\Delta t)^k$, $(\Delta x)^p$, $k, p \geqslant 1$), hence the accuracy is not degraded. At the same time, the increased overlapping width (and multiple overlapping interfaces) required additional computational work, hence reducing the parallel efficiency of the solvers.

Tai et al. [9] and later Wang et al. [10] employed a coarse-mesh-free characteristic domain decomposition method for unsteady convection–diffusion equations. An Eulerian–Lagrangian method was employed to prescribe interface boundary conditions at the subdomain boundaries. The authors employed Robin and Dirichlet conditions as interface matching conditions. The Dirichlet conditions were imposed at the outlet boundaries where $\mathbf{v} \cdot \mathbf{n} > 0$, while the Robin conditions were imposed at the inlets of the subdomains $\mathbf{v} \cdot \mathbf{n} < 0$; here $\mathbf{n}$ denotes the outward normal and $\mathbf{v}$ the *prescribed* velocity vector. This method was tested in a transient two-dimensional homogeneous convection–diffusion PDE with a variable velocity field, and favorable results were reported.

The aforementioned methods considered explicit treatment of the inter-patch boundary condition and conforming (at interfaces) meshes. There is a series of publications considering implicit solvers and matching solutions obtained in patches with non-conforming meshes or different discretization (i.e., spectral/*hp*, finite element, etc.) by introducing mortar elements, see Bernardi et al. [11], Cai et al. [12] and references herein.

Implementation of overlapping domains for solution of PDEs was advocated by Bjørstad and Widlund [13], Dryja and Widlund [14], Kim and Widlund [15], Dohrmann et al. [16], Pavarino and Zampieri [17] and others. In these works, overlapping Schwarz preconditioners for iterative solvers were developed in conjunction with finite element and also spectaral/*hp* element discretizations. Use of overlapping subdomains for preconditioning typically leads to faster convergence compared to the non-overlapping approach. It was also observed that geometrically smooth interfaces bounding the overlapping regions lead to faster convergence of the iterative solver. Typically, but not exclusively, the number of overlaps formed following the aforementioned approaches depends on the number of partitions, which increases significantly the computational workload.

Paraschivoiu et al. [18] presented a multi-domain multi-model formulation for simulation of compressible flows. The method considers breaking up the domain into several regions. Different models are then employed to compute a solution in each region, and global continuity is achieved by proper interface boundary conditions.

The multi-domain approach is general and can be applied in simulation of many physical phenomena, e.g., contact problems (see works of a group of B. Wohlmuth, for example [19] and references therein); fluid–structure interaction, thermal modeling and many more. An efficient implementation of the multi-domain methods depends primarily on the robustness of the interface conditions and multi-level parallelism for message passing between different processor-groups.

Current objectives: In this study we develop a new domain decomposition approach in order to perform numerical simulations of 3D *incompressible unsteady flows*. For solution of the tightly coupled problem *within each patch* we re-use a well-tested *spectral-hp element* flow solver *NεκTαrG*, developed at Brown University [20,21]. *NεκTαrG* employs a very effective parallel low-energy basis preconditioner (LEBP) [22,23]. Its scalability is similar to the embarrassingly parallel diagonal (or block-diagonal) preconditioner on thousands of computer processors, but at the same time it leads to computational savings of an order of magnitude with respect to other preconditioners.

We employ a non-iterative high-order time-stepping scheme, that is, sub-step iterations due to multi-patch partitioning are not required. The number of overlapping patches is independent of the number of computer processors, and it is typically very low. Moreover, since the overlapping regions are formed during the mesh generation their boundaries are smooth (planar). Forming overlapping regions while generating a mesh also helps to control the number of elements within a specific region (patch). In our simulations, the ratio of duplicated spectral elements to the total number of elements is typically 1/50 to 1/1000, hence the computational overhead is negligible. Most of the benchmarks we adopted are for well-known flows but we also present a very large-scale simulation of the intracranial arterial tree for a patient-specific geometry that contains an aneurysm, depicted in Fig. 1.

The paper is organized as follows: In section 2 we review the solution of flow equations and highlight some of the main issues that need to be resolved in order to make 3D simulations of blood flow in the entire human arterial tree feasible in the near future. Subsequently, we present technical details of our new approach, specifically we discuss the two-stage method that has features of discontinuous and continuous Galerkin formulations. We pay particular attention to implementing the data exchange at the inter-patch interfaces, using a multilevel communicating interface (MCI). In section 3 we present the accuracy and computational efficiency in simulations with the new method. In section 4 we conclude with a brief summary and discussion. In Appendix A we provide details on MCI.

2. Methods

In this section we first review the high-order time-stepping scheme applied to Navier–Stokes equations in a monolithic domain. Second, we provide the numerical scheme for solving the Navier–Stokes equation using the multi-patch decompo-

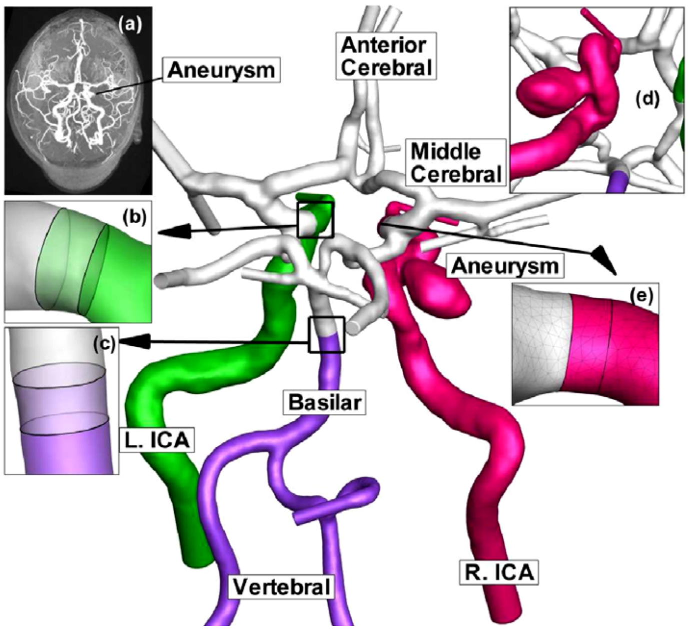

Fig. 1. Domain of intracranial vessels with aneurysm (d). The geometry was reconstructed from a set of 2D MRI images (a). The domain is subdivided into four overlapping patches marked by different colors. The overlapping regions are depicted in (b,c,e). MRI data – courtesy of T. Anor and Dr. J. Madsen, Harvard Medical School. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sition. We will denote the new (multi-patch) method by 2DD, where "2" refers to two levels of granularity in decomposing the computational domain. On the first level the domain Ω is decomposed into Np overlapping patches $\Omega_i \subset \Omega$, $i = 1,\ldots,Np$, while on the second level each patch is subdivided into non-overlapping spectral/hp elements. The notation 1DD will be used to denote the monolithic (standard) domain decomposition, i.e., $Np = 1$.

2.1. Discretization

We consider 3D unsteady incompressible flow in a rigid domain Ω; the flow is described by the Navier–Stokes equations:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot(\nabla\mathbf{v}) = -\nabla p + \nu\nabla^2\mathbf{v}, \nabla\cdot\mathbf{v} = 0, \tag{1}$$

where $\mathbf{v}$ is the velocity vector, p is the pressure, t is time and ν is the kinematic viscosity of a fluid. For spatial discretization we implement the spectral/*hp* element method (SEM). The computational domain Ω is decomposed into a set of polymorphic non-overlapping elements $\Omega^{e_i} \subset \Omega,\ i = 1,\ldots,Nel$, as illustrated in Fig. 2. Within each element the solution is approximated in terms of hierarchical, mixed-order, semi-orthogonal Jacobi polynomial expansions [20]. They are hierarchical in a sense that the modes are separated into vertex (linear term) $\Phi_k(\mathbf{x})$, edge $\Psi_k(\mathbf{x})$, face $\Theta_k(\mathbf{x})$ and interior or bubble modes $\Lambda_k(\mathbf{x})$. The polynomial approximation of a field $\mathbf{v}(t,\mathbf{x})$ at any point $\mathbf{x}_j$ is given by

$$\mathbf{v}(t,\mathbf{x}_j) = \sum_{k=1}^{Nv}\hat{\mathbf{v}}_k^V(t)\Phi_k(\mathbf{x}_j) + \sum_{k=1}^{Ne}\hat{\mathbf{v}}_k^E(t)\Psi_k(\mathbf{x}_j) + \sum_{k=1}^{Nf}\hat{\mathbf{v}}_k^F(t)\Theta_k(\mathbf{x}_j) + \sum_{k=1}^{Ni}\hat{\mathbf{v}}_k^I(t)\Lambda_k(\mathbf{x}_j), \tag{2}$$

where Nv, Ne, Nf and Ni are the number of vertex, edge, face and interior modes, respectively, while the quantities with hat denote modal amplitudes.

In order to solve Eq. (1) numerically we decouple the velocity and the pressure fields by applying a high-order time-splitting scheme; for the complete analysis of the scheme we refer to [20,24]. First, the provisional field $\mathbf{v}^*$ is computed in physical space (on the quadrature grid) using formula (3a). Second, we apply a Galerkin projection to obtain the weak formulations for the velocity and pressure variables, i.e.

$$\mathbf{v}^* = \sum_{k=0}^{Je-1}\alpha_k\mathbf{v}^{n-k} - \Delta t\left(\sum_{k=0}^{Je-1}\beta_k(\mathbf{nl})^{n-k} + \mathbf{f}\right), \quad \mathbf{nl} = \mathbf{v}\cdot(\nabla\mathbf{v}) \tag{3a}$$

$$\mathbf{L}\hat{p} = -\frac{1}{\Delta t}(\nabla\cdot\mathbf{v}^*,\phi) + \left(\frac{\partial p}{\partial n},\phi\right) - \mathbf{L}^D\hat{p}^D \tag{3b}$$

$$\mathbf{H}\hat{\mathbf{v}}^{n+1} = \frac{1}{\gamma_0}(\mathbf{v}^* - \Delta t\nabla p,\phi) + \frac{\Delta t\nu}{\gamma_0}\left(\frac{\partial\mathbf{v}}{\partial n},\phi\right)^{n+1} - \mathbf{H}^D\hat{\mathbf{v}}^{n+1,D}, \tag{3c}$$

where γ_0, α_k are coefficients of backward differentiation formula, β_k are coefficients of Adams–Bashforth integrator, Je is the order of the time discretization scheme, $\mathbf{H} = \mathbf{M} - \frac{\Delta t\nu}{\gamma_0}\mathbf{L}$, and $\mathbf{M}$ and $\mathbf{L}$ are the mass and stiffness matrices, respectively. Also, $\hat{\mathbf{v}}$ and $\hat{p}$ are *unknown* velocity and pressure variables, while $\hat{\mathbf{v}}^{\mathbf{D}}$ and $\hat{p}^D$ are *known* velocity and pressure variables. The last term in Eqs. (3b) and (3c) is due to lifting (subtraction) a known solution to impose Dirichlet boundary conditions [20]. The upper script D indicates that the operators H^D and L^D are constructed only for the boundaries, where Dirichlet boundary conditions are imposed. We consider an arterial network as the target domain, with boundaries the vessel walls and also the multiple inlets and outlets. The schematic representation of the domain boundaries is provided in Fig. 3. At the inlets (Ω_{in}), a Dirichlet

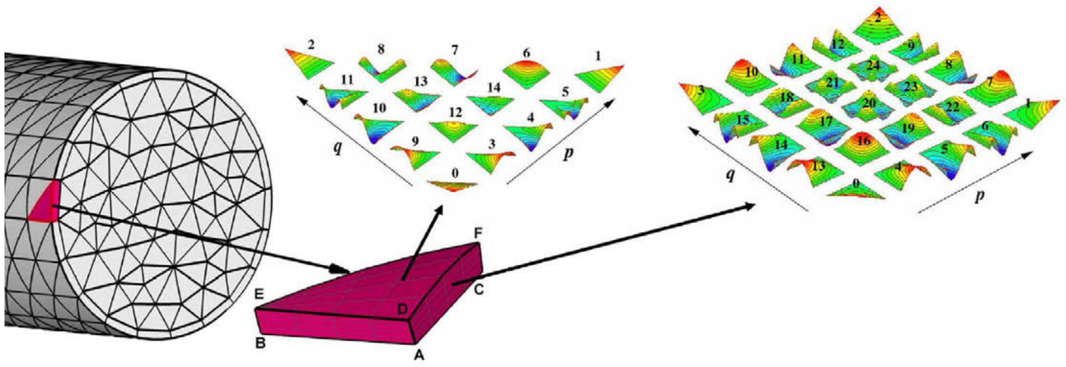

Fig. 2. Illustration of the unstructured surface grid and the polynomial basis employed in *NεκTαrG*. The solution domain (patch) is decomposed into non-overlapping elements. Within each element the solution is approximated by vertex, edge, face and (in 3D) interior modes. The shape functions associated with the vertex, edge and face modes for fourth-order polynomial expansion defined on triangular and quadrilateral elements are shown in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

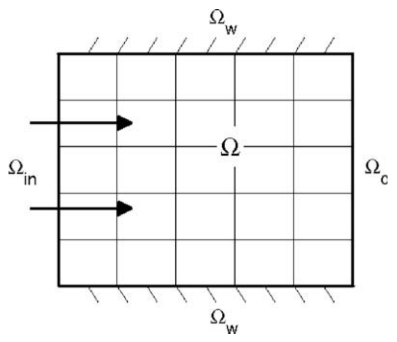

Fig. 3. Illustration of computational domain Ω discretized into quadrilateral elements. Arrows indicate the primary direction of the flow. Ω_{in} and Ω_o are the inlet and outlet boundaries; Ω_w correspond to the impermeable domain boundaries (walls).

boundary condition for the velocity field is imposed. At the outlets (Ω_o) we impose a Dirichlet boundary condition for the pressure and zero-Neumann boundary condition for the velocity. The Neumann pressure boundary condition at the boundaries with prescribed velocity (Ω_{in} and Ω_w) are computed from

$$\frac{\partial p}{\partial n} = \left[-\frac{\gamma_0 \mathbf{v}^{n+1} - \sum_{k=0}^{Je-1}(\alpha_k \mathbf{v}^{n-k})}{\Delta t} - \sum_{k=0}^{Je-1} [\beta_k (\mathbf{nl} + \nu \nabla \times (\nabla \times \mathbf{v}))]^{n-k} \right] \cdot \mathbf{n}. \tag{4}$$

Note, that the pressure boundary condition must be *provided* only at Ω_o, while at other boundaries it is *computed* locally from the velocity field. We will exploit this feature in designing inter-patch pressure boundary condition, e.g., the type of boundary conditions imposed at the boundaries of a patch Ω_i are identical to those for the full domain Ω.

In the C^0 approximation the global linear operators **H** and **L** (Eqs. (3b) and (3c)) are constructed from the local ones by *static condensation*. Due to sharing of the boundary degrees of freedom, the rank of the global operator is lower than the total number of local degrees of freedom (DOF), however, for large computational domains the rank of **H** and **L** is still very large, and consequently they have high condition number κ. In solving systems (3b) and (3c) considerable computational effort is required to invert the global linear operators with very high κ. Use of effective preconditioners can significantly reduce κ, however, most of the available *effective* parallel preconditioners do not scale well on thousands of processors.

2.2. Multi-patch Domain Decomposition (2DD) Method

To overcome the aforementioned problem we decompose the large computational domain into a series of *weakly coupled* subdomains or patches of *manageable* size for which high parallel efficiency can be achieved on a sub-cluster of processors. For example, the brain vasculature domain presented in Fig. 1 has been decomposed into four patches. The C^0 Galerkin projection is implemented *within* the patch. On the other hand, continuity of numerical solution across different patches is achieved by providing appropriate inter-patch boundary conditions (IPC). The 2DD is equivalent to decomposing the linear operators **H** and **L** into a sequence of *non-overlapping* operators of small size. Although the operators are statically condensed from the elementwise operators $\mathbf{H}^e$ and $\mathbf{L}^e$, the static condensation is performed within each patch *independently*. The method preserves the advantages of C^0 descritization, namely low number of degrees-of-freedom (DOFs) compared to a full discontinuous Galerkin (DG) formulation, and implicit treatment of the viscous terms of Navier–Stokes equation within a patch. The entire simulation is performed in two steps:

1. Solutions computed at time step t^n from both sides of the interpatch interfaces are exchanged and the boundary conditions required to compute the solution at time step t^{n+1} are computed explicitly as will be explained bellow.
2. Navier–Stokes equations in each patch Ω_i, $i = 1,\ldots,Np$ are solved on different groups of processors using a semi-implicit time-stepping scheme and C^0 polynomial approximation. At this step no interpatch communication is performed. From the computational perspective this step can be compared to solving concurrently Np unrelated problems on Np non-overlapping groups of computer processors.

In the following we provide the numerical scheme for the solution of a Navier–Stokes equations defined in a computational domain subdivided into two patches, as illustrated in Fig. 4. Extension to problems with more than two patches is straightforward.

Consider the computational domain $\Omega \in R^3$ subdivided into two non-overlapping patches Ω^A and Ω^B, and the interpatch intersection $\Omega^A \cap \Omega^B = \Gamma$, (see Fig. 4). Let us denote by $\mathbf{v}^A$ ($\mathbf{v}^B$) and p^A (p^B) the solution of the Navier–Stokes equation (1) in Ω^A (Ω^B). The velocity and pressure fields in Ω^A and Ω^B satisfy the following boundary conditions:

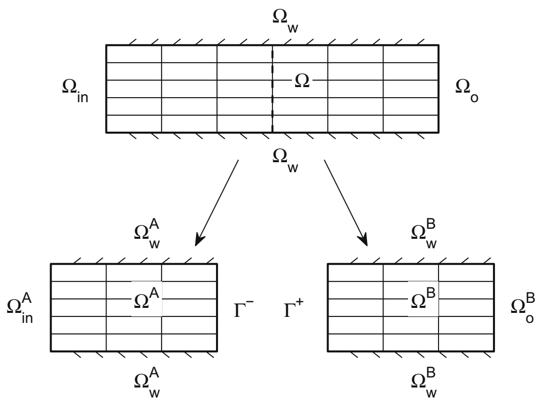

Fig. 4. Two-stage domain decomposition (2DD): patches Ω_A and Ω_B are connected by the interface boundary conditions. Velocity computed at the outlet of Ω_A (Γ^-) is imposed as boundary condition at the inlet of Ω_B (Γ^+); pressure and velocity flux computed at Γ^+ are imposed as boundary conditions at Γ^-.

$$\mathbf{v}^A|_{\Omega^A_{in}} = \mathbf{v}|_{\Omega_{in}}, \quad \mathbf{v}^A|_{\Omega^A_w} = \mathbf{v}|_{\Omega_w}, \quad \mathbf{v}_B|^{\Omega^B_w} = \mathbf{v}|_{\Omega_w},$$
$$\left.\frac{\partial v^B}{\partial n}\right|_{\Omega^B_o} = \left.\frac{\partial v}{\partial n}\right|_{\Omega_o},$$
$$p|_{\Omega^B_o} = p|_{\Omega_o}.$$

At Γ the following interpatch boundary conditions are prescribed:

$$\mathbf{v}^B|_{\Gamma^+} = f(\mathbf{v}^A|_{\Gamma^-}),$$
$$\left.\frac{\partial v^A}{\partial n}\right|_{\Gamma^-} = h\left(\left.\frac{\partial v^A}{\partial n}\right|_{\Gamma^-}, \left.\frac{\partial v^B}{\partial n}\right|_{\Gamma^+}\right),$$
$$p^A|_{\Gamma^-} = g(p^A|_{\Gamma^-}, p^B|_{\Gamma^+}).$$

The Neumann boundary condition for the pressure at boundaries $\Omega^A_{in}, \Omega^B_{in}, \Omega^A_w, \Omega^B_w$ are computed locally using formula (4). We extend the discussion on the functional relationships f,h,g in the next section.

At each time step the following sets of equations are solved simultaneously on different non-overlapping groups of processors.

To obtain the solution in domain Ω_A we solve

$$\mathbf{v}^{A,*} = \sum_{k=0}^{Je-1} \alpha_k \mathbf{v}^{A,n-k} - \Delta t\left(\sum_{k=0}^{Je-1} \beta_k (\mathbf{nl})^{n-k} + \mathbf{f}\right), \quad \mathbf{nl} = \mathbf{v}^A \cdot (\nabla \mathbf{v}^A) \tag{5a}$$
$$\mathbf{L}^A \hat{p}^A = -\frac{1}{\Delta t}(\nabla \cdot \mathbf{v}^{A,*}, \phi) + \left.\left(\frac{\partial p^A}{\partial n}, \phi\right)\right|_{\Omega^A_{in},\Omega^A_w} - \mathbf{L}^{A,D} g(\hat{p}^A|_{\Gamma^-}, \hat{p}^B|_{\Gamma^+})|_{\Omega^A_o} \tag{5b}$$
$$\mathbf{H}^A \hat{\mathbf{v}}^{A,n+1} = \frac{1}{\gamma_0}(\mathbf{v}^{A,*} - \Delta t \nabla p^A, \phi) + \frac{\Delta t \nu}{\gamma_0}\left.\left(h\left(\left.\frac{\partial v^A}{\partial n}\right|_{\Gamma^-}, \left.\frac{\partial v^B}{\partial n}\right|_{\Gamma^+}\right), \phi\right)\right|_{\Omega^A_o} - \mathbf{H}^{A,D} \hat{\mathbf{v}}^{A,n+1,D}|_{\Omega^A_{in},\Omega^A_w}. \tag{5c}$$

To obtain the solution in domain Ω_B we solve

$$\mathbf{v}^{B,*} = \sum_{k=0}^{Je-1} \alpha_k \mathbf{v}^{B,n-k} - \Delta t\left(\sum_{k=0}^{Je-1} \beta_k (\mathbf{nl})^{n-k} + \mathbf{f}\right), \quad \mathbf{nl} = \mathbf{v}^B \cdot (\nabla \mathbf{v}^B) \tag{6a}$$
$$\mathbf{L}^B \hat{p}^B = -\frac{1}{\Delta t}(\nabla \cdot \mathbf{v}^{B,*}, \phi) + \left.\left(\frac{\partial p^B}{\partial n}, \phi\right)\right|_{\Omega^B_{in},\Omega^B_w} - \mathbf{L}^{B,D} \hat{p}^{B,D}|_{\Omega^B_o} \tag{6b}$$
$$\mathbf{H}^B \hat{\mathbf{v}}^{B,n+1} = \frac{1}{\gamma_0}(\mathbf{v}^{B,*} - \Delta t \nabla p^B, \phi) + \frac{\Delta t \nu}{\gamma_0}\left.\left(\frac{\partial \mathbf{v}^B}{\partial n}, \phi\right)^{n+1}\right|_{\Omega^B_o} - \mathbf{H}^{B,D} f(\hat{\mathbf{v}}^A|_{\Gamma^-})|_{\Omega^B_{in}}, \tag{6c}$$

From the numerical point of view, such multi-patch decomposition reduces the size of linear system and consequently the number of iterations. From the parallel computing standpoint, due to the weak coupling of the computational subdomains, a preconditioned conjugate gradient solver can be applied within each patch *independently*. Each patch is assigned to a sub-group of processors; these groups are not overlapping and can vary in size according to the number of DOFs within a patch. Thus, the very expensive *blocking* communication between processes can be restricted to sub-groups of processes only. However, to exchange data across the patch interfaces an additional communication is required. In the following sections we overview the IPC and describe our approach to mesh generation; a Multilayer Communicating Interface (MCI) designed to handle the inter- and intra-patch communications is discussed in Appendix A.

2.3. Inter-patch conditions (IPC)

The IPC required for coupling the 3D patches involve the velocity boundary condition at the inlets along with pressure and velocity fluxes at the outlets. An illustration of two non-overlapping patches coupled by IPC is presented in Fig. 4. In Fig. 5 we provide an illustration of two *overlapping* patches. In simulations with overlapping patches the data for the pressure and velocity IPCs are extracted from the inner elements of the patch, thus *decoupling* the pressure and velocity interfaces. The width of the overlap can be chosen according to the error estimates by [6,24]. Karniadakis et al. [24] have shown that the time-splitting scheme introduces a (numerical) boundary layer of thickness $\delta_t \approx (\nu\Delta t)^{J_e}$; correspondingly the error decays exponentially over distance s ($error \propto e^{-s/\delta_t}$) in the inward direction of the domain. Hebeker [6] proved that for the Helmholtz equation the disturbance introduced at the boundary of the domain decays exponentially in the inward direction normal to the boundary. According to [6] the error at the edge of the overlap can be estimated from $e^{-k\delta_{overlap}/\sqrt{\nu}}$, where k is a positive constant, $\delta_{overlap}$ is the overlap width, and ν is the coefficient of the diffusive term; note that according to [6] the rate of decay does not depend on the discretization parameters, except $\delta_{overlap}$. Also, in [6] it was suggested that the optimal overlapping region should be three to four elements wide. Thus, use of overlapping domains minimizes the numerical error in imposing IPC, which also enhances stability.

In the case of non-overlapping domains the patches Ω_A and Ω_B share a single interface Γ, while in the case of overlapping domains we distinguish between two interfaces Γ_{in} and Γ_{out} as illustrated in Fig. 5. For the sake of simplicity of notation and without loss of generality, in the following sections we will omit the subscripts "*in*" and "*out*" while referring to the inter-patch interface.

To impose the IPC we follow a procedure similar to the discontinuous Galerkin method [25]. The hyperbolic component of the Navier–Stokes equation dictates the choice of interface condition for the velocity based on the upwinding principle. Assuming that $\mathbf{v}\cdot\mathbf{n} \geqslant 0$ (with $\mathbf{n}$ pointing outward) at the patch outlet we impose the inlet velocity condition in patch B as follows:

$$\mathbf{v}^{B,n+1}|_{\Gamma^+} = f(\mathbf{v}^A|_{\Gamma^-}) = \sum_{k=0}^{J_{IPC}-1} \xi_k \mathbf{v}^{A,n-k}|_{\Gamma^-}, \tag{7}$$

where the superscripts denote time steps, ξ_k are the extrapolation coefficients and J_{IPC} is the order of the polynomial extrapolation. The velocity flux at the outlet of Ω_A is computed as a weighted average of the normal velocity derivatives from both sides of the interface, i.e.,

$$\left.\frac{\partial v^{A,n+1}}{\partial n}\right|_{\Gamma^-} = h\left(\left.\frac{\partial v^A}{\partial n}\right|_{\Gamma^-}, \left.\frac{\partial v^B}{\partial n}\right|_{\Gamma^+}\right) = \sum_{k=0}^{J_{IPC}-1} \xi_k \left(\lambda \left.\frac{\partial v^{A,n-k}}{\partial n}\right|_{\Gamma^-} - (1-\lambda)\left.\frac{\partial v^{B,n-k}}{\partial n}\right|_{\Gamma^+}\right), \tag{8}$$

where $0 \leqslant \lambda \leqslant 1$. In the current study we employed central averaging to impose the Neumann B.C. for the velocity, i.g., $\lambda = 0.5$. To impose alternating fluxes for velocity and velocity derivatives (appropriate for diffusion problem [28]) the parameter λ can be set to $\lambda = 0$. Alternative choices of numerical fluxes may also be considered; for example, imposing the total flux for the velocity at Γ^+ was advocated in [26,27,10]. Also, different choices for the flux in the *DG* formulation can be found in [28,29]. Zhang and Shu [28] analyzed three formulations for numerical fluxes for discontinuous Galerkin method. According to [28], for a hyperbolic problem the upwinding formulation is appropriate, while for a diffusion problem the use of alternating fluxes is a proper choice.

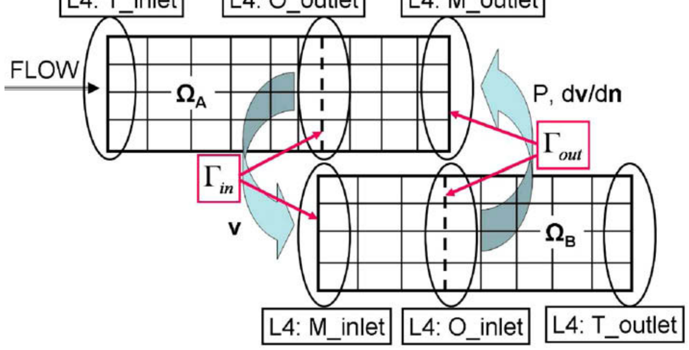

Fig. 5. Two-stage domain decomposition (2DD): schematic representation of two overlapping patches and interfaces.

The pressure at the patch outlet is given by

$$p^{A,n+1}|_{\Gamma^-} = g(p^A|_{\Gamma^-}, p^B|_{\Gamma^+}) = 1/2F(t)(p^{A,n}|_{\Gamma^-} + p^{B,n}|_{\Gamma^+}), \tag{9}$$

where $0 \leqslant F(t) = (1 - e^{-\alpha(t^n - t^0)})^\beta \leqslant 1$ is a filter function [3]; in the current study we use $\alpha = 20, \beta = 2$. The function $F(t)$ is applied to filter out erroneous pressure oscillations due to inconsistent initial conditions, for example $\nabla \cdot \mathbf{v} \neq 0$. Note that inter-patch boundary condition for the pressure is required only at one side of the interface (Γ^-) since the Neumann boundary condition for the pressure (Eq. (4)) at the (Γ^+) can be used. Depending on the numerical scheme and initial conditions, erroneous pressure oscillations at the inlet may be present during the first few time steps. Such oscillations do not reflect the physics of the problem and are associated with incompatible initial conditions that may lead to velocity divergence, i.e., $\nabla \cdot \mathbf{v}^{n=0} \neq 0$, and hence they must be filtered out [3]. The error introduced by the explicit treatment of the interface boundary conditions is controlled by the size of time step and the order of polynomial extrapolation J_{IPC} in Eq. (7).

In the SEM approach, the solution for velocity and pressure is performed in *modal* space, thus, the values of velocity and pressure can be transferred and imposed as boundary conditions in modal space, hence bypassing expensive transformations from modal to physical space and vice versa. Of course, this can be done only when the meshes at Γ^- and Γ^+ are conforming, otherwise use of mortar elements is appropriate. Imposing IPC in modal space has an additional advantage: we can exploit the hierarchical structure of the basis functions and reduce the communication associated with imposing IPC by transferring only the most energetic modes. Although a small reduction in accuracy may occur, from the computational standpoint, *limiting* the number of modes to be collected from one subset of processors to another leads to shorter messages and consequently to reduction in communication time associated with imposing IPC. The latter is very important in ultra-parallel simulations.

Special care should be taken when the velocity IPC are imposed with reduced resolution due to the following reason: Consider that the inlet boundary conditions at Ω^A_{in} are prescribed by a time varying function with period T, then imposing the velocity IPC with the full resolution will guarantee preservation of mass (and also higher order moments) over the time period T since

$$\int_T \int_\Gamma \mathbf{v}^A(t) \cdot \mathbf{n}^A \, d\Gamma^- \, dt = -\int_T \int_\Gamma \mathbf{v}^B(t) \cdot \mathbf{n}^B \, d\Gamma^+ \, dt,$$

while truncation of the polynomial expansion approximating the velocity field at Γ^+ ($\mathbf{v}^B_{trunc}$) may introduce analytical sources/sinks, i.e.,

$$\int_T \int_\Gamma \mathbf{v}^A(t) \cdot \mathbf{n}^A \, d\Gamma^- \, dt = -\int_T \int_\Gamma \mathbf{v}^B_{trunc}(t) \cdot \mathbf{n}^B \, d\Gamma^+ \, dt + Q_s,$$

where the negative sign is due to $\mathbf{n}^A = -\mathbf{n}^B$. In this study we perform numerical simulations with IPC for velocity and pressure imposed with both consistent ($P_{VBC} = P_{PBC} = P$) and reduced spatial resolutions, e.g., $P_{VBC} < P, P_{PBC} < P$. Here P_{VBC} (P_{PBC}) is the order of the polynomial expansion approximating the velocity (pressure) at Dirichlet boundaries. In simulations with $P_{VBC} < P$ we monitored the term Q_s and found that it was very small, since the energy associated with the high-order modes in sufficiently resolved solution is very small, hence no special treatment in minimizing Q_s was implemented.

2.4. Mesh generation

The computational domains for arterial networks are characterized by significant geometric complexity. The partitioning of Ω into patches is performed manually during the mesh generation stage. First, the 2D inter-patch mesh is generated, which guarantees conforming meshes at the interfaces between adjacent subdomains, and second, the volume mesh is generated within each patch independently. In generating the inter-patch interfaces (2D mesh, Γ) we adopt the following considerations: (i) Γ is a planar polygon. (ii) To minimize the numerical error due to incomplete transfer of boundary modes, the interfaces are created far from the arterial junctions and high curvatures, e.g., far from regions where strong secondary flows (including backflows) are expected to develop. We also employ h-refinement around the inter-patch regions. (iii) The size of each patch (in terms of number of degrees of freedom) allows good strong scaling. The typical number of spectral elements within a patch in our simulations of arterial flow is 20,000–160,000, and the typical number of computer processors dedicated to each patch is 100–4000.

At the beginning of simulations each patch is preprocessed separately on different groups of processors; this stage includes constructing local (to the patch) linear operators (**H** and **L**). At the next stage, the faces of elements containing inlet and outlet boundaries are marked, and then elements with conforming faces and normals pointing in the opposite directions are identified in the adjacent subdomains. We note that no *all-to-all* and *collective* communication involving *all* processors are performed. The data transfer algorithm implemented to impose the IPC conditions is discussed in Appendix A.

3. Results

In this section we evaluate the accuracy and computational efficiency of the multi-patch (2DD) method. First, we present the computational domains employed. Second, we evaluate convergence of the numerical solutions in unsteady flow simulations with 2DD against results obtained with 1DD (single patch domain decomposition). Third, we perform simulations investigating different locations and different sizes of the overlapping region, and we also consider full and reduced resolution in imposing IPC. Fourth, we present results of unsteady flow simulation with the 2DD method in a very complex network of brain arteries. Finally, we conclude this section by presenting the parallel performance of the *NεκTαrG* solver based on the 2DD method.

3.1. Parallel platforms

The computations presented in this paper were performed on (a) CRAY XT5 (Kraken) of National Institute for Computational Sciences (NICS [30]), (b) BlueGene/P (Intrepid) of Argonne National Laboratory [31] and (c) Sun Constellation Linux Cluster (Ranger) of Texas Advanced Computing Center (TACC [32]), ranked as number 3, 8 and 9 on the TOP500 list (http://www.top500.org, November 2009), correspondingly.

3.2. Computational domains

The following computational domains are employed:

(i) A 3D domain of a straight pipe with diameter $D = 2.35$ and length $L = 7$, discretized into 61,126 tetrahedral elements (see Fig. 6). The subdomains Ω_A and Ω_B were created by subdividing the aforementioned geometry into two overlapping parts of a length $L = 4$ each, i.e., the width of the overlapping region ($\Omega_A \cap \Omega_B$) is one length unit (four spectral elements). The number of spectral elements in Ω_A and Ω_B is 36,582 and 36,202, respectively.

(ii) A 3D domain of a converging pipe, subdivided into two overlapping patches as illustrated in Fig. 7. This domain mimics an axisymmetric tapering blood vessel. The first patch (Ω_A, 22,321 tetrahedral elements) is located between the inlet and the interface marked by $S2$, while the second patch (Ω_B, 28,101 tetrahedral elements) is located between interface $S1$ and the outlet.

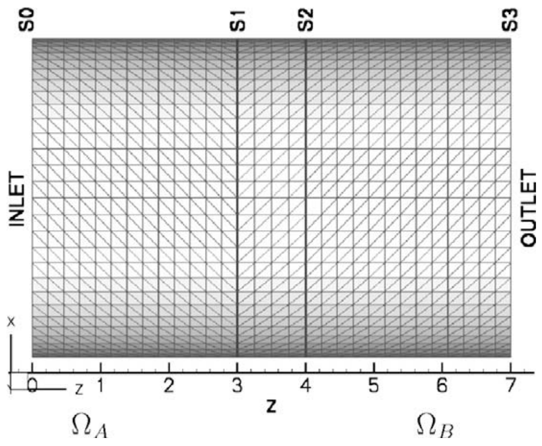

Fig. 6. Domain of a straight pipe subdivided into to overlapping patches: Ω_A (left patch) and Ω_B (right patch). The overlapping region is located between surfaces $S1$ and $S2$.

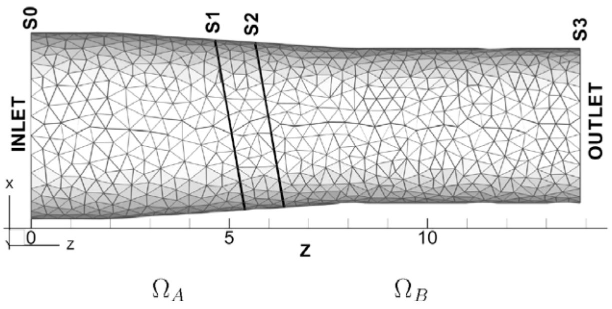

Fig. 7. Domain of a converging pipe subdivided into to overlapping patches: Ω_A (left patch) and Ω_B (right patch). The overlapping region is located between surfaces $S1$ and $S2$.

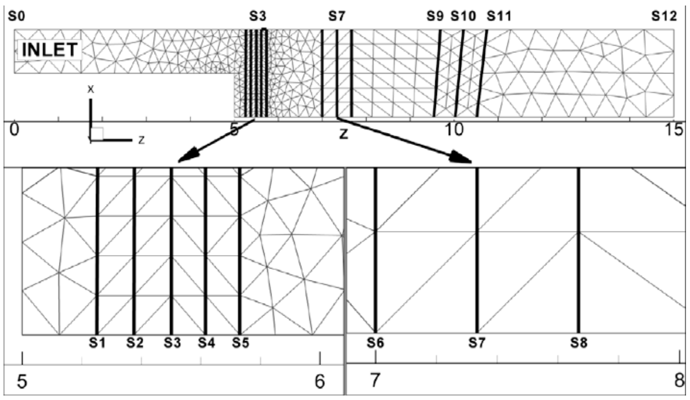

Fig. 8. Domain of a 3D channel with backward-facing step: computational mesh and inter-patch interfaces. S_0 (S_{12}) – inlet (outlet) of domain Ω. $S_j j = 1, \ldots, 11$ – possible inter-patch interfaces. The depth of the channel is 5 length units (in y-direction), the "step" height is 1 length unit. The bottom plots illustrate the surface mesh at $y = 0$ (and also at $y = 5$) around the inter-patch interfaces S_1–S_5 and S_6–S_8.

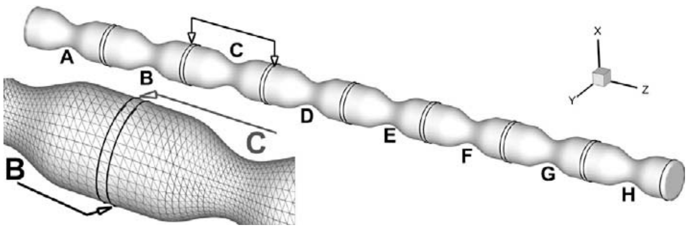

Fig. 9. Decomposition of the large computational domain into several overlapping patches.

(iii) A 3D channel with backward-facing step as illustrated in Fig. 8. We consider several variations of decomposition of the computational domain into patches, with the inter-patch interface located close to the step and further downstream; simulations are also performed with different widths of the overlapping region. Clearly, in this domain the flow behind the step separates and hence we can test the performance of the 2DD approach when the interface crosses the recirculation zone, which is a very rigorous test of the 2DD method.

(iv) Parallel efficiency of the method is evaluated by solving a transient flow problem in a pipe-like domain with multiple constrictions, illustrated in Fig. 9. The computational domain Ω is composed of multiple patches of 17,474 tetrahedral elements each, while the overlapping regions contain 1114 tetrahedral elements and its width is one element.

(v) A 3D domain of brain arteries reconstructed from MRI images, illustrated in Fig. 1. The domain was subdivided into four large overlapping patches: of 75,585 (colored in green), 137,408 (pink), 80,261 (blue) and 131,859 (silver) tetrahedral spectral elements.

To enhance the spatial accuracy in simulating flow in domains (ii), (iv) and (v) the flat faces of elements discretizing the domains surfaces were projected on the curved boundaries by the technique described in [33].

3.3. Accuracy verification

Unsteady flow – straight pipe: In our first numerical simulation we verify the spectral convergence of the error in solutions obtained with the 1DD and 2DD approaches. We perform simulations of unsteady flow in a pipe, driven by a periodically varying pressure gradient. The exact solution for such flow is known as Womersley solution [34]. The main flow characteristics are the Reynolds number $Re = DU/\nu = 350$ and the Womersley number $Ws = 0.5D\sqrt{\omega/\nu} = 4.375$, where D is the inlet diameter, U is the time–space-averaged velocity at the inlet, ω is a characteristic wave number and ν is the kinematic viscosity. The unsteady component was represented by 20 Fourier modes. The data presented here corresponds to the time where the instantaneous Reynolds number is $Re = 1160$. The Womersley velocity profile was imposed as Dirichlet velocity boundary condition at the inlet of Ω (and also Ω_A). Starting from the exact initial solution, time integration was performed over five time units with time step $\Delta t = 0.002$ (2500 time steps) and polynomial order $P = 3,4,5,6$, and $\Delta t = 0.0005$ (10,000 time steps) for $P = 7$; a second-order accurate semi-implicit time-splitting scheme was employed within each patch. In Fig. 10 we plot the convergence of the numerical solution obtained for unsteady flow in a pipe. In simulation with $J_{IPC} = 1$, spectral convergence is obtained in the patch Ω_A only, while in Ω_B the temporal error (of order $O(\Delta t)$) dominates for

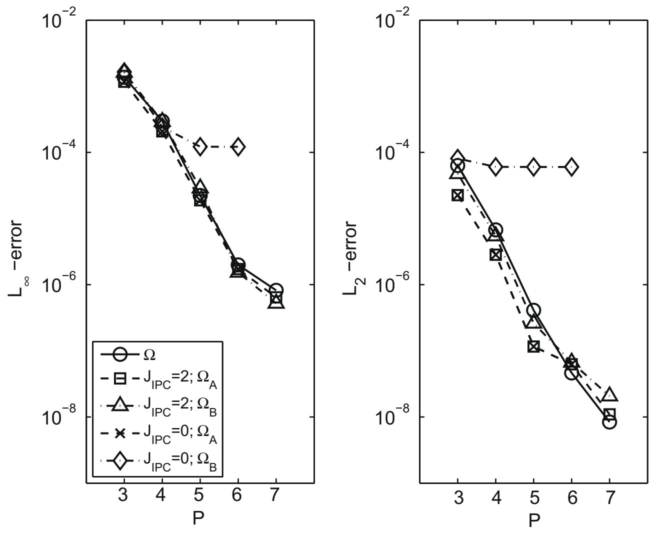

Fig. 10. Unsteady flow simulation: convergence of numerical error. The computational domain Ω is subdivided into two overlapping patches Ω_A and Ω_B. The width of the overlapping region is four elements. Exact solution: Womersley velocity profile represented by 20 unsteady (Fourier) and one steady (parabolic) modes. $Re = 350$, $Ws = 4.375$. Solution performed with $P = 3,4,5,6$ and $\Delta t = 0.002$, and $P = 7$ and $\Delta t = 0.0005$; $P_{VBC} = P_{PBC} = P$. Initial solution: exact velocity profile. Errors measured after five time units.

$P > 4$, due to $O(\Delta t)$ error in imposing IPC for velocity at the inlet. *Spectral convergence* of the error in Ω_B is recovered when the velocity IPC are extrapolated with second-order accuracy (or higher), i.e., $J_{IPC} \geqslant 2$ (7). We note that the error and the rate of its decay in simulations with 2DD are similar to those in simulations with 1DD, i.e., in the standard spectral element method.

Steady/unsteady flow – converging pipe: We start by comparing simulation results obtained in steady flow simulations with and without overlap. In the first scenario Ω_A is interfaced with Ω_B at S_2 only, while in the second the overlapping region is located between S_1 and S_2 as illustrated in Fig. 7. In Fig. 11 we plot the differences in streamwise velocity obtained in the two simulations with respect to 1DD simulation. Although, the differences are relatively small in both cases, the use of overlapping domains is clearly advantageous. However, the main advantage of introducing the overlapping region is the superior stability. Simulations with zero overlap required filtering the high pressure modes ($P_{PBC} = P - 1$, $P = 4$); moreover, simulations performed with $P > 4$ were unstable, which was not observed in simulations with overlap.

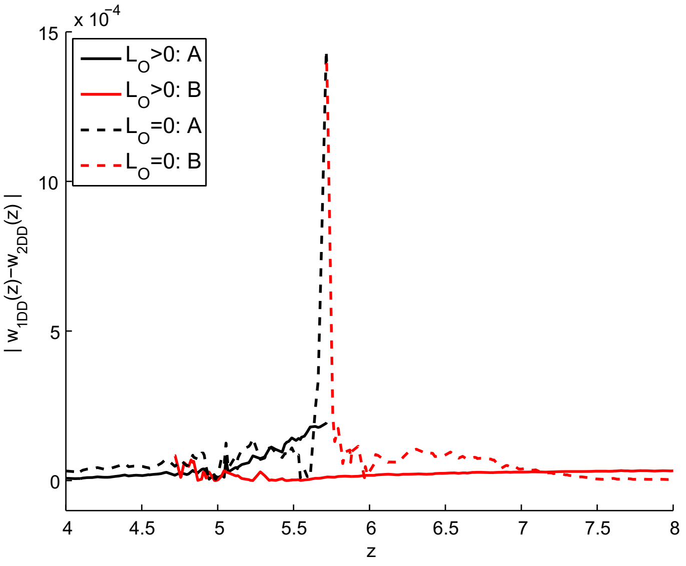

Fig. 11. Steady flow simulation with and without overlap: accuracy. Computational domain Ω is subdivided into two patches Ω_A and Ω_B. Solid line – overlapping region is located between S_1 and S_2; dash line – patches Ω_A and Ω_B are interfacing at S_2 only. Data is extracted along a line $y = 0, x = 1.6$. Solution performed with $\Delta t = 0.002$, and $P_{VBC} = P = 4$, $P_{PBC} = 3$; $Re = 350$, the difference has been measured at $T = 42$.

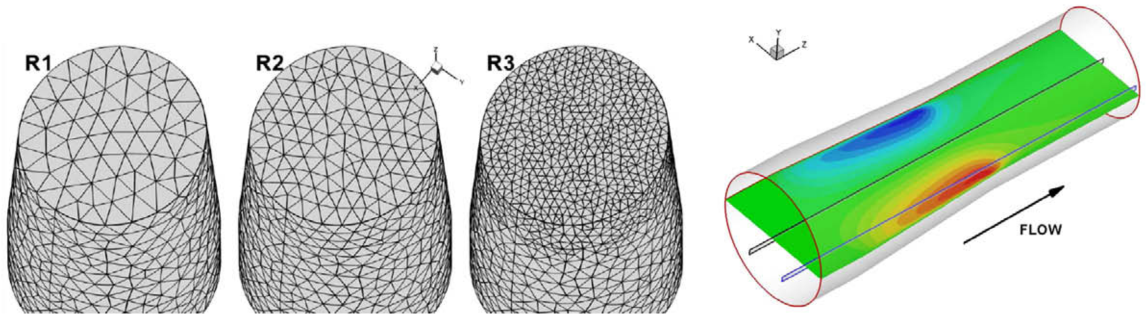

Fig. 12. *h*-Refinement study: surface mesh of the patch Ω_A (see Fig. 7). Plots $R1$, $R2$ and $R3$ depict *h*-refinement at the intepatch surfaces $S2$. The plot on the right illustrates locations of lines $y = 0, x = 0$ and $y = 0, x = -1.6$ where the velocity and pressure are extracted for comparison; color corresponds to velocity component in *x*-direction.

Next, we perform steady and unsteady flow simulations with three levels of *h*-refinement at the inter-patch interfaces $S1$ and $S2$ as illustrated in Fig. 12. At the inlet of Ω (Ω_A) the unsteady (Womerseley) velocity profile was imposed; the corresponding Reynolds and Womerseley numbers are $Re = 350$ and $Ws = 4.375$. The data presented here correspond to the time when the instantaneous Reynolds number was $Re = 1160$. At the outlet of Ω (Ω_B) a Neumann boundary condition for the velocity and zero Dirichlet boundary condition for the pressure were imposed, assuming a fully developed flow. The simulations were performed with the 1DD and 2DD approaches for each level of *h*-refinement. The numerical solutions were obtained with $P_{PBC} = P_{VBC} = P = 5$, $\Delta t = 0.0005$, $J_{IPC} = 3$. In the following, we present results of unsteady flow simulations; results corresponding to steady flow are comparable in terms of deviation between the 1DD and 2DD solutions.

In Figs. 13 and 14 the difference between results obtained with 1DD and 2DD for the three *h*-refinements are presented. The data are extracted along lines $y = 0, x = 0$ and $y = 0, x = -1.6$ as depicted in Fig. 12(right); similar differences between the 1DD and 2DD simulations were observed at other locations. In both simulations we observe about an order of magnitude reduction in the difference between the 1DD and 2DD results due to *h*-refinement. The very good agreement between the flow fields obtained with 1DD and 2DD simulations suggests that the solution obtained with the 2DD method converges to the exact one at a similar rate as the solution obtained with the 1DD method. Due to the $O(\Delta t)$ approximation in treating the pressure IPC, a discontinuity between the pressure fields computed in Ω_A and Ω_B is observed.

Simulations in a domain of 3D channel with backward-facing step: We also performed steady and unsteady flow simulations in the 3D domain presented in Fig. 8 using the 2DD approach investigating various locations of the inter-patch interfaces and different widths of the overlapping region. The reference solution was obtained using a 1DD simulation, performed with the same mesh and spatio-temporal resolution. With this test we want to investigate the effect of performing the multi-patch decomposition with the inter-patch interfaces located in the zone of separated flow. We also investigate the dependence of

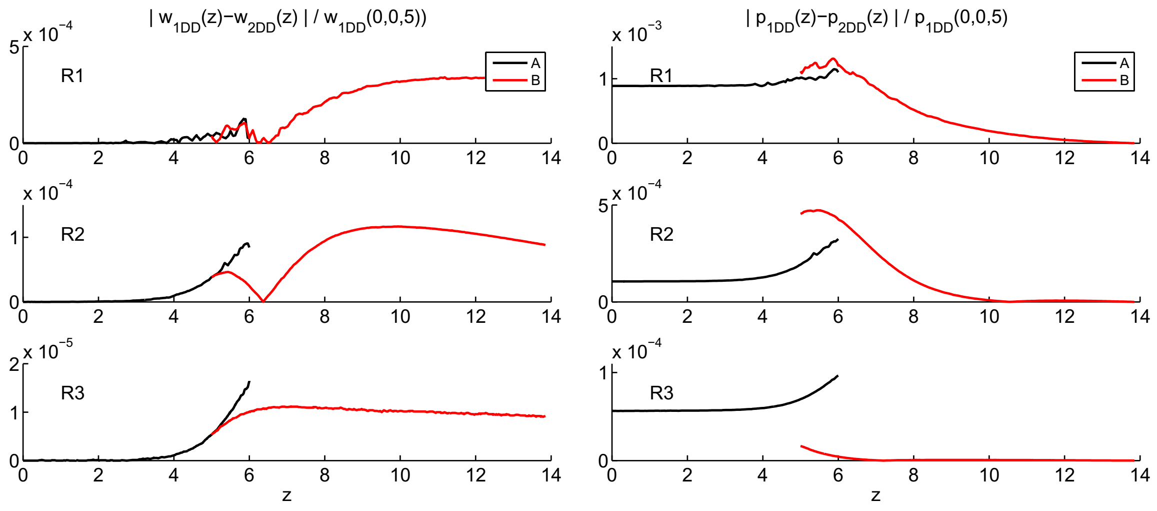

Fig. 13. *Unsteady* flow simulations in domain of Fig. 12 with 1DD and 2DD discretization. Differences in the streamwise velocity (left) and pressure (right) extracted along the centerline of the domain, and computed with 1DD and 2DD. $P = 5$, $P_{VBC} = P_{PBC} = 5$, $\Delta t = 0.0005$. The velocity profile at the inlet of Ω_B was obtained with third-order extrapolation; $Re = DU/\nu = 350$, $Ws = 4.375$; data corresponds to instantaneous $Re = Du/\nu = 1160$, where u is the instantaneous space-averaged velocity at the inlet.

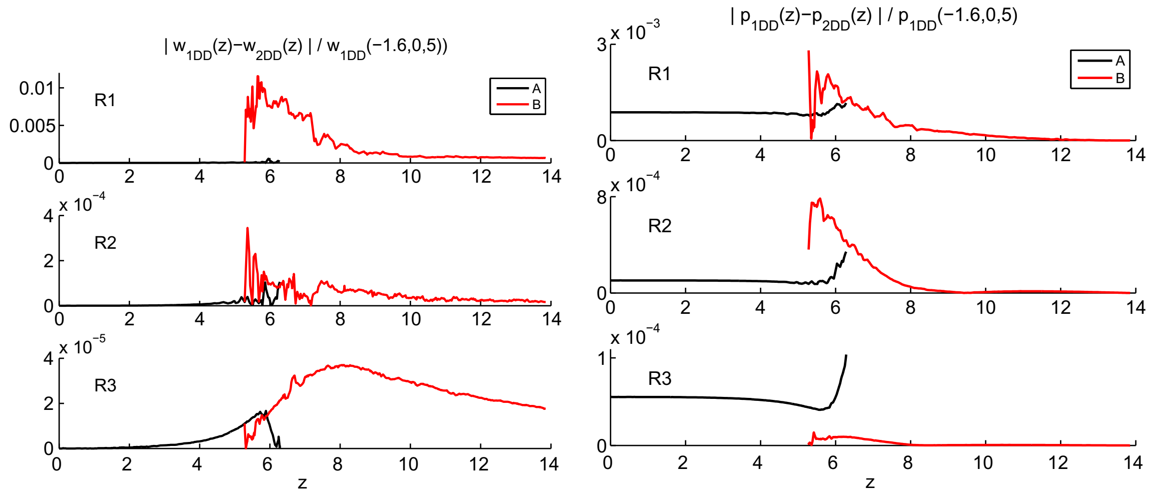

Fig. 14. Same as in Fig. 13 but data are extracted close to the wall.

accuracy on the spectral resolution at the inter-patch that affects directly the transfer of spectral modes and hence the communication cost. Hence, we investigate cases for which $P_{VBC} \leqslant P$ and $P_{PBC} \leqslant P$. (We note here that the spectral element method we use is stable also for representations of the pressure with the same order as the velocity.)

We start from steady flow simulations. In Fig. 15 contours of the streamwise component of velocity vector and pressure are presented for (a,d) simulations with 1DD; (b,e) and (c,f) simulations with 2DD and overlapping patches. Very good agreement between the 1DD and the 2DD simulation results is observed. The pressure field presented in plot Fig. 15(e) deviates only slightly from the pressure presented in plots Fig. 15(d,f) (see pressure contour around $z = 9$). The larger deviation is due to insufficient resolution in imposing IPC; specifically, due to the truncation error corresponding to the choice of $P_{VBC} = P_{PBC} = 3$, $P = 5$. In simulations with interface located between S_1 and S_5 the inter-patch interface is located in the region of relatively high velocity and pressure gradients while in the simulation corresponding to Fig. 15(c,f) the interface is located in the region where velocity and pressure are very smooth and hence the truncation error in IPC is lower. In order to "zoom-in" into the differences between the velocity computed with 1DD and 2DD, we first extract data along the line $x = 1.25, y = 2.5$ (located above the step), and second along the line $x = 0.5, y = 2.5$ (located bellow the step). The results presented in Figs. 16 and 17 correspond to simulations with reduced and full ("consistent") resolution, i.e., $P_{VBC} = P_{PBC} = 3 < P$ and $P_{VBC} = P_{PBC} = P$.

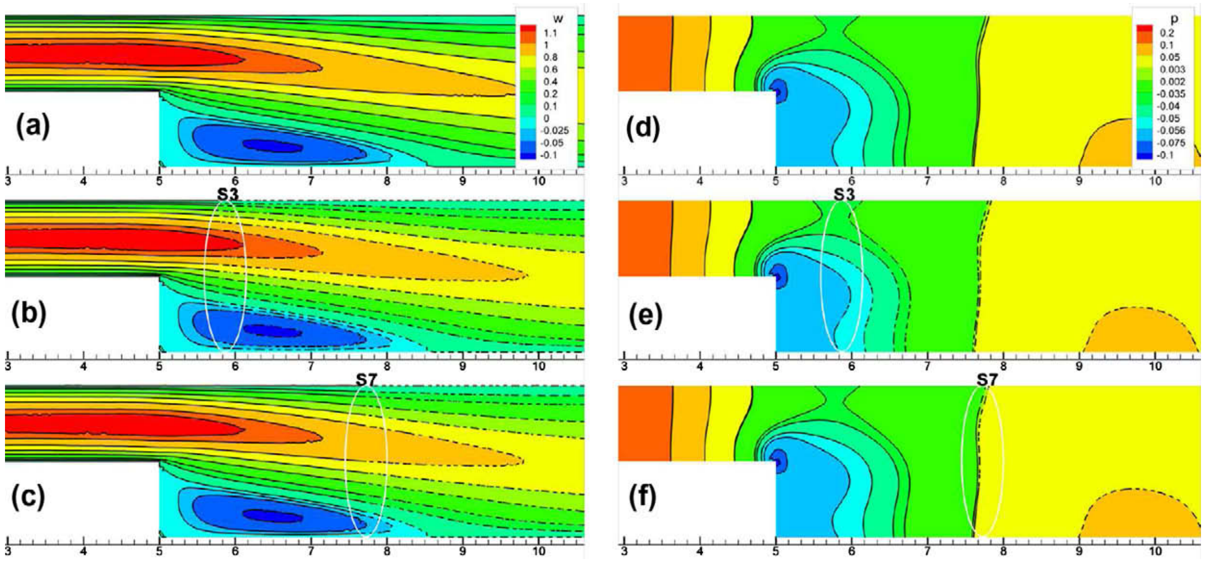

Fig. 15. *Steady* flow simulation in 3D channel with backward facing step with 2DD and overlapping patches. Left – contours of (streamwise) w-velocity components: $w(x,2.5,z)$. Right – pressure contours: $p(x,2.5,z)$. (a,d) 1DD; in (b,e) the overlapping region is located between S_1–S_5; in (c,f) the overlapping region is located between S_6–S_8, respectively, as illustrated in Fig. 8. $P = 5$, $P_{VBC} = P_{PBC} = 3$, $\Delta t = 0.002$, $Re = 72$.

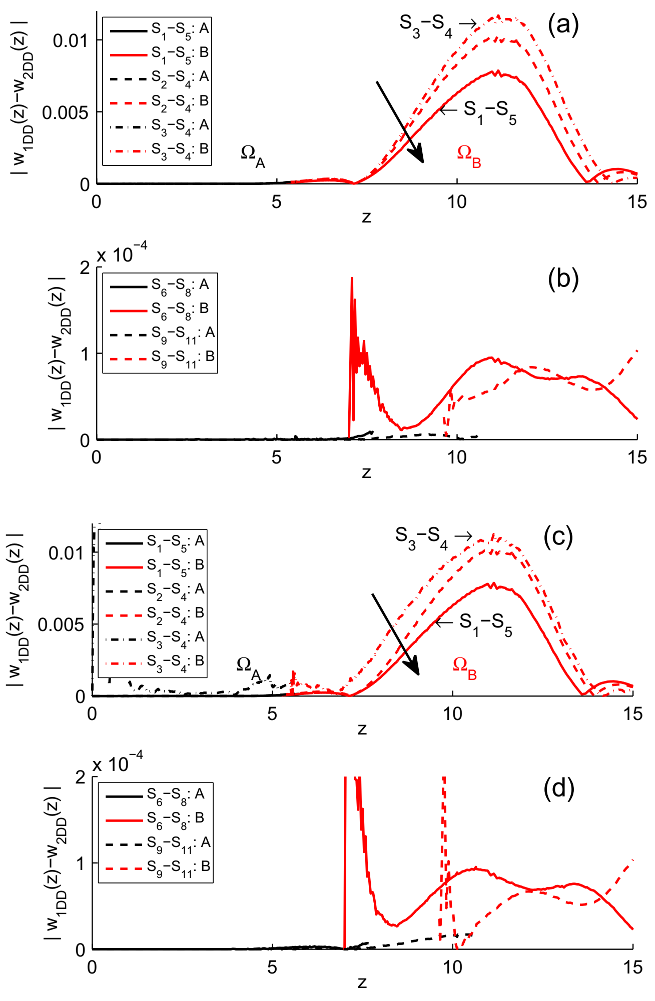

Fig. 16. *Steady* flow simulation in the domain of Fig. 8 with 1DD and 2DD: differences in streamwise w-velocity. The overlapping region is located (a, c) close to the step; (b, d) at the end and behind the recirculation region. The arrow indicates increase in the width of the overlap. The data are extracted along $x = 1.25$, $y = 2.5$. $\Delta t = 0.002$, $Re = 72$, $P = 5$; (a, b) $P_{VBC} = P_{PBC} = 3$, (c, d) $P_{VBC} = P_{PBC} = 5$.

Fig. 16(a, c) correspond to three simulations performed with different sizes of overlapping region: (i) S_1–S_5, (ii) S_2–S_4 and (iii) S_3–S_4. The interface is located very close to the "step" and intersects the recirculation region. Results presented in Fig. 16(b, d) correspond to the cases where the interface is located further downstream, which leads to about one order of magnitude lower difference between the data from the 1DD and 2DD simulations. In plots 16(a, b) we present results obtained with IPC imposed with reduced resolution, while in plots 16(c, d) we present results of simulations with IPC imposed with consistent spatial resolution, i.e., $P_{VBC} = P_{PBC} = P$. Note that for the overlapping one-element-wide region (S_3–S_4) and $P_{VBC} = P_{PBC} = P$, the deviation of the numerical solution between the 1DD and 2DD simulations is larger as expected due to insufficient distance from the interface for the decay of perturbation imposed at the interface boundaries. However, filtering out the high modes (particularly for the pressure) helps to reduce the erroneous oscillations. Similar results were obtained for the pressure field computed with 1DD and 2DD at other locations and widths of the overlap. We note that Neumann velocity boundary conditions are prescribed at the outlet of Ω_A (interface S_2) whereas Dirichlet velocity boundary conditions are prescribed at the inlet of Ω_B (interface S_1). Simulations of flow over the backward-facing step with *non-overlapping* domains with either consistent or reduced spatial resolution in IPC resulted in numerical instabilities.

The data presented in Fig. 17 are extracted across the recirculation region, where the flow changes direction and crosses the interfaces S_1 and S_2 toward the domain Ω_A. In the arterial flow simulations (see next section) the interfaces are created in regions where the velocity direction is uniform, hence, the IPC for velocity will be always imposed following the upwinding criteria at any element of Γ^+.

In Fig. 18 we compare data computed with 1DD and 2DD in *unsteady* flow simulations. In this simulation the inlet velocity was $w = 1 + 0.2\,sin(0.002t)$. The results were obtained from simulations with the largest width of the overlapping region S_1–S_5. The results clearly demonstrate that filtering the pressure field extracted from the downstream domain and imposed as

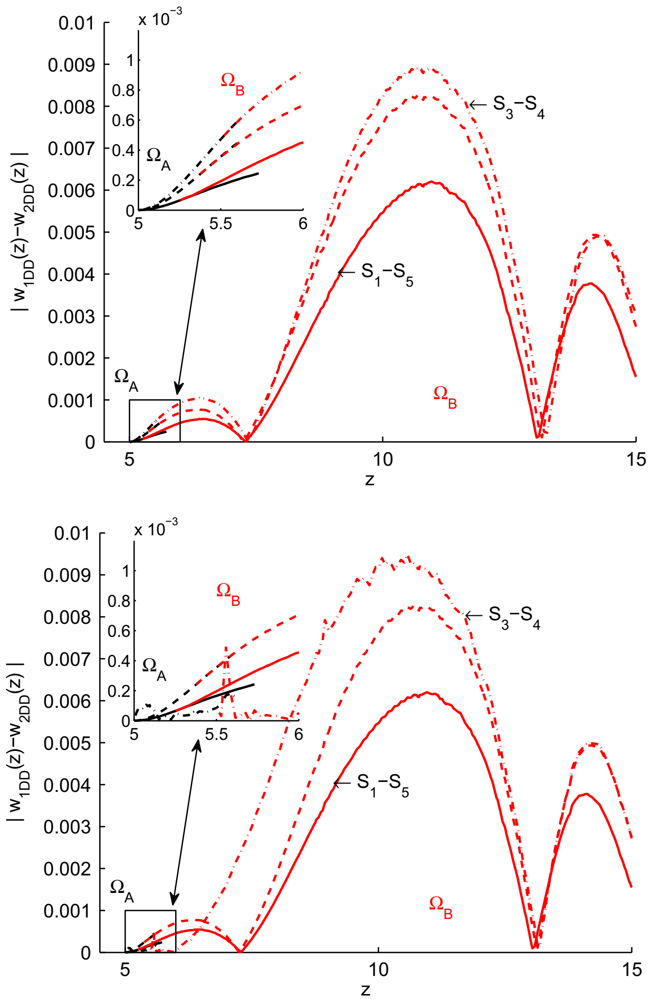

Fig. 17. *Steady* flow simulation in the domain of Fig. 8 with 1DD and 2DD and various width of the overlapping region: difference in *w*-velocity. The overlapping region is located close to the step: S_1–S_5, solid line; S_2–S_4, dash line; S_3–S_4, dash line, dash–dot. The data are extracted along $x = 0.5$, $y = 2.5$. $\Delta t = 0.002$, $Re = 72$, $P = 5$; (top) $P_{VBC} = P_{PBC} = 3$, (bottom) $P_{VBC} = P_{PBC} = 5$.

Dirichlet pressure IPC at the outlet of the upstream domain has a favorable effect on reducing the error in both the velocity and the pressure fields.

A summary on the computational efficiency in simulating the step-flow with 1DD and 2DD is provided in Table 1. We observe that simulations with the 2DD approach are more efficient even for this relatively small problem size. The parallel efficiency of the 2DD approach in solutions of large problems will be discussed in Section 3.4.

Pulsatile flow in the human brain: Next we present results from a large-scale patient-specific simulation in a very complex domain consisting of a network of big arteries in the human brain. The geometry was obtained from MRI images while velocity boundary conditions at the four inlets were based on flowrate measurements by Phase Contrast MRI (Courtesy of T. Anor and J.R. Madsen, Harvard Medical School). This simulation was performed for two cardiac circles. The data presented here were extracted at the time corresponding to the maximum flow rate at the inlet of the left internal carotid artery (ICA). The main parameters of the simulation are: $\Delta t = 1E-3$, $P = 6$, $P_{VBC} = P_{PBC} = 3$ and $J_{IPC} = 1$. At the four inlets Womersley velocity profiles were imposed as a superposition of 1 steady and 10 unsteady modes. In Fig. 19 we present results demonstrating the continuity in the velocity field extracted along the inter-patch regions.

To summarize this section we provide the main findings of the accuracy verification study:

- *p-Refinement study*: Results of simulation of unsteady flow in a straight pipe show that the 2DD method exhibits spectral convergence, similar to the standard 1DD method.
- *Time discretization: High-order* extrapolation of the velocity boundary conditions is required to match the time- and space-discretization errors.

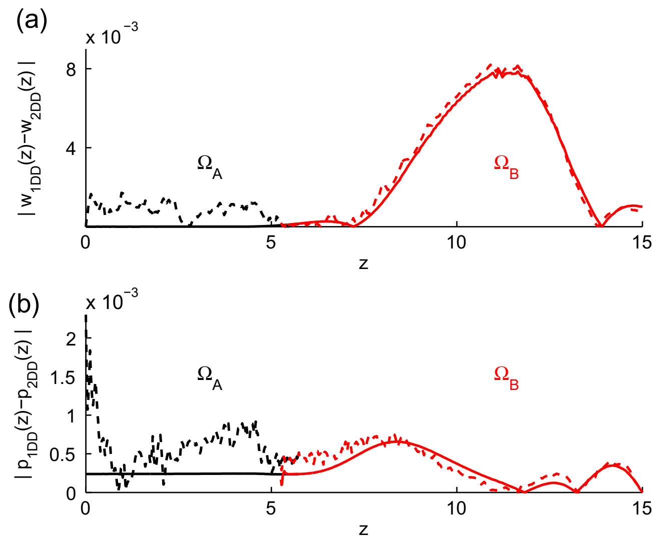

Fig. 18. *Unsteady* flow simulation in the domain of Fig. 8 with 1DD and 2DD with consistent and reduced resolution in imposing IPC for the pressure: difference in w-velocity and pressure. (solid line) $P_{VBC} = P$, $P_{PBC} = P - 2$, (dash line) $P_{VBC} = P_{PBC} = P$. (black) Ω_A, (red) Ω_B. The overlapping region is located close to the step: S_1–S_5. The data are extracted along $x = 1.25$, $y = 2.5$. $\Delta t = 0.002$, $Re = 72$, $P = 5$.

Table 1
Steady flow simulation in a 3D channel with backward facing step: performance on the CRAY XT5 (NICS). *Nel*, number of elements in two patches $Nel(\Omega_A) + Nel(\Omega_B)$. N_{CPU}, number of processes assigned to patches. CPU-time, average time required for one time step.

Overlap	*Nel*	N_{CPU}	CPU-time (s);	CPU-time (s);
			$P = 3$	$P = 5$
1DD	31,518	240	0.21	0.31
S_1–S_5	16,651 + 23,636	96 + 144	0.13	0.26
S_3–S_4	14,633 + 18,930	96 + 144	0.11	0.22
S_6–S_8	24,492 + 9148	176 + 64	0.14	0.25

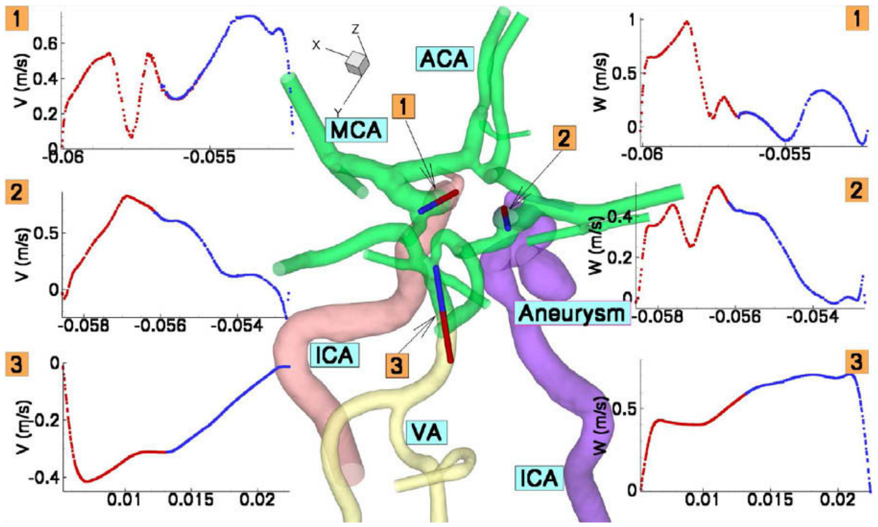

Fig. 19. *Unsteady* flow simulation in the intracranial arterial network with 2DD and overlapping patches: continuity of the velocity field across the interpatch interfaces. The data are extracted along red–blue lines marked by numbers 1, 2 and 3. $\Delta t = 0.001$, $P = 6$; $Re = 394$, $Ws = 3.7$; $P_{VBC} = P_{PBC} = 3$. The total number of spectral elements is $Nel = 425{,}113$, the number of quadrature points in each spectral element $Nq = (P+3)(P+2)^2 = 576$ and number of degrees of freedom $DOF = (Nel)(Nq)4 = 979{,}460{,}352$. The simulation was performed on 3712 cores of the CRAY XT5 (Kraken, NICS). (Courtesy of Yue Yu, Brown University).

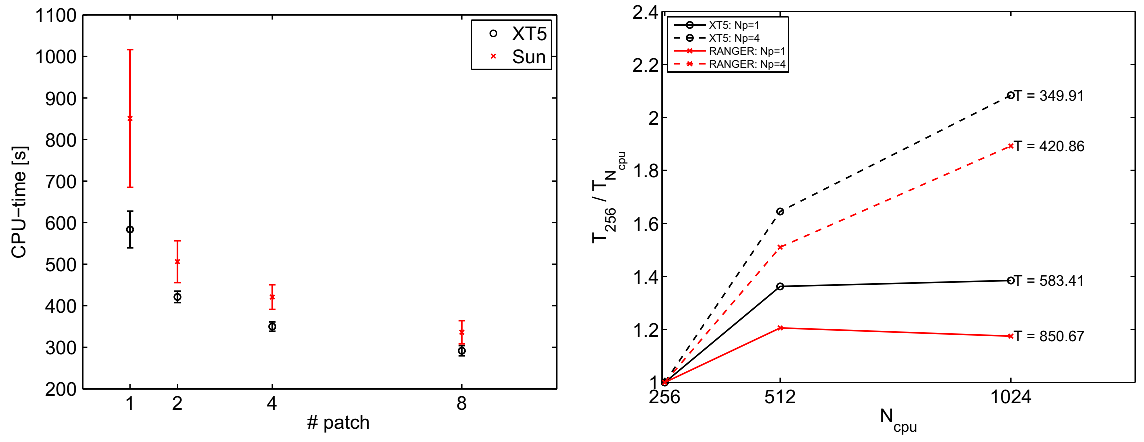

Fig. 20. Flow simulations in a domain of Fig. 9. Simulations with 1, 2, 4 and 8 patches, with 1024, 512, 256 and 128 cores per patch, respectively. Mean CPU-time and standard deviation. CPU-time, time required for 1000 time-steps; *Np*, number of patches. Simulations performed on the CRAY XT5 (Kraken) with 8 cores per node and on the Sun Constellation Linux Cluster (Ranger) with 16 cores per node.

Table 2
Kraken and Ranger: flow simulation in the domain of Fig. 9 using $Np = 1$ and $Np = 4$ patches. The mean CPU-time required for 1000 time steps (excluding preprocessing), results were averaged over 10 simulation for each setup. $P = 4$, $\Delta t = 0.002$, $Re = 470$, preconditioner – LEBP. Simulations were performed using 8 (on Kraken) and 16 (on Ranger) cores per node.

N_{CPU}	CRAY XT5 (Kraken)		Sun (Ranger)	
	$Np = 1$	$Np = 4$	$Np = 1$	$Np = 4$
256	807.7 s	729.1 s	999.0 s	796.3 s
1024	583.4 s	349.9 s	850.7 s	420.9 s

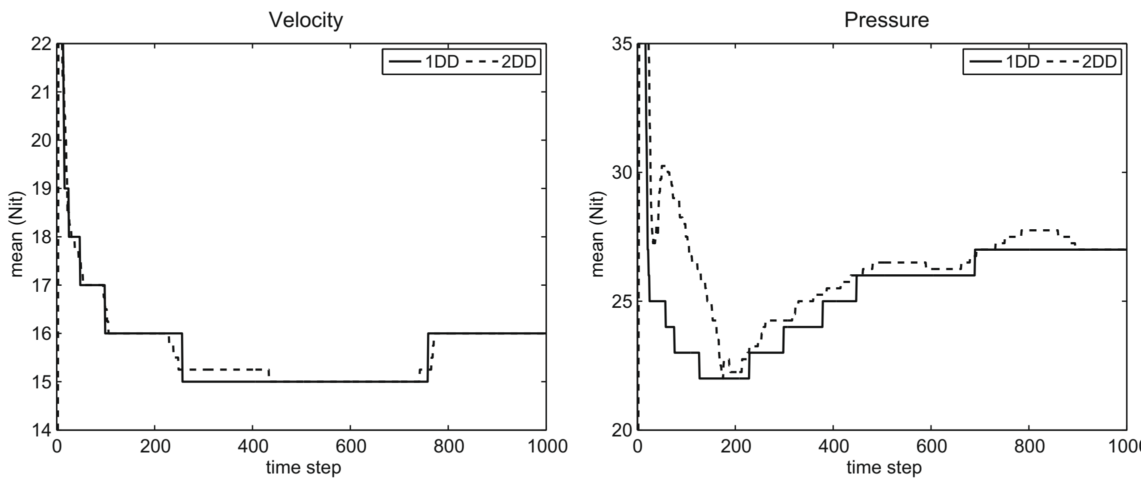

Fig. 21. Flow simulations in a domain of Fig. 9: iteration count. Simulations with one (1DD) and four (2DD) patches. *mean*(*Nit*), mean number of iterations at each time step. Preconditioner – LEBP. $P = 4$, $\Delta t = 0.002$.

- *h-Refinement study*: The difference between the results obtained with the 1DD and 2DD methods can be effectively minimized using local mesh refinement. The *p*- and *h*-refinement study suggest that the results obtained with the 1DD and 2DD methods converge to the exact solution with similar rate (here we assume that the 1DD solution converges to the exact one).
- Use of overlapping patches enhances both the accuracy and the stability of the 2DD method.
- Filtering of high pressure modes reduces the numerical error due to the IPC, and also enhances stability, particularly in the case of very small width of the overlapping region.

- *Optimal* choice of IPC: The performed numerical tests suggest that use of $J_{IPC} = J_e = 2$ (or $J_{IPC} = 3$) is sufficient to minimize the error introduced by the explicit treatment of the IPC. The choice of $P_{VBC} = P$ helps to conserve the mass (and also the high-order moments), while the choice of $P_{VBC} = P - 2$ helps to remove the spurious oscillations introduced by the IPC for the pressure.

Partial results for stability and convergence analysis can be found in prior works on non-iterative solvers and multi-domain decomposition [4–6]. More rigorous theoretical results are required for stability and convergence analysis for various IPCs. Additional research is required to find optimal IPC, particularly for the cases of the reversal flow, such as in the simulations of forward facing step flow.

3.4. Parallel efficiency

In this section we present strong and weak scaling results obtained in simulations of transient flow. First, we compare the strong scaling obtained with the 1DD and 2DD approaches. Second, we focus on the weak scaling obtained in simulations with the 2DD approach. We monitor the CPU-time required to integrate the solution over 1000 time steps; these simulations were performed 10 times on each computer to take into account the system "noise".

The first test employs the computational domain Ω of Fig. 9, subdivided into 2, 4 and 8 overlapping patches. The number of spectral elements in a single-patch domain was $Nel_1 =$ 130,880, while in the cases of 2, 4 and 8 patches the total number of elements was $Nel_2 =$ 131,994, $Nel_4 =$ 134,222 and $Nel_8 =$ 138,678, respectively. The increasing number of elements is due to overlapping regions with each overlapping region having 1114 elements shared by two neighbor patches. The computations were performed on 1024 cores of Ranger and Kraken with $\Delta t = 0.002$, $P = 4$ and initial condition $\mathbf{v} = 0$. The mean CPU-time and the standard deviations are presented in Fig. 20(left). Note, that despite the increasing problem size (due to the overlap), the computational cost is decreasing due to lower volume of the communication and lower conditional number corresponding to local (to each patch) linear operators. To appreciate the effect of 2DD on the *strong* scaling we performed simulations with one and four patches ($Np = 1$ and $Np = 4$) and relatively low-order polynomial approximation $P = 4$. These simulations were performed on Kraken and Ranger using 256 and 1024 cores equally subdivided between four patches. The choice of relatively low *P*, high core count and use of computers with relatively fast CPUs usually results in poor strong scalability. The results are summarized in Fig. 20(right) and Table 2. By comparing results for $Np = 1$ and $Np = 4$ with respect to the number of processes we can observe considerably better strong scaling in simulations with 2DD. In Fig. 21 we plot the iteration count (*Nit*) for the Helmholtz solver for velocity and the Poisson solver for pressure in solution with 1DD and 2DD using four patches. The mean *Nit* is computed at each time step as $mean(Nit) = 1/Np\sum_{i=1}^{Np} Nit(\Omega_i)$. The higher *Nit* at the beginning of the simulation is due to the IPC, specifically due to discontinuity in the pressure field between the patches. After some transient period, the *Nit* in 1DD and 2DD is about the same, despite the smaller size of a problem defined within each patch. This is not surprising and is related to excellent *h*- and *p*-scaling of the LEBP. Based on the data presented in Table 2 and Figs. 20 and 21, we can conclude that the lower CPU-time requirements by the 2DD method are due to decreased volume of communication, rather due to the conditioning of the local to each patch linear operators.

In simulation with the diagonal preconditioner the condition number κ of the Laplacian operator is expected to grow linearly with the number of elements *Nel* [20], hence $Nit \propto \sqrt{Nel}$. In Fig. 22 we plot *Nit* required by the Poisson and Helmholtz solvers in simulations with 1DD and 2DD. The iteration count for the pressure solver with 2DD decreased approximately by a factor of $\sqrt{Nel(\Omega)/Nel(\Omega_i)}$ as expected. The number of iterations required by the Helmholz solver did not change signifi-

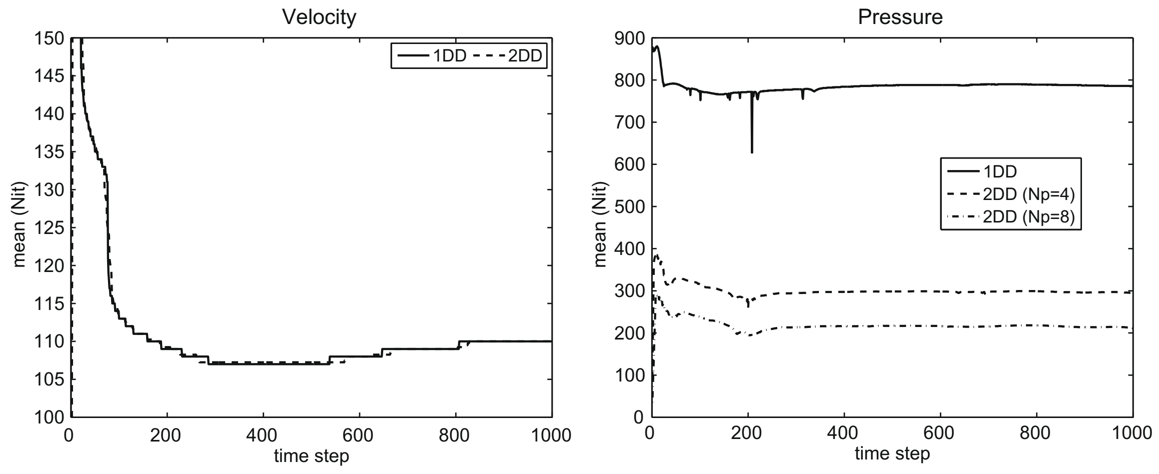

Fig. 22. Same as in Fig. 21, but with diagonal preconditioner and also *mean*(*Nit*) for the case of $Np = 8$.

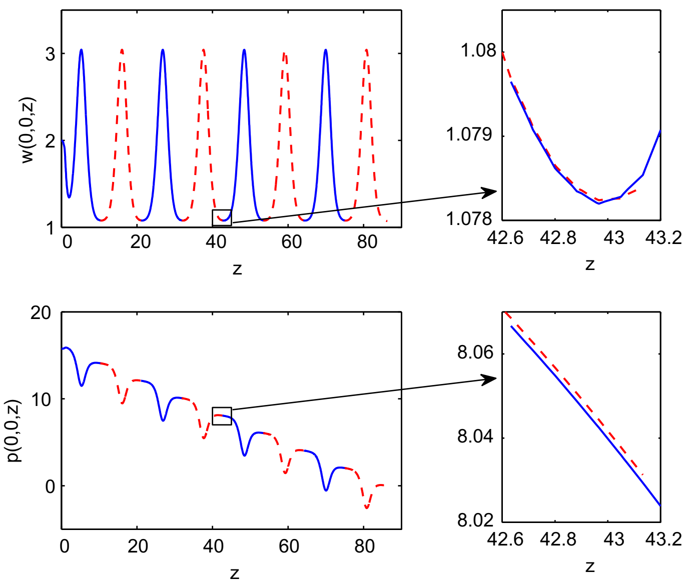

Fig. 23. Flow simulations in the domain of Fig. 9 with 8 patches: continuity of the w-velocity (streamwise) and pressure between the patches. Results have been obtained after 1000 timesteps. $Re = 350$, $\Delta t = 0.002$, $P = 4$.

cantly. Due to relatively small Δt and P the Helmholtz operator is Mass-matrix dominant and the low sensitivity of Nit required by velocity solver to Nel reflects favorable scaling of the diagonally preconditioned operator.

The relatively small gain in the CPU-time in simulations with 8 patches with respect to the simulations with the four patches observed in the first test suggests that further decomposition of the domain into more patches will not be effective. The saturation in minimization of the solver time by increasing number of patches raises a question on the optimal patch size. It has been shown in [3] (Section 3.1) that the strong scaling of the 2DD solver is bounded by the strong scaling of a solver applied within each patch. Parallel efficiency of a solver applied within a patch typically grows with the reduced patch's size and the number of cores assigned to the patch. Hence, additional subdivision of already small enough patches may not lead to better scaling. The optimal patch size is closely related to the number of cores employed for solution of a problem within the patch, and the optimal combination is the one which results in very good strong scaling. We note that for different problems and also different computer architectures the optimal patch size may vary.

In our second test we verify the good weak and strong scaling on up to 32,768 cores of the BlueGene/P and CRAY XT5 computers. Simulations were performed with 3, 8 and 16 patches with about 17 K spectral elements in each patch. The convergence of velocity and pressure at the inter-patch interfaces after 1000 timesteps is shown in Fig. 23.

The results of weak scaling tests are summarized in Table 3 showing very good weak scaling. The performance of the 2DD approach was also verified in simulations with 40 patches on 96,000 cores of the CRAY XT5. The number of degrees of freedom in this simulation was about 8.21B ($P = 12$). In this simulation convergence of the iterative solver was accelerated by an

Table 3
BlueGene/P and CRAY XT5: flow simulation in the domain of Fig. 9 using 3, 8 and 16 patches. Np, number of patches. CPU-time, time required for 1000 time steps (excluding preprocessing). $P = 10$, $\Delta t = 0.0005$, $Re = 470$, preconditioner – LEBP. Case A1(2,3) corresponds to simulations with 1024 cores/patch, case B1(2,3) corresponds to simulations with 2048 cores/patch. On BlueGene/P simulations have been performed using four cores per node (mode *vn*). On CRAY XT5 simulations have been performed using 8 cores per node.

Case	*Np* (DOF)	# of Cores/patch	Total # of cores	CPU-time (s)	Weak (strong) Parallel efficiency
BlueGene/P					
A1	3 (0.384B)	1024	3072	996.98	100% (100%)
A2	8 (1.038B)	1024	8192	1025.33	97.2% (100%)
A3	16 (2.085B)	1024	16,384	1048.75	95.1% (100%)
B1	3 (0.384B)	2048	6144	650.67	100% (76.6%)
B2	8 (1.038B)	2048	16,384	685.23	95% (74.8%)
B3	16 (2.085B)	2048	32,768	703.4	92% (74.5%)
CRAY XT5					
B1	3 (0.384B)	2048	6144	462.3	100%
B2	8 (1.038B)	2048	16,384	477.2	96.9%
B3	16 (2.085B)	2048	32,768	505.1	91.5%

accurate prediction of the initial state [21]. It was observed that the average CPU-time per time step required to solve such a large problem was about 0.25 s. Moreover, considering the very good weak scaling, there is the potential for solving truly ultrascale flow problems (e.g., blood flow in the *entire* arterial tree) within similar wall-clock time per time step.

To summarize this section we provide the main findings of the scalability study:

- *Strong scaling*: The 2DD method offers dual path to strong scaling of a solver. Such scaling can be improved by performing interpatch optimization and also by decomposing a fixed size computational domain into more patches.
- *Weak scaling*: The method exhibits very good weak scaling. This result is expected as solvers applied within each patch are executed on non-overlapping groups of processors which involve no communication. The point-to-point communication between different patches is performed only three times per time step (to pass the velocity, pressure and velocity flux values) hence, its impact on the overall scaling is negligible.
- *Optimal patch size*: Based on our recent findings and also on the previous study [3] we determine the optimal patch size as a domain size for which good strong scaling can be achieved. Further decomposition of the domain into even smaller patches may not improve the parallel efficiency. We note that decomposition into loosely coupled patches introduces numerical error at the interfaces. Hence, ideally (in terms of accuracy) the optimal number of patches per domain is strictly one. However, due to poor scalability of a (semi-)implicit *tightly* coupled solvers on tens of thousands computer processors, decomposing a computational domain into loosely coupled patches provides an efficient way of balancing the computational efficiency with accuracy. We also note that the optimal patch size depends on the architecture of the computer on which the solver is executed as well as on the spatial resolution employed (polynomial order and number of spectral elements). For example, due to network differences the optimal patch size in simulations on BlueGene can be greater than on the Sun Constellation Linux Cluster (Ranger, TACC). Use of high-order polynomial approximation (large P) leads to better scalability than solving the same problem with lower resolution (in terms of P). The later occurs due to the maximizing the ratio of the volume of computation to the volume of communication within each patch, hence, with very high P good parallel efficiency can be achieved on within every a patch.
- Multi-patch decomposition offers additional advantages (not discussed in this paper) for data post-processing. Post-processing a data file produced in smaller domain (patch) is computationally favorable than post-processing extremely large data sets.

4. Summary and discussion

Current CFD or other general computational mechanics codes that employ implicit or semi-implicit time-stepping algorithms are not scalable to thousands of processors/cores available in the emerging petaflop computers. The main bottleneck is the lack of scalability of *effective* linear iterative solvers for problems involving matrices with size one billion or greater corresponding to extremely large condition number. In this paper, a new scalable approach is presented specific to numerical simulation of incompressible flows. The target application is simulation of blood flow in the human arterial tree but the same method can be applied to other complex networks or extended to other fields. The proposed approach provides a dual path to scalability by affecting favorably both the strong and weak scalability.

This two-level domain decomposition method (2DD) has been implemented in conjunction with the spectral/*hp*-element for spatial discretization and a high-order semi-implicit time-splitting scheme. Results of steady and unsteady 3D flow simulations in simple and complex geometry domains show excellent agreement between the standard single-patch approach (1DD) and the multi-patch approach (2DD). Flow simulations on up to 96 K cores showed both very good time per time step but also favorable scaling; such simulations are not feasible with a monolithic scheme.

The new 2DD method is suitable for the hybrid architecture of the emerging multi-core petaflop systems with 100-way or more cores, in that a direct map of each patch to a “fat” node can be performed. This, in turn, provides great flexibility in balancing accuracy and parallel efficiency. Moreover, the 2DD approach leads to less intensive communication among nodes, it considerably reduces the number of iterations of the linear solvers, and translates the problem of overall scalability to optimizing the scalability of individual patches – a much simpler task indeed! From the implementation standpoint, we note that the 2DD approach is an almost “non-intrusive” method as it requires minor modifications in the *existing* codes. Specifically, the most significant modification in our code was replacing the *main* function by a function with a different name and adding one more variable to the arguments of this function – the pointer to sub-communicator, provided by the Multilevel Communication Interface (see Appendix A), over which the patch local solution is computed. The inter-patch boundary conditions (IPC) developed in this study are rather simple in order to avoid unnecessary communication costs. Alternative IPC must be considered, particularly for Poisson solver for the pressure. To this end, more research is required to develop robust and scalable interface conditions that take advantage of the advances in bandwidth in the next generations of parallel computers.

Acknowledgments

This work was supported by the NSF under grants OCI-0636336 and NSF OCI-0904288. Supercomputer resources were provided by the NSF Large Allocations Resource. The authors thank Yue Yu and Hyoungsu Baek from Brown University for performing the numerical simulation in the large network of brain arteries with aneurysm and providing graphics.

We also want to thank Lonnie D. Crosby of NICS and Dr. John Cazes from TACC for their support in optimizing our software. We also thank Bruce Loftis for providing us with additional computational time. This research also used resources of the ALCF at ANL, which is supported by the Office of Science of the U.S. DoE under contract DE-AC02-06CH11357.

Appendix A. Multilayer Communicating Interface (MCI)

A parallel numerical simulation performed with 2DD involves two types of communication: (a) *local* or intra-patch communications, where processes from the same group communicate, and (b) *global* or inter-patch communications, where processes from different groups communicate. The inter-patch communication developed for 2DD can be efficiently performed within a single supercomputer as well as across geographically distributed computers. A schematic representation of MCI semi-hierarchical decomposition of the global communicator is illustrated in Figs. 24 and 25. In the following we provide a detailed description of different layers of communicators.

The first layer (L1) of the MCI is the default communicator *MPI_COMM_WORLD*. The second layer (L2) is derived from L1 using topology-aware splitting, processes executing on the same computer are grouped in a non-overlapping manner (groups S_1, S_2 and S_3 in Fig. 24). The splitting of L1 is done dynamically, in simulations on distributed computers MPICH-g2 and MPIg middlewares provide a functionality to easily split processes according to topology. The third layer (L3) is derived from L2 using task oriented decomposition; each L3 sub-group is dedicated to parallel solution of one tightly coupled problem (groups T_1, T_2, T_3 and T_4 in Fig. 24). The task-oriented decomposition is controlled by information provided as an input to *NεκTαrG*. When the program is executed on a single supercomputer the L3 communicators are derived directly from L1.

Solving the tightly coupled problem may involve specific tasks typically executed by a subset of processes, hence a fourth layer (L4) of MCI is created. In the arterial flow simulation such tasks involve treatment of boundary conditions at the arterial inlets and outlets. In *NεκTαrG* different types of inlets and outlets are represented by three objects: (1) class TerminalBoundary, (2) class OvrLapBoundary and (3) class MergingBoundary. Each of the three objects handles communication and provides the required functionality for inlets and outlets, i.e., computing flowrates, mean pressure, mapping between patches, exchanging data for IPC, etc., for additional information we refer to [21]. The numerical integration over multiple inlets/outlets is performed concurrently and the blocking *MPI_Allreduce* is executed simultaneously over different L4 sub-communicators. The L4 sub-communicators implemented in *NεκTαrG* are subdivided into three groups (as illustrated in Fig. 5): (i) *T_inlet* (*T_outlet*): sub-communicator consists of processes assigned to partitions with elements facing inlet (outlet) of the global domain Ω. (ii) *M_inlet* (*M_outlet*): sub-communicator consists of processes assigned to partitions with elements facing inlet (outlet) of the patch that interfaces with the adjacent patch. (iii) *O_inlet* (*O_outlet*): sub-communicator consists of processes assigned to partitions with elements which faces are conforming with the faces of elements of the outlet (inlet) of the adjacent patch. The point-to-point communication between the ROOTs of L4 is performed over MPI_COMM_WORLD. Additional sub-communicator, connecting the ROOTs of all L4 sub-communicators derived from the same L3 communicator and also the ROOT of corresponding L3, is required to transfer flowrates and mean pressures computed locally to the ROOT of L3, which outputs these data to a file or transfers it to the dedicated IO processor. The mapping as well as number of processes in the L4 communicators is not known a priori, and it is done during the preprocessing stage. The zero overlap can be considered as a limiting case of the overlapping domains, and from the standpoint of parallel algorithm the two cases can be treated in exactly the same way.

Fig. 25(top) depicts an example of a solver executing four tasks (four L3 sub-communicators). Only one process is assigned to the first task, while 14 processes are assigned to tasks 2, 3 and 4. Task 1 is dedicated to the 1D arterial flow solver and is employed for co-processing of the intermediate results, e.g., IO, system calls, etc., while tasks 2–4 solve the 3D flow equations in arterial domains schematically represented at the right of Fig. 25(top). In the illustration of Fig. 25(top) each patch has one inlet and different number of outlets, and, accordingly, different number of L4 sub-communicators. In simulations on low number of processes the L4 communicators may overlap, however, this is not a problem from the algorithmic point of view.

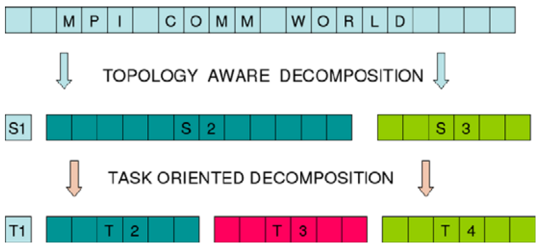

Fig. 24. Multilayer Communicating Interface: high level communicator splitting. *MPI_COMM_WORLD* is subdivided according to the computer topology to form three non-overlapping process sub-group (S_j). The S_2 group is subdivided using task-oriented splitting and four non-overlapping T_j groups are created. Cells represent processes.

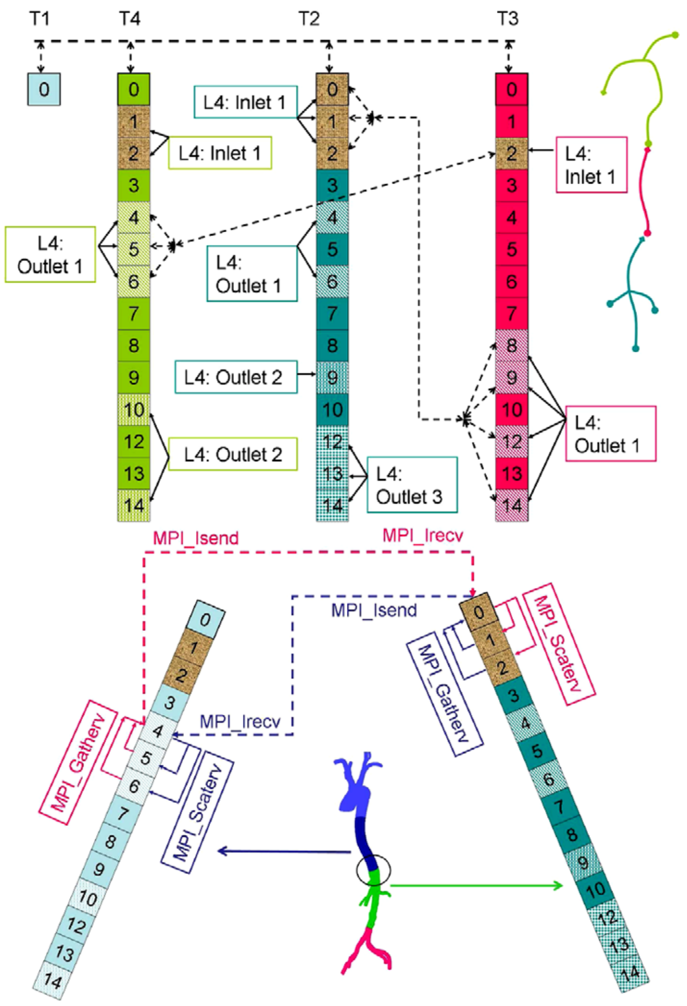

Fig. 25. Multilayer Communicating Interface. (top) Low level communicator splitting. Here four third-level process sub-groups (T_j) are shown. The low level communicator splitting is performed within each of the T_j sub-group. The inlet and outlet communicators (L4) are created. (bottom) Three-step algorithm for inter-patch communication in an "aorta" domain (Fig. 1).

The processes of L5 sub-communicator include ROOTs of L3 sub-communicators, their main purpose is to synchronize computation between the tasks.

Solving flow equations with 2DD requires imposing interface boundary conditions for the velocity and pressure as explained in the main paper, which necessitates data exchange across the patch interfaces. The inter-patch communication is implemented in three steps, as depicted in Fig. 25(bottom) and explained next. In the illustration of Fig. 25(bottom) we consider non-overlapping patches; the communication pattern in the case of overlapping patches is similar. Processes 4–6 compose L4: M_outlet communicator of the blue patch, while processes 0–2 compose L4: M_inlet of the green patch. On the first step, data from the interface partitions is gathered to the ROOT of L4. On the second step, point-to-point communication between the ROOTs of corresponding L4 occurs in passing data between the adjacent patches. In the third step, data received by the ROOT of L4 is scattered (or broadcasted) to the interface partitions only.

Exchanging data through the ROOTs of L4 sub-communicators is an optimization made for simulations on distributed computers to minimize the traffic over the slower network. The emerging petaflop supercomputers have features of a heterogeneous network in terms of communication speed. The differences in latency and bandwidth in communication between different partitions of heterogeneous computers must be accounted for in the design of the MCI; also different tasks executed by a program might be mapped on different partitions, e.g., MPI, IO or floating-point intensive operations. The hierarchical multi-level parallelism allows execution of blocking communication over different groups of processes. Iterative solution of linear systems performed by PCG requires three *MPI_Allreduce* calls at each iteration. The high penalty of communication cost on thousands of cores affects the parallel efficiency adversely. For example, our tests indicate that the *MPI_Allreduce*-time on 11,152 cores of Ranger is about five time slower than on 4096 cores.

References

[1] International assessment of research and development in simulation-based engineering and science, WTEC Panel Report, 2009. Available from: <www.wtec.org>.
[2] D. Keyes, P. Colella, T.H. Dunning Jr., W.D. Gropp (Eds.), A Science-based Case for Large-scale Simulation, vols. 1&2, Office of Science, U.S. Department of Energy (DOE), Washington, DC, 2003.
[3] L. Grinberg, G.E. Karniadakis, A scalable domain decomposition method for ultra-parallel arterial flow simulations, Commun. Comput. Phys. 4 (5) (2008) 1151–1169.
[4] H. Chen, R.D. Lazarov, Domain splitting algorithm for mixed finite element approximations to parabolic problems, East–West J. Numer. Math. 4 (1995) 121–135.
[5] F.K. Hebeker, Yu.A. Kuznetsov, Unsteady convection and convection–diffusion problems via direct overlapping domain decomposition methods, Numer. Meth. Partial Differential Equations 14 (3) (1997) 387–406.
[6] F.K. Hebeker, A domain splitting method for heat conduction problems in composite materials, Math. Model. Numer. Anal. 34 (1) (2000) 47–62.
[7] Z.N. Wu, H. Zou, Grid overlapping for implicit parallel computations of compressible flows, J. Comput. Phys. 157 (2000) 243.
[8] Z.N. Wu, H. Zou, Efficient parallel method for implicit upwind schemes approximating the Euler equations in gas dynamics, J. Comput. Phys. 187 (2) (2003) 683–715.
[9] X. Tai, T. Johansen, H. Dahle, M.A. Espedal, Characteristic domain splitting method, in: Glowinski et al. (Eds.), Domain Decomposition Methods in Science and Engineering, Proceedings of 8th International Conference on Domain Decomposition Methods, Wiley, New York, 1997, pp. 317–323.
[10] H. Wang, J. Liu, M.S. Espedal, R.E. Ewing, A characteristic nonoverlapping domain decomposition method for multidimensional convection-diffusion equations, Numer. Methods Partial Differential Equations 21 (1) (2004) 89–103.
[11] C. Bernardi, Y. Maday, A.A. Patera, New nonconforming approach to domain decomposition: the mortar element method, in: H. Brezis, J.L. Lions (Eds.), Nonlinear PDEs and their Applications, Collegede France Seminar 1994: vol XI, Pitman Research Notes in Math. Series 299, Longman, London.
[12] X. Cai, M. Dryja, M. Sarkis, Overlapping nonmatching grid mortar element methods for elliptic problems, SIAM J. Numer. Anal. 36 (1999) 581–606.
[13] P.E. Bjørstad, O.B. Widlund, To overlap or not to overlap: a note on a domain decomposition method for elliptic problems, SIAM J. Sci. Statist. Comput. 10 (5) (1989) 1053–1061.
[14] M. Dryja, O.B. Widlund, Domain decomposition algorithms with small overlap, SIAM J. Sci. Statist. Comput. 15 (3) (1994) 604–620.
[15] H.H. Kim, O.B. Widlund, Two-level Schwarz algoriths, using overlapping subdomains, for mortar finite element methods, SIAM J. Numer. Anal. 44 (4) (2006) 1514–1534.
[16] C.R. Dohrmann, A. Klawonn, Olof B. Widlund, Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions, SIAM J. Numer. Anal. 46 (4) (2008) 2153–2168.
[17] L.F. Pavarino, E. Zampieri, Overlapping Schwarz and spectral element methods for linear elasticity and elastic waves, J. Sci. Comput. 27 (1-3) (2006) 51–73.
[18] M. Paraschivoiu, X.C. Cai, M. Sarkis, D.P. Young, D.E. Keyes, Multi-domain multi-model formulation for compressible flows: conservative interface coupling and parallel implicit solvers for 3D unstructured meshes, AIAA Paper 99-0784, American Institute of Aeronautics and Astronautics, Reston, VA, 1999.
[19] J. Li, M. Melenk, B. Wohlmuth, J. Zou, Optimal convergence of higher order finite element methods for elliptic interface problems, Appl. Numer. Math. 60 (2010) 19–37.
[20] G.E. Karniadakis, S.J. Sherwin, Spectral/hp Element Methods for CFD, second ed., Oxford University Press, Oxford, 2005. 650 pp.
[21] L. Grinberg, Topics in ultrascale scientific computing with application in biomedical modeling, Ph.D. Thesis, Brown University, 2009.
[22] S.J. Sherwin, M. Casarin, Low-energy basis preconditioning for elliptic substructured solvers based on unstructured spectral/hp element discretization, J. Comput. Phys. 171 (1) (2001) 394–417.
[23] L. Grinberg, D. Pekurovsky, S.J. Sherwin, G.E. Karniadakis, Parallel performance of a low energy basis preconditioner for spectral/hp elements, Parallel Comput. (2009), doi:10.1016/j.parco.2008.12.002.
[24] G.E. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.
[25] B. Cockburn, G.E. Karniadakis, C.W. Shu, Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000.
[26] T.J.R. Hughes, G. Scovazzi, P.B. Bochev, A. Buffa, A multiscale discontinuous Galerkin method with the computational structure of a continuous Galerkin method, Comput. Methods Appl. Mech. Eng. 195 (2006) 2761–2787.
[27] M. Gander, C. Japhet, Y. Maday, F. Nataf, A new cement to glue nonconforming grids with robin interface conditions: the finite element case, in: Domain Decomposition Methods in Science and Engineering Series: Lecture Notes in Computational Science and Engineering 40 (4) (2006) 259–266.
[28] M. Zhang, C.W. Shu, An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations, Math. Models Meth. Appl. Sci. 13 (3) (2003) 395–413.
[29] R.M. Kirby, G.E. Karniadakis, Selecting the numerical flux in discontinuous Galerkin methods for diffusion problems, J. Sci. Comput. 22–23 (3) (2005) 385–411.
[30] National Institute for Computational Sciences (NICS). Available from: <http://www.nics.tennessee.edu>.
[31] Available from: <http://www.alcf.anl.gov>.
[32] Available from: <http://services.tacc.utexas.edu/index.php/ranger-user-guide>.
[33] P. Volino, N. Magnenat-Thalmann, The SPHERIGON: A Simple Polygon Patch for Smoothing Quickly your Polygonal Meshes, Proceedings of the Computer Animation, 1998, pp. 72–78.
[34] J.R. Womersley, Oscillatory motion of a viscous liquid in a thin-walled elastic tube. I. The linear approximation for long waves, Philos. Mag. 46 (1955) 199–221.