Topics in Ultrascale Scientific Computing with Application in Biomedical Modeling

by

Leopold Grinberg

Sc.M., Applied Mathematics, Brown University, USA, 2007
Sc.M., Mechanical Engineering, Ben-Gurin University of the Negev, Israel, 2003
B.S., Mechanical Engineering, Ben-Gurin University of the Negev, Israel, 2001
Physician Assistant, Kamensk-Uralsky Medical College, Russia, 1991

Thesis

Submitted in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

May 2009

Abstract of "Topics in Ultrascale Scientific Computing with Application in Biomedical Modeling," by Leopold Grinberg, Ph.D., Brown University, May 2009

In this Thesis we focus on simulations of blood flow in three-dimensional patient-specific arterial networks. We employ high-order spectral/*hp*-element spatial discretization and concentrate on computational efficiency in solving multi-million degrees of freedom (DOF) flow problems on petaflop computers. We develop new two-level domain decomposition method and multilevel communicating interface for ultra-parallel flow simulations. Specifically, at the coarse level the computational domain is subdivided into several big patches. Within each patch a spectral element discretization (fine level) is employed. New interface conditions for the Navier-Stokes equations are developed. The proposed numerical approach has been tested in arterial flow simulations with up to 147 arteries. Solution of 2.87B DOF problem was computed on 18,576 processors in less than one second at each time step. A scalable and fast parallel low-energy bases preconditioner (LEBP) in conjunction with coarse-space linear vertex solver is developed. We provide details on optimization, parallel performance and implementation of the coarse-space solver and show scalability of LEBP on thousands processors of the IBM BlueGene/L and the Cray XT3. An embarrassingly parallel but extremely efficient accelerator for iterative solver has been proposed. The new approach reduces the number of conjugate gradient iterations and exhibits grid independent scaling, while the computational overhead is negligible. A new type of outflow boundary condition for networks with multiple outlets has been developed. The method is based on a time-dependent resistance-capacitance model, where the resistance values are related to the measured flowrates. The aforementioned methods have been employed to study unsteady flow in patient-specific intracranial arterial trees. Results of a comparative study of 3D rigid wall and 1D flexible wall modeling of flow in complex arterial networks are presented. Transient turbulent flow in a carotid arterial bifurcation with a stenosed internal carotid artery has been studied in details. To analyze the intermittent in time and space laminar-turbulent flow a new methodology based on time- and space-window Proper Orthogonal Decomposition (POD) is proposed. A simplified version of the POD analysis that utilizes 2D slices only - more appropriate in the clinical setting - is also investigated.

This dissertation by Leopold Grinberg is accepted in its present form
by the Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of Doctor of Philosophy

Date..... .....
George Em Karniadakis, Director

Recommended to the Graduate Council

Date..... .....
Jan Sickmann Hesthaven, Reader

Date..... .....
Peter Damian Richardson, Reader

Approved by the Graduate Council

Date..... .....
Sheila Bonde
Dean of the Graduate School

The Vita of Leopold Grinberg

Education

- Sc. M. Applied Mathematics, Brown University, USA, 2007.
- M. Sc. Department of Mechanical Engineering Ben-Gurion University of the Negev, Israel. 2003, Cum Laude.
- Physician Assistant, Kamensk-Uralsky Medical College, Russia, 1991, Cum Laude.

Publications

1. L. Grinberg, A. Yakhot and G. E. Karniadakis, *Analyzing Transient Turbulence in a Stenosed Carotid Artery by Proper Orthogonal Decomposition*, Annals of Biomedical Engineering (provisionally accepted, 2009).
2. L. Grinberg, T. Anor, E. Cheever, J. R. Madsen and G. E. Karniadakis, *Simulation of the Human Intracranial Arterial Tree, Philosophical Transactions of the Royal Society A*, (special issue, accepted, Dec. 2008).
3. L. Grinberg, D. Pekurovsky, S. Sherwin and G. E. Karniadakis, *Parallel Performance of the Coarse Space Linear Vertex Solver and Low Energy Basis Preconditioner for Spectral/hp Elements*, Parallel Computing, http://dx.doi.org/10.1016/j.parco.2008.12.002 (2008).
4. L. Grinberg and G. E. Karniadakis, *A Scalable Domain Decomposition Method for Ultra-Parallel Arterial Flow Simulations*, Communications in Computational Physics; (special issue) 4, 1151-1169 (2008).
5. L. Grinberg and G. E. Karniadakis, *Outflow Boundary Conditions for Arterial Networks with Multiple Outlets*, Annals of Biomedical Engineering, 36(9), 1496-1514 (2008).
6. L. Grinberg, T. Anor, J. R. Madsen, A. Yakhot and G. E. Karniadakis, *Large-Scale Simulation of the Human Arterial Tree*, Clinical and Experimental Pharmacology and Physiology, 36(2), 194-205 (2009). (special issue; doi: 10.1111/j.1440-1681.2008.05010.x).

7. L. Grinberg and G. E. Karniadakis, *Hierarchical spectral basis and Galerkin formulation using barycentric quadrature grids in triangular elements*, Journal of Engineering Mathematics, 56(3), 289-306 (2007).

8. G. Lin, L. Grinberg and G. E. Karniadakis, *Numerical studies of the stochastic Korteweg-de Vries equation*, Journal of Computational Physics, 213(2), 676-703 (2006).

9. A. Yakhot, L. Grinberg and N. Nikitin, *Modeling rough stenoses by an immersed-boundary method*, Journal of Biomechanics 38(5), 1115-1127 (2005).

10. A. Yakhot, L. Grinberg and N. Nikitin, *Simulating pulsatile flows through a pipe orifice by an immersed-boundary method*, Journal of Fluids Engineering 126(6), 911-918 (2004).

11. A. Yakhot and L. Grinberg, *Phase shift ellipses for pulsating flows*, Physics of fluids 77(15), 2081-2083 (2003).

Conferences

1. L. Grinberg and G. E. Karniadakis, *Unsteady 3D flow simulations in cranial arterial tree*, SIAM Conference on Computational Science and Engineering(CSE09), FL (Miami 2009).

2. L. Grinberg and G. E. Karniadakis, *Unsteady 3D flow simulations in cranial arterial tree*, 61st Annual Meeting of the APS Division of Fluid Dynamics, San Antonio, TX (November 2008).

3. L. Grinberg and G. E. Karniadakis, *Large scale 3D Arterial Flow Simulation with Hierarchical Domain Decomposition Method*, 16th Congress of the European Society of Biomechanics, Lucerne, Switzerland (July, 2008).

4. L. Grinberg, B. Toonen, N. Karonis and G. E. Karniadakis, *A Multilayer Approach to Simulate Large Multiscale Computational Mechanics Problems Using Grids*, Open Source Grid and Cluster Software, Oakland, CA (May 2008).

5. L. Grinberg, A. Yakhot and G. E. Karniadakis, *Onset of Turbulence in a Stenosed Carotid Artery*, Inaugural International Conference of the Engineering Mechanics Institute, Minneapolis, MN (May 2008)

6. L. Grinberg and G. E. Karniadakis, *A Multiscale Model for the Brain Vascular Network*, 60th Annual Meeting of the APS Division of Fluid Dynamics, Salt Lake City (2007).

7. L. Grinberg and G. E. Karniadakis, *Spectral/hp Element Simulation of the Human Arterial Tree on the TeraGrid*, USNCCM9 , San Francisco, CA (2007).

8. L. Grinberg, B. Toonen, N. Karonis and G. E. Karniadakis, *A New Domain Decomposition Technique for TeraGrid Simulations*, TeraGrid '07 , Madison, WI, (2007).

9. L. Grinberg and G. E. Karniadakis, *Decomposition of the Spectral Element Mesh in TeraGrid Simulation of the Human Arterial Tree*, ICOSAHOM, Beijing, China, (2007).

10. S. Dong, L. Grinberg, A. Yakhot, S. Sherwin and G. E. Karniadakis, *Simulation of Blood Flow in Human Arterial Tree on the TeraGrid*, SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, CA, Feb. (2006).

11. L. Grinberg, A. Yakhot and G. E. Karniadakis, *DNS of Flow in Stenosed Carotid Artery*, 59th Annual Meeting of the APS Division of Fluid Dynamics, Tampa, FL, (2006).

12. L. Grinberg and G. E. Karniadakis, *The Tale of Two Spectral Bases on the Triangle*, USNCCM8, Austin, TX (2005).

13. S. Dong, L. Grinberg, A. Yakhot, S. Sherwin and G. E. Karniadakis, *TeraGrid Simulations of Blood Flow in Human Arterial Tree*, 58th Annual Meeting of the APS Division of Fluid Dynamics, Chicago, IL, (2005).

Presentations

1. (seminar) L. Grinberg, *Large Scale Simulation of the Human Arterial Tree*, Boston University, Mechanical Engineering Department, January 23, (2009).

2. L. Grinberg, *Ultrascale Simulation of the Human Intracranial Arterial Tree*, SC08, Austin, TX (2008).

3. (seminar) L. Grinberg, *Large scale 3d arterial flow simulation with hierarchical domain decomposition method*, Institut Jean le Rond d'Alembert, Universit Paris 6 July 17, (2008).

4. L. Grinberg and G. E. Karniadakis, *Terascale Simulation of Arterial Blood Flow on the TeraGrid*, SC07, Reno, NV (2007) (invited talk).

5. L. Grinberg, K. Eschenberg and N. Stone, *Real-Time Visualization for Terascale Simulations with Spectral/hp Element Methods*, SC07, Reno, NV (2007).

6. L. Grinberg and G. E. Karniadakis, *Multi-Level Parallel Paradigm and Domain-Decomposition Technique for Human Arterial Tree Simulation*, SC07, Reno, NV (2007).

7. L. Grinberg, *Multilevel Parallelism and Locality-Aware Algorithms*, Petascale Applications Symposium, Pittsburgh Supercomputing Center, PA (2007).

8. L. Grinberg, K. Eschenberg and N. Stone, *Interactive Insight to Ongoing Computations*, SC06, Tampa, FL (2006).

9. S. Dong, L. Grinberg, J. Insley, N. Karonis, S. Spencer and G. E. Karniadakis, *TeraGrid Cross-Site Simulations and Visualizations of the Human Arterial Tree*, SC05 , Seattle, WA (2005).

Posters

1. L. Grinberg, J. Cazes and G. E. Karniadakis. *A Scalable Domain Decomposition Method for Ultra-Parallel Arterial Flow Simulation*, SC08, Austin, TX (2008, "The Best Poster" award winner).

2. L. Grinberg, J. Cazes and G. E. Karniadakis. *A Scalable Domain Decomposition Method for Ultra-Parallel Arterial Flow Simulation*, Fast Algorithms for Scientific Computing, NYU, New York, NY (2008).

3. T. Anor, L. Grinberg, J. R. Madsen and G. E. Karniadakis, *Large-scale Simulations of the Human Cranial Arterial tree: Utility in Hydrocephalus*, 52nd Annual Scientific Meeting, Society for Research into Hydrocephalus and Spina Bifida, Providence, RI (2008).

4. L. Grinberg, S. Dong, J. Noble, A. Yakhot, G. E. Karniadakis and N.T. Karonis, *Human arterial tree simulation on TeraGrid*, SC06, Tampa, FL (2006).

Honors and Awards

1. The Best Poster Award, Supercomputing'08 (2008).

2. Fulbright, United States-Israel Education Foundation (2003).

3. Wolf Foundation Award, Israel (2003).

Acknowledgments

During six years at Brown, I had an opportunity to interact with the faculty and staff members from the Division of Applied Mathematics, but I also collaborated with many scientists and HPC specialists from other academic institutions, National Laboratories, Supercomputing centers and hospitals. I am convinced that their support has been indispensible contribution to my research and I am sincerely grateful for their help. In particular, I want express my gratitude to the following people:

First of all, my utmost gratitude goes to my thesis adviser, Professor George Em Karniadakis, for giving me the inspiration, opportunity to learn, develop myself and succeed in the area of parallel scientific computing. I am thankful to Professor Karniadakis for providing me with a free highway, and low-latency high-bandwidth connections to the world of High Performance Computing, and giving me full support in experimenting, inventing and implementing my ideas on numerical and computational methods. Professor Karniadakis has been always constructive in his criticism and helped me a lot in structuring the way I present my research in journal papers, conferences and seminars.

I thank Professor Alexander Yakhot, who was my academic adviser at the Ben-Gurion University in the Negev in Israel and continued to collaborate with me after I left for Brown. I am grateful to him for broadening my knowledge in the area of fluid dynamics and numerical methods, and for long hours of fruitful scientific discussions.

I am grateful to my dissertation committee, Professors Jan Hesthaven and Peter Richardson, who have been kind to critique my work as readers of this Thesis.

I want also to thank:

- *Children's Hospital in Boston, Department of Neurosurgery*: Dr. Joseph Madsen for productive collaboration in research related to intra-cranial arterial flow.
- *Imperial College, London*: Professor Spencer Sherwin for productive collaboration.
- *Argonne National Laboratory*: Nicholas Karonis, Brian Toonen and Joseph Insley for a fruitful collaboration on Grid-computing and visualization.
- *San Diego Supercomputing Center:* Krishna Muriki and Dmitry Pekurovsky, for exceptional support and commitment in providing technical support and help in scaling my application on thousands of processors.

- *Pittsburgh Supercomputing Center:* David O'Neal, Greg Foss, Kent Eschenberg, Nathan Stone, Shawn Brown, Sergiu Sanielevici and Rolf Roskies (check names) for extraordinary support, and collaboration in developing new software for computation and visualization.

- *National Center of Scientific Applications:* David McWilliams for exceptional support he provided as an NCSA user support team member.

- *Texas Advanced Computing Center:* John Cazes, for exceptional user support and help leading to the Best Poster prize in Supercomputing'09.

- Numerous anonymous staff members who contributed to my research by providing a support in porting, debugging, optimizing and running my software at the following supercomputing centers:
  Pittsburgh Supercomputing Center (PSC),
  National Center of Scientific Applications (NCSA),
  San Diego Supercomputing Center (SDSC),
  Argonne National Laboratory (ANL),
  Texas Advanced Computing Center (TACC),
  National Institute for Computational Sciences at University of Tennessee/Oak Ridge National Laboratory (NICS),
  Arctic Supercomputing Center (ARSC),
  Engineer Research and Development Center (ERDC),
  Argonne Leadership Computing Facility (ALCF).

I thank my colleagues PhD and post-doc students as well the staff of the Division of Applied Mathematics for their support. I want to acknowledge and thank the National Science Foundation which has supported the research presented in this Thesis through the following grants: CI-TEAM grant OCI-0636336 and OCI-0845449.

I am grateful to Fulbright, United States-Israel Education Foundation, for supporting me to start my studies at Brown. I feel privileged to have been selected for the Fulbright award, and I sincerely hope that I will return it as a contribution to the broad scientific community.

Last but not least, I am sincerely thankful to my parents Alexander and Rivka Grinberg who have been supportive in so many ways. They never questioned my choice to study for

many years and so far away from home. I can not thank my parents enough for their unconditional love and understanding as I perused my academic degrees.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Multiscale modeling of the Virtual Physiological Human (VPH) has attracted a lot of attention recently and efforts are currently underway to address many aspects of this problem in the US, Europe and Japan [2]. The VPH project aims to simulate *integrated* physiological processes across different length and time scales (multi-scale modeling). It operates under assumption that it will eventually lead to a better healthcare system, and one of the goals of the project is personalized care solutions. The considerable variability of the genomic sequence, pathologies and even geometry of certain organs and systems (i.e. cardiovascular system) suggests that use of *patient-specific* data must be integrated in the system modeling. Simulating blood flow in the entire arterial tree is an indispensable component of VPH modeling and perhaps the most complex one from the mathematical and computational standpoint.

Computational fluid dynamics (CFD) enables accurate simulations of blood flow in arterial bifurcations, stented vessels and arteries with aneurysms or stenoses, among others. Results of these CFD simulations may aid in understanding the biomechanics of such pathologies and accelerate the use of CFD in presurgical computerassisted planning. Similarly, CFD simulations of blood flow in entire vascular networks will lead to a better understanding of oxygen transport to the tissues, as well as helping to design more effective procedures for drug delivery. In recent years, numerous studies have been devoted to CFD modeling of arterial flows, but most are limited to one or two arteries only. Interactions of blood flow in the human body occur between different scales, in which the large-scale flow features are coupled with cellular and subcellular biology, or at similar scales in different

regions of the vascular system. At the largest scale, the human arterial system is coupled through the wave-like nature of the pulse information traveling from the heart into elastic arteries. Surgical interventions, such as bypass grafts, leading to a change in the arterial network alter the wave reflections, which, in turn, can modify the flow waveforms at seemingly remote locations. Subsequently, the modification of a local waveform can lead to the onset of undesirable wall stresses, which could start another pathological event. At the cellular scale, the blood cell that plays a central role in the blood clotting process is the platelet. Blood vessels injured by smoking, cholesterol or high blood pressure develop cholesterol-rich plaques that line the blood vessel wall; these plaques can rupture and cause the platelets to form a clot, which leads to a blockage of blood flow. In arteries of the heart, the process leads to chest pain and heart attack; in arteries of the neck and brain, the process causes strokes. *Numerical simulations* of such pathologies require multiscale modeling across many orders of magnitude of spatial and temporal scales. Indeed, there is the prospect of coupling multiscale representations of blood flow, ranging from a quasi one-dimenstional (1D) transient flow in compliant vessels at the largest scale, to unsteady three-dimensional (3D) flows in curved and flexible vessels at the mm range, to μm scale thrombus formation at a fissure in the lumen of such a vessel with an atherosclerotic plaque. **The main challenge of simulating blood flow interactions, from the scale of the entire human arterial tree down to the scale of platelet aggregation, lies in the high demand such a task places on new algorithms for supercomputing**. High-performance computing is essential to modeling the 3D unsteady fluid mechanics within sites of interest, such as arterial branches and the process of platelet aggregation and thrombosis with tens of millions of platelets.

Three-dimensional simulation of unsteady blood flow in entire vascular system requires solution of Navier-Stokes equations with billions degrees of freedom. High numerical accuracy is important in order to minimize the discretization error and be able to focus on the proper blood flow modeling. In this study we consider the high-order spectral/*hp* element discretization (SEM). SEM provides high accuracy and treats geometrical complexity effectively. In order to exploit effectively the advantages of SEM fast and scalable parallel numerical solvers must be designed, so the advantages of the method will not be outweighed by high computational cost. **Designing robust numerical methods and scalable parallel algorithms while maintaining high spatial and temporal resolution is the**

main focus of this Thesis. In particular, we emphasized high-order accuracy, computational efficiency and extreme scalability. The developed methods have been applied to simulate laminar and transient flow in arteries, reconstructed from patient-specific data; however, they can also be applied in solving a wide range of physical problems, e.g., from simulations multiphysics in virtual nuclear reactors to modeling environmental flows.

Outline

Chapter 2 is devoted to numerical study of two spectral bases and three sets of quadrature grid defined on two-dimensional triangular elements. As a model problem, the Navier-Stokes equations is considered. The semi-implicit and implicit time-stepping schemes are applied. The advantages and disadvantages of discretization of the pressure field with the reduced-space approach are discussed.

Chapter 3 focuses on fast and scalable iterative solver for solution of Helmhlotz and Poisson equations for the velocity and pressure variables, respectively. Specifically, we focus on the parallel low-energy bases preconditioner and coarse-space linear vertex solver. We also propose robust methods to accelerate convergence of the iterative solver in solving dynamical systems by providing a better initial guess. As a model problem we consider flow simulations in patient-specific arterial trees. The large size of computational domain and use of high-resolution spatial discretization leads to system of non-linear equations with millions degrees of freedom. However, despite the large problem size the numerical simulation can be completed very fast, if proper numerical and parallel algorithms are employed.

In *Chapter* 4 we address numerical challenges in flow simulations in complex patient-specific arterial networks, with multiple inlets and outlets. First, we present a new type of outflow boundary condition which allows to incorporate efficiently available clinical data into numerical simulation. Second, we present a new computational approach for solution of extremely large flow problems (with billions of degrees of freedom). Particularly, we focus on a two-level domain decomposition method and developing an ultra-parallel paradigm for solving flow problems on a network of distributed computers and also on a single cluster. Third, we present results of unsteady 1D and high-resolution 3D simulations of a flow in the brain vasculature.

In *Chapter* 5 we present a study of intermittent laminar-turbulent-laminar flow in stenosed carotid artery. High-resolution three-dimensional simulations (involving 100 million degrees of freedom) were employed to study transient turbulent flow in a carotid ar-

terial bifurcation with a stenosed internal carotid artery (ICA). The geometrical model of the arteries was reconstructed from MRI images, and in vivo velocity measurements were incorporated in the simulations to provide inlet and outlet boundary conditions. Due to high degree of the ICA occlusion and variable flow rate, a transitional and intermittent flow between laminar and turbulent states was established. Time- and space-window Proper Orthogonal Decomposition (POD) was applied to quantify the different flow regimes in the occluded artery. A simplified version of the POD analysis that utilizes 2D slices only more appropriate in the clinical setting - is also investigated.

In *Chapter* 6 we conclude with a brief discussion and outlook.

Chapter 2

A study of Barycentric and Cartesian tensor product spectral element bases and quadrature grids in triangular elements

2.1 Introduction

Spectral element methods in two-dimensional domains consisting of triangular elements have only been developed in the last fifteen years [79, 78]. They are particularly effective for solving time-dependent partial differential equations in truly complex-geometry domains. The spectral expansions employed in each triangle to represent the data, solution and geometry can be either of *modal* or *nodal* type. Hierarchical modal expansions can be cast in a tensor-product form if properly formulated, see [49]; they are typically used in conjunction with Galerkin projections. Nodal expansions are non-hierarchical and are usually employed in collocation type methods; computing the proper set of nodes in a triangle has been the subject of several studies, see [29, 45, 91, 23].

In this Chapter we focus on two fundamental aspects of SEM/*hp* discretization, specifically the spectral bases and the quadrature grid. We compare the computational efficiency of solution partial differential equations with different bases functions and quadrature grids. The Chapter is organized as follows. First, we overview the cartesian and barycentric coor-

dinate systems. Second, we present the cartesian and barycentric tensor product bases and compare numerical properties of linear operators constructed from these bases. Third, we present two numerical schemes for solution of Navier-Stokes equations, discuss aspects of $P/P-k,\ k=0,1,2$ approach and present results of numerical solution of two flow problems. Forth, we focus on performance of Navier-Stokes solver in conjunction with three sets of quadrature grids. We conclude with a summary in the last section of this Chapter.

List of Symbols/Notations

Ω - computational domain.

Ω_e - computational domain corresponding to spectral element e.

$\mathbf{x}$ - cartesian coordinate system.

ξ - coordinates of cartesian system, defined on Ω_e.

η - collapsed coordinate system.

$\mathbf{L}$ - barycentric coordinate system, defined on Ω_e.

$\mathbf{\Phi} = [\Phi_1, ..., \Phi_N]^T$, and $\mathbf{\Psi} = [\Psi_1, ..., \Psi_N]^T$ - cartesian and barycentric tensor product basis, collectively denoted by $\mathbf{\Lambda} = [\lambda_1, ..., \lambda_N]^T$.

P - order of polynomial expansion.

$\mathbf{V}$ - Vandermond matrix.

$\mathbf{M}$, $\mathbf{D}$ and $\mathbf{L}$ - projection, advection and laplacian operators.

$\omega_i, i = 1, ..., N$ - integration weights.

2.2 A study of barycentric and cartesian tensor product bases

Different sets of polynomial bases spanning the same space are equivalent in terms of accuracy. However, the corresponding computational efficiency depends on the *type* of basis functions. We present here a systematic study of the relative merits of two sets of tensor-product basis functions defined on a triangular element.

The *first* set of basis functions is described in [79]. This tensor-product basis consists of hierarchical functions that are defined in terms of the so-called "collapsed" cartesian coordinates, here and thereafter these basis functions are denoted by $\mathbf{\Phi}(\xi)$, $\xi = [\xi_1, \xi_2]$. Three-dimensional bases defined on polymorphic domains (e.g., tetrahedra, prisms and pyramids) are readily constructed with appropriate modification of the coordinate in each direction. The L_2 basis set is orthogonal while the C^0 basis set is semi-orthogonal. The bases lack rotational symmetry. The *second* set of basis functions was introduced in [22]. The tensor-product bases $\Psi(\mathbf{L})$ is constructed using barycentric coordinates $\mathbf{L} = [L_1, L_2, L_3]$. These basis functions offer rotational symmetry but not orthogonality.

The aim of the present study is to compare both accuracy and computational efficiency of the two spectral representations on computational domains consisting of uniform and highly distorted triangular elements. While the comprehensive study of $\mathbf{\Phi}(\xi)$ has been performed,

little is known about the barycentric bases $\mathbf{\Psi}(\mathbf{L})$.

Software: Two-dimensional spectral/*hp* element solver has been developed. The software uses two sets of C^0 modal spectral bases defined on a triangular element, and three sets of quadrature points. The linear systems arising in different problems are solved directly or iteratively with preconditioned conjugate gradient solver, the diagonal and incomplete Cholesky preconditioners are employed. The software makes extensive use of sparse algebra; specifically, the matrix-vector multiplications required by the conjugate gradient solver are executed using sparse matrices stored in a compressed row storage format. For solution of Navier-Stokes equations an implicit and semi-implicit time stepping schemes are implemented, in the later the non-linear term is integrated using upto third-order Adams-Bashforth method, while the linear part is solved implicitly. For solution of Navier-Stokes problem, the so-called $P/P-k,\ \ k=0,1,...P-1$ approach is implemented, i.e., the velocity field is approximated by a polynomial expansion of order P, while the pressure is approximated with a polynomial expansion of order $P-k$. To this end, we first present the two spectral bases, examine the properties of the projection, advection and diffusion operators; and subsequently solve several prototype problems in order to evaluate the performance of the two bases.

2.3 Local coordinate systems

Two types of 2D coordinate systems are employed: a) the Cartesian and b) the Barycentric. The basis $\Phi_k(\xi)$ is defined on Cartesian coordinate system, whereas the basis $\Psi_k(\mathbf{L})$ is defined on Barycentric coordinate system.

Cartesian coordinate system

Consider a standard quadrilateral element and a 2D Cartesian coordinate system η_1, η_2. The dimensions of standard element are bounded by constant limits, i.e.,

$$Q^2 = (\eta_1, \eta_2)| -1 \leq \eta_1, \eta_2 \leq 1,$$

as shown in figure 2.1. Now consider a standard triangular element in the cartesian coordinate system (ξ_1, ξ_2)

$$T^2 = (\xi_1, \xi_2)| -1 \leq \xi_1, \xi_2; \xi_1 + \xi_2 \leq 0,$$

also shown in figure 2.1. Then, the ξ_1, ξ_2 coordinate system on a triangular element is obtained by mapping of the standard coordinate system by the transformation

$$\eta_1 = 2\frac{1+\xi_1}{1-\xi_2} - 1, \quad \eta_2 = \xi_2. \tag{2.1}$$

The transformation (2.1) defines the *collapsed-coordinate system*. The backward transformation is given by:

$$\xi_1 = \frac{(1+\eta_1)(1-\eta_2)}{2} - 1, \quad \xi_2 = \eta_2. \tag{2.2}$$

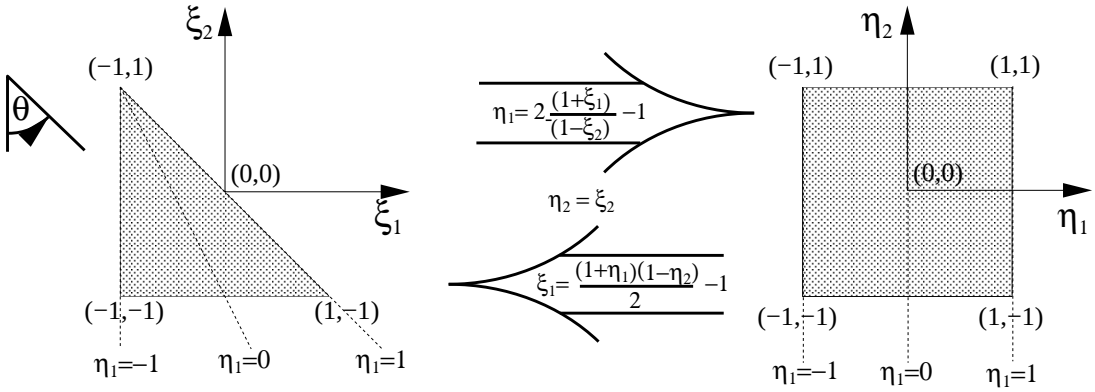


Figure 2.1: Cartesian (ξ_1, ξ_2) and collapsed-cartesian (η_1, η_2) coordinate systems.

The transformation from a *local* to element coordinate system ξ_1, ξ_2 to global coordinate system x, y is given by

$$x_i = x_i^A \frac{-\xi_2 - \xi_1}{2} + x_i^B \frac{1+\xi_1}{2} + x_i^C \frac{1+\xi_2}{2}, \tag{2.3}$$

where $x_1 = x$, $x_2 = y$, and x_j^A, x_j^B, x_j^C are vertex coordinates.

Barycentric coordinate system

The barycentric coordinate system on a triangular region is defined by three *non-independent* coordinates $L_1, L_2, L_3, \quad L_1 + L_2 + L3 = 1$ as illustrated in figure 2.2a. The coordinates of

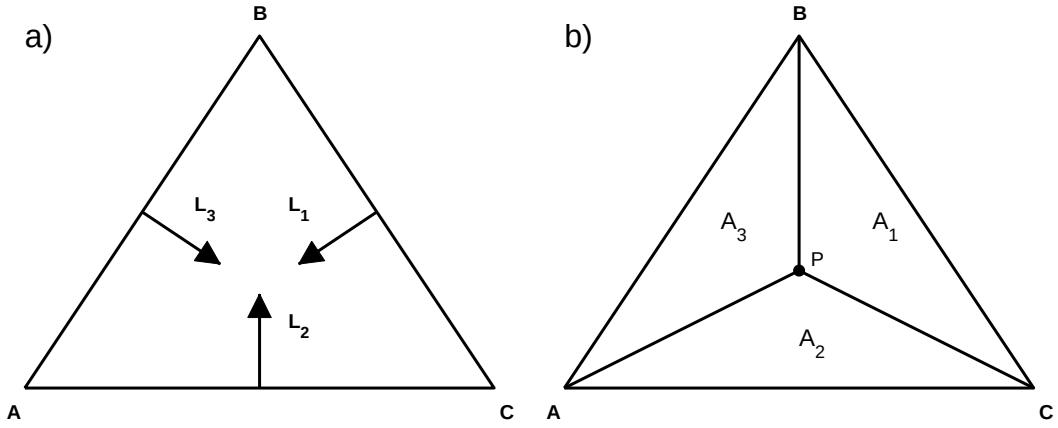


Figure 2.2: Triangular elements and barycentric coordinate system.

a grid point $P(L_{1_i}, L_{2_i}, L_{3_i})$ are computed as a ratio of areas of triangles ABP, BCP, CAP (figure 2.2b) to the area of an element ABC, i.e.,

$$L_i = \frac{A_i}{A}, \quad i = 1, 2, 3.$$

Coordinate transformation

The transformation from a Cartesian coordinate system x, y into Barycentric coordinate system is given by

$$\begin{aligned} L_1 &= [x^C y^B - x^B y^C + x(y^C - y^B) + y(x^B - x^C)]\tfrac{1}{2A'} \\ L_2 &= [x^A y^C - x^C y^A + x(y^A - y^C) + y(x^C - x^A)]\tfrac{1}{2A'} \\ L_3 &= [x^B y^A - x^A y^B + x(y^B - y^A) + y(x^A - x^B)]\tfrac{1}{2A'} \end{aligned}, \tag{2.4}$$

where A' is an area of the triangle.

The transformation from Barycentric coordinate system into collapsed coordinate system η_1, η_2 is given by

$$\begin{aligned} \eta_1 &= 2L_2 - 1 \\ \eta_2 &= \frac{2L_3}{1-L_2} - 1 \end{aligned} \tag{2.5}$$

and the backward mapping is given by

$$\begin{aligned} L_1 &= 1 - L_2 - L_3 \\ L_2 &= 0.5(1 + \eta_1) \\ L_3 &= 0.25(1 - \eta_1)(1 + \eta_2) \end{aligned} \tag{2.6}$$

2.4 Two-dimensional spectral bases

Consider two dimensional region $\Omega \subset R^2$ subdivided into Nel non-overlapping elements $\Omega = \cup_{e=1}^{Nel}\Omega_e, \quad \Omega_i \cap \Omega_j = 0, \quad i \neq j$. In each element the solution field is approximated by a finite-order polynomial expansion defined in a space $\mathcal{V}^\delta$ of order P:

$$u(\mathbf{x}) \approx u^\delta(\mathbf{x}) = \sum_{k}^{dim(\mathcal{V}^\delta)} \hat{v}_k \Phi_k(\xi) \equiv \sum_{k}^{dim(\mathcal{V}^\delta)} \hat{u}_k \Psi_k(\mathbf{L}). \tag{2.7}$$

By $\Phi_k(\xi)$ and $\Psi_k(\mathbf{L}), \quad k = 1, ..., dim(V^\delta)$ we denote the Cartesian and Barycentric tensor product bases; $\mathbf{L}$ and ξ are local to the element Barycentric and Cartesian coordinates. The superscript δ emphasize that we use a finite (truncated) space, notation $\mathcal{V}^P$ will denote space spanned by polynomials of order $\leq P$. Both bases are subdivided into a linear (vertex) modes, edge modes and interior modes. The linear modes have zero support at all vertices except one; analogously, the edge modes have zero support at all edges except one edge. The interior modes (also called bubble modes) have zero support on the perimeter of the element.

2.4.1 Cartesian tensor product bases

Consider standard triangular element Ω_e and a local to this element Cartesian coordinate system ξ_1, ξ_2 as shown in figure 2.1(left). Standard triangular element is mapped to a quadrilateral region by means of a mapping function 2.1 the mapping is illustrated in figure 2.1. The C^0 continuous expansion *modal* bases are defined on η_1, η_2 as following:

Vertex A: $\left(\frac{1-\eta_1}{2}\right)\left(\frac{1-\eta_2}{2}\right)$

Vertex B: $\left(\frac{1+\eta_1}{2}\right)\left(\frac{1-\eta_2}{2}\right)$

Vertex C: $\left(\frac{1+\eta_2}{2}\right)$

Edge AB: $\left(\frac{1-\eta_1}{2}\right)\left(\frac{1+\eta_1}{2}\right)P_{p-1}^{1,1}(\eta_1)\left(\frac{1-\eta_1}{2}\right)^{p+1} \quad (0 < p < P_1)$

Edge BC: $\left(\frac{1+\eta_1}{2}\right)\left(\frac{1-\eta_2}{2}\right)\left(\frac{1+\eta_2}{2}\right)P_{q-1}^{1,1}(\eta_2) \quad (0 < q < P_2)$

Edge AC: $\left(\frac{1-\eta_1}{2}\right)\left(\frac{1-\eta_2}{2}\right)\left(\frac{1+\eta_2}{2}\right)P_{q-1}^{1,1}(\eta_2) \quad (0 < q < P_2)$

Interior: $\left(\frac{1-\eta_1}{2}\right)\left(\frac{1+\eta_1}{2}\right)P_{p-1}^{1,1}(\eta_1)\left(\frac{1-\eta_2}{2}\right)^{p+1}\left(\frac{1+\eta_2}{2}\right)P_{q-1}^{2p+1,1}(\eta_2)$

$$(0 < p, q; p < P_1; p + q < P_2, P_1 \leq P_2),$$

where $P_i^{\alpha,\beta}$ are Jacobi polynomials of order i, P_1 and P_2 define the highest order of the Jacobi polynomials in direction η_1 and η_2 respectively. In the present study we let $P_1 = P_2 \equiv P$.

A convenient numeration of the expansion modes, Φ_i, is as following:

Φ_i for i = 1,2,3 correspond to the linear Vertex modes (A, B and C).

Φ_{i+e} for i = 1,...,(Pmax-1) correspond to the Edge modes. For the shape function which are not zero on a) Edge AB, $e = 3$; b) Edge BC, $e = 3 + (P - 1)$; c) Edge AC, $e = 3 + 2(P - 1)$.

Φ_{i+k} for i = 1,...,(P-1)(P-2)/2 and k = 3P correspond to the interior modes. Additionally, in order to achieve higher computational efficiency, index q runs faster then p, i.e., $p = q = 1$ corresponds to the first face mode, $p = 1,\ q = 2$ for the second and so on.

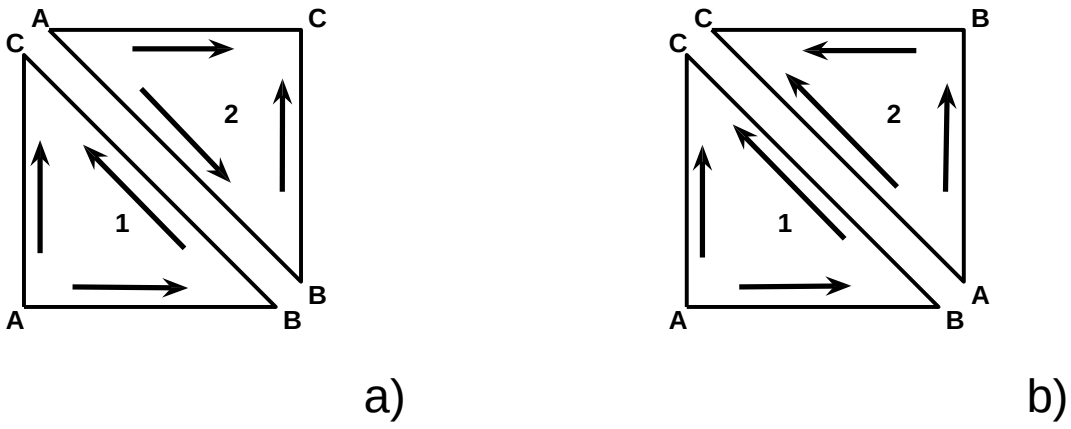


Figure 2.3: Imposing C^0 continuity on standard triangular elements. Arrows indicate the direction of local coordinate system η_1, η_2.

When a two dimensional region Ω is decomposed into a set of non-overlapping elements, i.e., $\Omega = \sum_{e=1}^{Nel} \Omega_e,\ \Omega_i \cap \Omega_j = 0,\ i \neq j$, and global C^0 continuity is required, the following modification of edge shape functions might be necessary. In Figs. 2.3a and 2.3b we present two possible configurations where

a) Edge BC of Element 1, and Edge AB of Element 2 have the opposite direction (defined by the collapsed coordinates). Then the global C^0 continuity between Element 1 and 2 is insured by means of negating the odd Edge modes of one of the elements.

b) Edge BC of Element 1, and Edge AC of Element 2 have the same direction, consequently the global continuity is preserved, and no modification is required.

2.4.2 Barycentric tensor product bases

To describe the $\mathbf{\Psi}$ expansion bases we consider a triangular element and Barycentric coordinate system L_1, L_2, L_3 as illustrated in figure 2.2. Note that the coordinates satisfy $L_1 + L_2 + L_3 = 1$, hence they are linearly depended. The C^0 continuous expansion modal

bases are defined on L_1, L_2, L_3 as following:

Vertex A: L_1

Vertex B: L_2

Vertex C: L_3

Edge AB: $L_1 L_2 P_{p-1}^{\alpha,\beta}(2L_1-1) P_{q-1}^{\alpha,\beta}(2L_2-1), \quad (p+q=P, \quad 0<p,q<P)$

Edge BC: $L_2 L_3 P_{q-1}^{\alpha,\beta}(2L_2-1) P_{r-1}^{\alpha,\beta}(2L_3-1), \quad (q+r=P, \quad 0<q,r<P)$

Edge CA: $L_3 L_1 P_{r-1}^{\alpha,\beta}(2L_3-1) P_{p-1}^{\alpha,\beta}(2L_1-1), \quad (r+p=P, \quad 0<r,p<P)$

Interior: $L_1 L_2 L_3 P_{p-1}^{\alpha,\beta}(2L_1-1) P_{q-1}^{\alpha,\beta}(2L_2-1) P_{r-1}^{\alpha,\beta}(2L_3-1), \quad (p+q+r=P, \quad 0<p,q,r<P-1)$.

A convenient numeration of the expansion modes Ψ_i is as follows:

Ψ_i for i = 1,2,3 correspond to the linear Vertex modes (A, B and C).

Ψ_{i+e} for i = 1,...,(Pmax-1) correspond to the Edge modes. For the shape function which are not zero on a) Edge AB, $e = 3$, b) Edge BC, $e = 3 + (P-1)$, and c) Edge AC, $e = 3 + 2(P-1)$. Since each Edge mode is defined as a tensorial product of polynomials of order p and q (for the Edge AB, q and r for the Edge BC, r and p for the Edge CA) we let the first index to run from 1 to $P-1$ and compute the second index as $q = P - p$. For example, consider the Edge AB and $P = 4$, then

$\Psi_4 = L_1 L_2 P_{1-1}^{\alpha,\beta}(2L_1-1) P_{3-1}^{\alpha,\beta}(2L_2-1)$

$\Psi_5 = L_1 L_2 P_{2-1}^{\alpha,\beta}(2L_1-1) P_{2-1}^{\alpha,\beta}(2L_2-1)$

$\Psi_6 = L_1 L_2 P_{3-1}^{\alpha,\beta}(2L_1-1) P_{1-1}^{\alpha,\beta}(2L_2-1)$

Ψ_{i+k} for $i = 1, ..., (P-1)(P-2)/2$ and $k = 3P$ correspond to the interior modes.

In this study we use $\alpha = 0, \ \beta = 2$.

The C^0 continuity between adjacent elements is imposed by reversing the $p-q$ ($q-r$, $r-p$) sequence for Edge AB (BC, CA). For instance, consider the 2-element mesh as shown in Fig. 2.4 and $P = 4$. To ensure the C^0 continuity between Edge BC functions of Element 1 end Edge CA functions of Element 2 we define the corresponding basis functions of Element 1 in the following pattern:

$\Psi_7 = L_2 L_3 P_{\mathbf{1}-1}^{\alpha,\beta}(2L_2-1) P_{\mathbf{3}-1}^{\alpha,\beta}(2L_3-1)$

$\Psi_8 = L_2 L_3 P_{\mathbf{2}-1}^{\alpha,\beta}(2L_2-1) P_{\mathbf{2}-1}^{\alpha,\beta}(2L_3-1)$

$\Psi_9 = L_2 L_3 P_{\mathbf{3}-1}^{\alpha,\beta}(2L_2-1) P_{\mathbf{1}-1}^{\alpha,\beta}(2L_3-1)$

and corresponding base functions of Element 2 as:

$\Psi_{10} = L_3 L_1 P_{\mathbf{3}-1}^{\alpha,\beta}(2L_3-1) P_{\mathbf{1}-1}^{\alpha,\beta}(2L_1-1)$

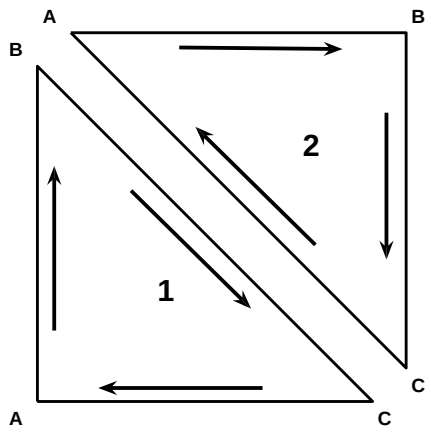


Figure 2.4: Imposing C^0 continuity on a standard triangular elements. Arrows indicate a positive direction of barycentric coordinates along edges: L_1 along edge CA, L_2 along edge AB and L_3 along edge BC.

$$\Psi_{11} = L_3 L_1 P^{\alpha,\beta}_{\mathbf{2}-1}(2L_3 - 1) P^{\alpha,\beta}_{\mathbf{2}-1}(2L_1 - 1)$$
$$\Psi_{12} = L_3 L_1 P^{\alpha,\beta}_{\mathbf{1}-1}(2L_3 - 1) P^{\alpha,\beta}_{\mathbf{3}-1}(2L_1 - 1)$$

2.4.3 Numerical integration

The exact integration of a function u on a region Ω_e

$$I = \int_{\Omega_e} u dA$$

can be approximated by numerical integration, i.e.,

$$I = \sum_{i=1}^{N} \omega_i u_i + R,$$

where ω_i is i-th quadrature weight, u_i is a value of a function u computed at a point i and R is the quadrature error. Different types of grids defined on a triangular region lead to different integration techniques.

Integration on the collapsed-cartesian grid

The exact integration of a function u on the standard triangular region using the collapsed-coordinate system (η_1, η_2) is expressed as

$$I = \int_{\tau^2} u(\xi_1, \xi_2) d\xi_1 d\xi_2 = \int_{-1}^{1} \int_{-1}^{-\xi_2} u(\xi_1, \xi_2) d\xi_1 d\xi_2 = \int_{-1}^{1} \int_{-1}^{1} u(\eta_1, \eta_2) J d\eta_1 d\eta_2, \qquad (2.8)$$

where J is the Jacobian of the transformation (2.2)

$$J = \left| \frac{\partial(\xi_1, \xi_2)}{\partial(\eta_1, \eta_2)} \right|.$$

The numerical counterpart of the last term in (2.8) is

$$I = \sum_{i=1}^{i=Q_1} \omega_{1_i} \left[\sum_{j=1}^{j=Q_2} \omega_{2_j} u(\eta_{1_i}, \eta_{2_j}) J(\eta_{1_i}, \eta_{2_j}) \right] + R. \qquad (2.9)$$

It is efficient to use Gauss quadrature points and weights for numerical integration. The Jacobian, J, is usually included into the weights ω_1, ω_2. In the case where u is a polynomial of degree $P_1(P_2)$ in $\xi_1(\xi_2)$ we need only $Q_1 = (P_1+1)/2$ ($Q_2 = (P_2+1)/2$) Gauss quadrature points to perform the exact numerical integration. Thus, the precise cost of numerical integration of a polynomial of order P (here and thereafter we assume that $P_1 = P_2 = P$ and $Q_1 = Q_2 = Q$) on the collapsed-coordinate system is $(3Q^2 + Q)$ multiplications and summations, with function evaluations at Q^2 quadrature points.

Integration on the barycentric grids

The exact integration of a function u on the standard triangular region using the barycentric coordinate system is given by

$$I = 2A \int_0^1 \int_0^{1-L2} f(\mathbf{L}) dL_1 dL_2,$$

where A is the elements area. The numerical integration on a barycentric grid over a triangular region with area A is given by

$$I = 2A \sum_{i=1}^{N} \omega_i u(L_{1_i}, L_{2_i}) + R',$$

with the weights, ω_i. Several sets of integration points and corresponding weights are available, and the discussion on a various choices is provided in section 2.8.1. The general approach to obtain ω_i for a given set of *barycentric* quadrature points L_i is as follows. Consider polynomial basis $\lambda_k(m,n)) \in \mathcal{V}^P,\ \mathcal{V}^P = span\{x_1^n y_2^m, m+n \leq P\},\ k = 1, ..., Q$, a set of integration points $L_i, i = 1, ..., Q$ and the corresponding square Vandermonde matrix $\mathbf{V}$ constructed from $\lambda(m,n)$ evaluated at grid points L_i, i.e, $V(k,i) = \lambda_k(L_i)$. Then the weights ω_i can be computed by solving a linear problem

$$\mathbf{V}\omega = \int \mathbf{\Lambda} dA,$$

where $\mathbf{\Lambda}$ is a vector formed from the polynomial basis $\lambda(m,n)$.

2.4.4 Numerical differentiation

The evaluation of derivatives in a physical space can be performed using a chain rule. For the barycentric bases we use:

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial L_2}{\partial x} & \frac{\partial L_3}{\partial x} \\ \\ \frac{\partial L_2}{\partial y} & \frac{\partial L_3}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial L_2} \\ \\ \frac{\partial}{\partial L_3} \end{bmatrix}. \tag{2.10}$$

Since the three barycentric axes are not orthogonal we my choose a different coordinate transformation, for example $xy \rightarrow L_1 L_2$ or $xy \rightarrow L_1 L_3$. Computing derivatives directly in a barycentric coordinate system instead of $\eta_1\eta_2$ coordinate system is more efficient from computational point of view. For example the derivatievs of the Vertex modes are computed from $\partial\Psi_i/\partial x = \partial$Ł$_i/\partial x$. Derivatievs of the Edge modes, $\Psi(L_i, L_j)$ are computed using corresponding $xy \rightarrow L_i L_j$ transformation. Finally derivatives of face modes can be computed by $xy \rightarrow L_1 L_2$ and substituting $L_3 = 1 - L_1 - L_2$. We can also compute the derivatives using $\eta_1\eta_2$ coordinate system:

$$\begin{bmatrix} \frac{\partial}{\partial L_2} \\ \\ \frac{\partial}{\partial L_3} \end{bmatrix} = \frac{1}{J} \begin{bmatrix} \frac{\partial L_3}{\partial \eta_2} & -\frac{\partial L_3}{\partial \eta_1} \\ \\ -\frac{\partial L_2}{\partial \eta_2} & \frac{\partial L_2}{\partial \eta_1} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \eta_1} \\ \\ \frac{\partial}{\partial \eta_2} \end{bmatrix}, \tag{2.11}$$

where $J = (1 - \eta_1)/8$. Thus the derivatives in x- and y- directions are computed from:

$$\frac{\partial}{\partial x} = \frac{\partial L_2}{\partial x}\frac{1}{J}\left[\frac{1-\eta_1}{4}\frac{\partial}{\partial \eta_1} + \frac{1+\eta_2}{4}\frac{\partial}{\partial \eta_2}\right] + \frac{\partial L_3}{\partial x}\frac{1}{J}\frac{1}{2}\frac{\partial}{\partial \eta_2} \tag{2.12}$$

$$\frac{\partial}{\partial y} = \frac{\partial L_2}{\partial y}\frac{1}{J}\left[\frac{1-\eta_1}{4}\frac{\partial}{\partial \eta_1} + \frac{1+\eta_2}{4}\frac{\partial}{\partial \eta_2}\right] + \frac{\partial L_3}{\partial y}\frac{1}{J}\frac{1}{2}\frac{\partial}{\partial \eta_2} \tag{2.13}$$

To compute derivatives of cartesian bases we use:

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{2}{A'}\begin{bmatrix} \frac{\partial y}{\partial \xi_2} & -\frac{\partial y}{\partial \xi_1} \\ \\ -\frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_1} \end{bmatrix}\begin{bmatrix} \frac{\partial}{\partial \xi_1} \\ \\ \frac{\partial}{\partial \xi_2} \end{bmatrix}, \tag{2.14}$$

where the partial derivatives in ξ_i are computed from:

$$\begin{bmatrix} \frac{\partial}{\partial \xi_1} \\ \\ \frac{\partial}{\partial \xi_2} \end{bmatrix} = \begin{bmatrix} \frac{2}{1-\eta_2}\frac{\partial}{\partial \eta_1} \\ \\ 2\frac{1+\eta_1}{1-\eta_2}\frac{\partial}{\partial \eta_1} + \frac{\partial}{\partial \eta_2} \end{bmatrix} \tag{2.15}$$

2.5 Construction of linear operators

The three linear operators considered in this section are the Projection (**M**), Advection (**D**), and Diffusion (**L**) operators. These operators are required for solutions of many physical problems. For example, consider one-dimensional unsteady advection-difusion problem

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2}.$$

The weak formulation of this problem is obtained by pre-multiplying the left and the right hand sides of the equation by the weight of a test function $\mathbf{\Lambda} = [\lambda_1, ..., \lambda_N]^T$ and integrating over Ω.

$$\int \mathbf{\Lambda} \left[\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} \right] dx.$$

Following the Bubnov-Galerkin formulation we approximate the solution u by a polynomial expansion of order P (see formula 2.7) and use the same basis functions for to test functions. For convenience let us rewrite the above integral as $I_1 = I_2 + \nu I_3$ where

$$I_1 = \left(\sum_{i=1}^{N} \frac{\partial}{\partial t} \hat{u}_i \lambda_i \ , \ \mathbf{\Lambda} \right) = \mathbf{M} \frac{\partial}{\partial t} \hat{\mathbf{u}},$$

$$I_2 = \left(\sum_{i=1}^{N} \hat{u}_i \frac{\partial}{\partial x} \lambda_i \ , \ \mathbf{\Lambda} \right) = \mathbf{D} \hat{\mathbf{u}},$$

$$I_3 = \left(- \sum_{i=1}^{N} \hat{u}_i \frac{\partial}{\partial x} \lambda_i \ , \ \frac{\partial}{\partial x} \mathbf{\Lambda} \right) = -\mathbf{L} \hat{\mathbf{u}},$$

in the last equality integration by parts was applied and zero flux boundary conditions were assumed.

2.5.1 Construction of a Mass Matrix

The construction of Mass matrix (Projection operator) $\mathbf{M}^e$ corresponding to element e for different expansion bases requires evaluation of the inner product of the modes in the expansion basis with themselves, i.e.,

$$M^e_{i,j} = (\lambda_i, \lambda_j) = \int_{\Omega_e} \lambda_i(\xi_1, \xi_2) \lambda_j(\xi_1, \xi_2) d\xi_1 d\xi_2,$$

where $\lambda_i = \Phi_i$ or $\lambda_i = \Psi_i$. Mass matrices constructed from the expansion bases $\mathbf{\Psi}$ and $\mathbf{\Phi}$ bases are not the same, they differ in structure (sparsity) and eigenproperties. Understanding and proper utilization of the operator structure may lead to considerable computational savings in terms of floating point operations and storage. The eigenproperties of the operators are important in analysis of stability and also convergence of iterative solvers. In the following we describe the structure of $\mathbf{M}^e$ constructed from the two spectral basis.

The Mass matrix constructed from $\mathbf{\Phi}$ is symmetric and consequently we have to compute only the diagonal and half of off-diagonal entries. Second, the Mass matrix is very sparse, thus we are allowed to skip the computation of a Matrix components which values are zero. In figure 2.5 we present a structure of $\mathbf{M}^e$ for a triangular expansion of order $P = 14$. It is clearly seen that the interior-interior block has a well defined bandwidth.

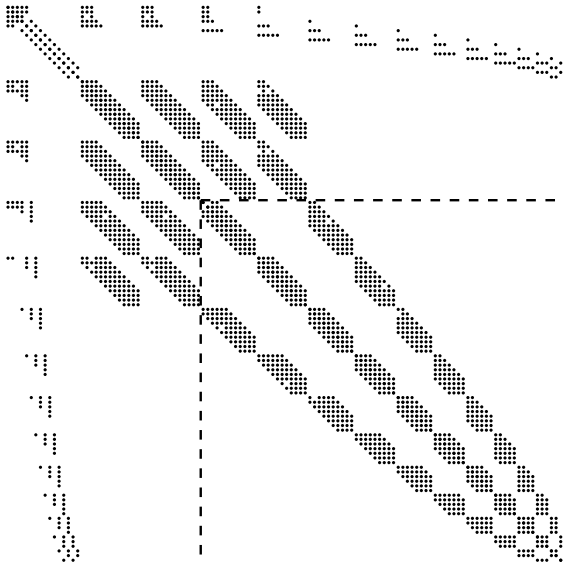


Figure 2.5: Schematic structure of a Mass matrix constructed from a Φ-type bases with $P = 14$. The dots represent the non-zero components of the matrix. The dash lines separate the interior-interior part. The bandwidth of the interior-interior block is $(P-2)+(P-3)+1$.

The Mass matrix, constructed form $\mathbf{\Psi}$ bases does not show the same sparsity, actually it is almost a full matrix. The last implies that computation of $\mathbf{M}$ is expensive. However, exploiting the rotational symmetry of the expansion bases $\mathbf{\Psi}$ reduces significantly the computational cost. To appreciate this fact let us divide the matrix $\mathbf{M}^e$ into several parts. By shaded area in figure 2.6 we denote the components of the matrix $\mathbf{M}^e$ that have to be computed. The rest of the values of $\mathbf{M}^e$ are obtained by appropriate mapping of the computed entries. In the following, for convenience of notation, we drop the superscript e.

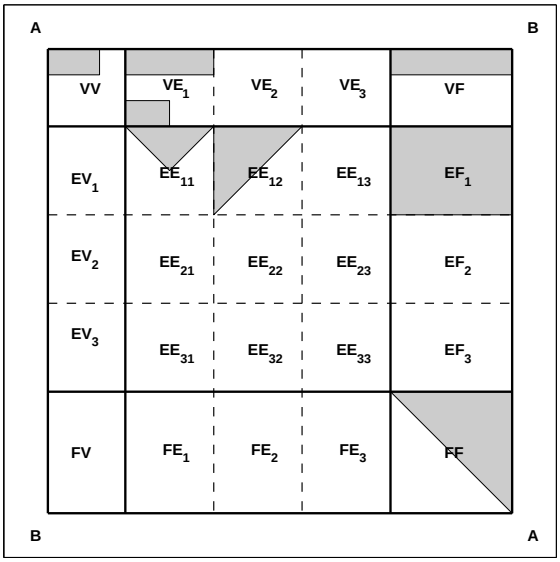


Figure 2.6: Schematic partitioning of a Mass matrix. P = 5. The shaded area represents the location of a matrix entries that have to be computed for Ψ-type bases.

Part 1 - *Vertex-Vertex Block.*

This section of the operator has nine terms, but due to the symmetry of the expansion bases only two of them have to be computed: $M_{i,j} = M_{j,i} = C_1$ and $M_{i,i} = C_2$ for all i, where C_1 and C_2 are different constants.

Part 2 - *Vertex-Edge.*

Since triangular element has 3 Edges we consider 3 blocks of the Vertex-Edge section, each of them corresponds to a different Edge. Each block has 3 rows and (P-1) columns. The Vertex-Edge product for the first Edge is obtained from

$$\int L_i L_1 L_2 P_{p-1}^{\alpha,\beta}(2L_1 - 1) P_{q-1}^{\alpha,\beta}(2L_2 - 1) dA, \quad (p + q = P), \tag{2.16}$$

where index $i = 1, 2, 3$ denotes the Vertex mode and the row of the Mass Matrix as well. We recall that index p runs from 1 to $P - 1$, and $q = P - p$. It can be easily shown that $VE_{1\ 2,j} = VE_{1\ 1,P-j}$ and $VE_{1\ 3,j} = VE_{1\ 3,P-j}$. Thus we have to compute only half of the values in this block. It turns out that due to the symmetry of the Edge modes the second and the third blocks should not be computed at all!, instead we can map the values at each block using: $VE_{1\ 1,j} = VE_{2\ 2,j} = VE_{3\ 3,j}$, $VE_{1\ 2,j} = VE_{2\ 3,j} = VE_{3\ 1,j}$, $VE_{1\ 3,j} = VE_{2\ 1,j} = VE_{3\ 2,j}$.

Part 3 - *Edge-Edge.*

This section of the operator has nine $(P-1)$ by $(P-1)$ blocks. First, due to the symmetry of the operator, we have $M_{i,j} = M_{j,i}$. Second, the diagonal blocks, that represent a product of Edge i with Edge i are identical for all i. Third, each of these diagonal blocks is symmetric not only with respect to its main diagonal $(A-A)$ but also with respect to the diagonal $B-B$ (see Fig. 2.6). Forth, block constructed as a product of the Edges 1 and 2 (EE_{12}) is also symmetric and a block constructed as a product of the Edges 1 and 3 (EE_{13}) is simply a transpose of a EE_{12}. Fifth, the Edge-Edge section is symmetric with respect to the $B-B$ diagonal, thus block EE_{23} is a mirror of EE_{12}.

Part 4 - *Vertex-Interior, Edge - Interior.*

The rotational symmetry of the expansion bases can be used to reduce the computational cost of evaluating this part as well. The recipe of the mapping is similar to the one provided in the next part.

Part 5 - *Edge - Interior.*

This part consists of 3 blocks, each has $(P-1)$ rows and $(P-1)(P-2)/2$ columns. Due to the rotational symmetry of the expansion function only one of this blocks should be computed, the other two are mapped from the first one.

To demonstrate the mapping procedure we consider $P = 5$, then the interior modes, which are products of the Jacobi polynomials of order $p-1$, $q-1$ and $r-1$ (see section 2.4.2) are ordered according to the following table:

$$pqr1 = \begin{array}{cccc} index & p & q & r \\ 1 & 1 & 1 & 3 \\ 2 & 1 & 2 & 2 \\ 3 & 1 & 3 & 1 \\ 4 & 2 & 1 & 2 \\ 5 & 2 & 2 & 1 \\ 6 & 3 & 1 & 1 \end{array} \tag{2.17}$$

The *index* in (2.17) provides the numaration of the interior shape function and points to the column number in the EF_1 block as well. In order to fill the next, EF_2 block, the table

given by (2.17) should be sorted with respect to the column q and then r to obtain:

$$pqr2 = \begin{array}{cccc} index & p & q & r \\ 6 & 3 & 1 & 1 \\ 4 & 2 & 1 & 2 \\ 1 & 1 & 1 & 3 \\ 5 & 2 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 3 & 1 & 3 & 1 \end{array} \tag{2.18}$$

Now assume that j is a row number in $pqr2$, then for each $j = 1, 2, ..., (P-1)(P-2)/2$ we have $EF_2(i, pqr2(j, 1)) = EF_1(i, j)$, where i is an index of row in the $Edge - Interior$ blocks and runs from $i = 1$ to $i = (P-1)$. For example the column 1 in the EF_2 block is identical to the column 6 of the EF_1 block and the column 2 in the EF_2 block is identical to the column 4 of the EF_1 block.

To fill the third block, EF_3, we sort (2.17) first by the column r and then by the column p to obtain

$$pqr3 = \begin{array}{cccc} index & p & q & r \\ 3 & 1 & 3 & 1 \\ 5 & 2 & 2 & 1 \\ 6 & 3 & 1 & 1 \\ 2 & 1 & 2 & 2 \\ 4 & 2 & 1 & 2 \\ 1 & 1 & 1 & 3 \end{array} \tag{2.19}$$

and, using the same notation for i and j as before, fill the third block according to $EF_3(i, pqr3(j, 1)) = EF_1(i, j)$.

As an example, lets consider $\mathbf{M}$, corresponding to $P = 5$. The block EF_1 is

```
[ -1/11880,  -19/13860,  -7/3960,     1/3780,    -1/1540,   1/4620]
[  -1/6930,  -59/23100,   -1/420,  -61/69300, -199/69300,  -1/2310]
[  -1/6930,  -61/69300,  -1/2310,  -59/23100, -199/69300,   -1/420]
```

```
[ -1/11880,     1/3780,   1/4620,  -19/13860,    -1/1540,  -7/3960],
```

and the block EF_2 is

```
[  -7/3960,    -1/1540,   1/4620,  -19/13860,     1/3780, -1/11880]
[   -1/420, -199/69300,  -1/2310,  -59/23100,  -61/69300,  -1/6930]
[  -1/2310, -199/69300,   -1/420,  -61/69300,  -59/23100,  -1/6930]
[   1/4620,    -1/1540,  -7/3960,     1/3780,  -19/13860, -1/11880].
```

Now consider that index $j = 1$, then the column 6 ($pqr2(1,1) = 6$) of EF_2 is identical to the column 1 of EF_1; similarly column 4 of EF_2 ($pqr2(j = 2, 1) = 4$) is the same as the column 2 of EF_1.

Part 6 - *Interior-Interior.*
Here the symmetry of the operator with respect to the AA diagonal can be employed to reduce the computational cost.

The evaluation of local operator in each elements can be substituted by a simple procedure. First, standard Mass matrix for an element with a unit area is computed. Then the standard operator is scaled by the area of each particular element. For Ψ-type bases, swapping of columns and rows corresponding to the edges that have to be reversed to complete the task. For Φ-type bases the procedure is completed by negating the columns and rows corresponding to the odd modes of the reversed edges. According to suggested procedure, for a time depended problems, where the computational mesh may deform, local Mass matrix is never recomputed, but it's scaled by the new versus old area ratio.

2.5.2 Construction of a Laplacian operator

The construction of Laplacian operator $\mathbf{L}^e$ for different expansion bases (Λ_i) requires evaluation of the inner product of the derivatives of modes in the expansion basis with themselves, i.e.,

$$L^e_{i,j} = \left(\frac{\partial \lambda_i}{\partial x}, \frac{\partial \lambda_j}{\partial x}\right) + \left(\frac{\partial \lambda_i}{\partial y}, \frac{\partial \lambda_j}{\partial y}\right).$$

Construction of a local to each element Laplacian matrix $\mathbf{L}^e$ is more complicated than construction of a Mass matrix, since it involves derivatives of the mapping functions (2.4, 2.3). The last are varying from element to element, what suggests that $\mathbf{L}^e$ should be computed in each element. For the straight sided linear triangular elements the transformation

from xy coordinates into $\xi_1\xi_2$ (or L_1, L_2, L_3) is linear, thus the derivatives of the mapping functions are constants. In the following we overview a computationally efficient routine to compute $\mathbf{L}^e$. The idea is to compute only three standard matrices (defined in a local to element coordinate system) and use the linear combination of them in order to obtain $\mathbf{L}^e$ matrix for an arbitrary triangular element. The coefficients of the linear combinations are the derivatives of the elemental mapping functions scaled by the area of an element. To guarantee the global C^0 connectivity, the appropriate modes of the Edge functions are negated - in a case of Φ-type bases or swapped in a case of Ψ-type bases. Suggested methodology reduces the computational effort in computing the $\mathbf{L}^e$ for each element, particularly for a time-dependent problems where the computational mesh is deformed or a new computational mesh should be generated.

To clarify the suggested procedure we provide the following example. The $\mathbf{L}_{i,j}$ component of the Laplacian matrix, based on the Ψ-type bases is computed from

$$\mathbf{L}^e_{i,j} = \mathbf{Lx}^e_{i,j} + \mathbf{Ly}^e_{i,j} = \int \frac{\partial \Psi_i}{\partial x}\frac{\partial \Psi_j}{\partial x}dxdy + \int \frac{\partial \Psi_i}{\partial y}\frac{\partial \Psi_j}{\partial x}dxdy. \tag{2.20}$$

$$\mathbf{Lx}^e_{i,j} = \left(\frac{\partial L_2}{\partial x}\right)^2 2A'\mathbf{LS1}_{i,j} + \left(\frac{\partial L_3}{\partial x}\right)^2 2A'\mathbf{LS2}_{i,j} + \frac{\partial L_2}{\partial x}\frac{\partial L_3}{\partial x}2A'\mathbf{LS3}_{i,j} \tag{2.21}$$

$$\mathbf{Ly}^e_{i,j} = \left(\frac{\partial L_2}{\partial y}\right)^2 2A'\mathbf{LS1}_{i,j} + \left(\frac{\partial L_3}{\partial y}\right)^2 2A'\mathbf{LS2}_{i,j} + \frac{\partial L_2}{\partial y}\frac{\partial L_3}{\partial y}2A'\mathbf{LS3}_{i,j}, \tag{2.22}$$

where A' is an area of the element, and

$$\begin{aligned}
\mathbf{LS1}_{i,j} &= \int \left(\frac{\partial \Psi_i}{\partial L_2}\frac{\partial \Psi_j}{\partial L_2}\right) dL_1 dL_2 \\
\mathbf{LS2}_{i,j} &= \int \left(\frac{\partial \Psi_i}{\partial L_3}\frac{\partial \Psi_j}{\partial L_3}\right) dL_1 dL_2 \\
\mathbf{LS3}_{i,j} &= \int \left(\frac{\partial \Psi_i}{\partial L_2}\frac{\partial \Psi_j}{\partial L_3} + \frac{\partial \Psi_i}{\partial \text{L}_3}\frac{\partial \Psi_j}{\partial \text{L}_2}\right) dL_1 dL_2;
\end{aligned}$$

or, alternatively:

$$\begin{aligned}
\mathbf{LS1}_{i,j} &= \int \left(\frac{1-\eta_1}{4}\frac{\partial \Psi_i}{\partial \eta_1} + \frac{1+\eta_2}{4}\frac{\partial \Psi_i}{\partial \eta_2}\right)\left(\frac{1-\eta_1}{4}\frac{\partial \Psi_j}{\partial \eta_1} + \frac{1+\eta_2}{4}\frac{\partial \Psi_j}{\partial \eta_2}\right)\frac{1}{J}d\eta_1 d\eta_2 \\
\mathbf{LS2}_{i,j} &= \int \frac{1}{4}\frac{\partial \Psi_i}{\partial \eta_2}\frac{\partial \Psi_j}{\partial \eta_2}\frac{1}{J}d\eta_1 d\eta_2 \\
\mathbf{LS3}_{i,j} &= \int \left[\frac{1-\eta_1}{8}\left(\frac{\partial \Psi_i}{\partial \eta_1}\frac{\partial \Psi_j}{\partial \eta_2} + \frac{\partial \Psi_j}{\partial \eta_1}\frac{\partial \Psi_i}{\partial \eta_2}\right)\frac{1+\eta_2}{4}\left(\frac{\partial \Psi_i}{\partial \eta_2}\frac{\partial \Psi_j}{\partial \eta_2}\right)\right]\frac{1}{J}d\eta_1 d\eta_2
\end{aligned}$$

Thus, **LS1**, **LS2** and **LS3** are the three standard matrices that must be computed. Then, *all* local Laplacian operators of straight sided elements are constructed from these matrices. We recall that additional work can be required in order to complete the construction of the local operator, namely: negating the odd modes of reversed Edges for the cartesian tensorial bases or swapping the columns and rows corresponding to the reversed Edges for barycentric tensorial bases.

2.5.3 Construction of Advection operator

In construction of Advection operator $\mathbf{D}^e$ we can use the same concept as in construction of the Laplacian matrix. But this time we need only two standard matrices in order to obtain an elemental Advection Matrix. To illustrate this approach we formulate the procedure to compute $\mathbf{D}^e_{i,j} = U\mathbf{Dx}^e_{i,j} + V\mathbf{Dy}^e_{i,j}$ in the following way:

$$U\mathbf{Dx}^e_{i,j} + V\mathbf{Dy}^e_{i,j} = U\left(\lambda_i, \frac{\partial \lambda_j}{\partial x}\right) + V\left(\lambda_i, \frac{\partial \lambda_j}{\partial y}\right). \tag{2.23}$$

Using transformation of coordinate system from xy to L_1, L_2, L_3 (or $\eta_1\eta_2$) we obtain $\mathbf{Dx}^e_{i,j}$ and $\mathbf{Dy}^e_{i,j}$ from

$$\mathbf{Dx}^e_{i,j} = \frac{\partial L_2}{\partial x} 2A'\mathbf{S1}_{i,j} + \frac{\partial L_3}{\partial x} 2A'\mathbf{S2}_{i,j} \tag{2.24}$$

$$\mathbf{Dy}^e_{i,j} = \frac{\partial L_2}{\partial y} 2A'\mathbf{S1}_{i,j} + \frac{\partial L_3}{\partial y} 2A'\mathbf{S2}_{i,j}, \tag{2.25}$$

where

$$\begin{aligned} \mathbf{S1}_{i,j} &= \int \Psi_i \frac{\partial \Psi_j}{\partial L_2} dL_1 dL_2 \\ \mathbf{S2}_{i,j} &= \int \Psi_i \frac{\partial \Psi_j}{\partial L_3} dL_1 dL_2; \end{aligned}$$

or, equivalently:

$$\begin{aligned} \mathbf{S1}_{i,j} &= \int \Psi_i \left[\frac{1-\eta_1}{4}\frac{\partial \Psi_j}{\partial \eta_1} + \frac{1+\eta_2}{4}\frac{\partial \Psi_j}{\partial \eta_2}\right]\frac{1}{J} d\eta_1 d\eta_2 \\ \mathbf{S2}_{i,j} &= \int \Psi_i \frac{1}{2}\frac{\partial \Psi_j}{\partial \eta_2}\frac{1}{J} d\eta_1 d\eta_2. \end{aligned}$$

Thus the Advection operator for each element is constructed as a linear combination of the standard matrices **S1** and **S2** multiplied by the gradients of the mapping functions and an area of the element.

2.5.4 Construction of global operator

In the previous section construction of a *local* (elemental) operator was presented. The *global* operator $\mathbf{A}$ is assembled as a linear combination of local operators $\mathbf{A}^e$ by a technique known as *static condensation*:

$$\mathbf{A} = \sum_{e=1}^{Nel} \mathcal{P}^e(\mathbf{A}^e),$$

here $\mathcal{P}^e$ is a projection operator [79], which maps the local degrees of freedom to the global. The operator $\mathbf{A}^e$ can be any of the three operators discusses in the previous section or a linear combination of those. We therefore assume that we have a system of the form

$$\mathbf{A}\hat{\mathbf{u}} = \hat{\mathbf{f}},$$

where $\hat{\mathbf{u}}$ is a vector of unknowns and $\hat{\mathbf{f}} = \sum_{e=1}^{Nel} \mathcal{P}^e(\mathbf{f^e})$ is a forcing term. We can further distinguish between the boundary and interior degrees of freedom and split the operator $\mathbf{A}$ into

$$\begin{bmatrix} \mathbf{A}_{bb} & \mathbf{A}_{bi} \\ \mathbf{A}_{bi}^T & \mathbf{A}_{ii} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}^b \\ \hat{\mathbf{u}}_i \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{f}}_b \\ \hat{\mathbf{f}}_i \end{bmatrix}. \tag{2.26}$$

The block $\mathbf{A}_{bb}$ represents the boundary-boundary coupling, $\mathbf{A}_{bi}$ represents the boundary-interior coupling, and $\mathbf{A}_{ii}$ represents the interior-interior coupling. Since the interior modes have zero support on the boundaries of each element, the $\mathbf{A}_{ii}$ consist of smaller interior-interior *non-overlapping* matrices $\mathbf{A}_{ii}^e$. To solve system (2.26) we multiply it (from left) by the matrix

$$\begin{bmatrix} \mathbf{I} & -\mathbf{A}_{bi}\mathbf{A}_{ii}^{-1} \\ 0 & \mathbf{I} \end{bmatrix}, \tag{2.27}$$

to arrive at

$$\begin{bmatrix} \mathbf{A}_{bb} - \mathbf{A}_{bi}\mathbf{A}_{ii}^{-1}\mathbf{A}_{bi}^T & 0 \\ \mathbf{A}_{bi}^T & \mathbf{A}_{ii} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_b \\ \hat{\mathbf{u}}_i \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{f}}_b - \mathbf{A}_{bi}\mathbf{A}_{ii}^{-1}\hat{\mathbf{f}}_i \\ \hat{\mathbf{f}}_i \end{bmatrix}. \tag{2.28}$$

Solution of system (2.28) can be performed in two steps. At the first step, the equation for boundary modes is solved

$$\mathbf{A}_{sc}\hat{\mathbf{u}}_b = \left[\mathbf{A}_{bb} - \mathbf{A}_{bi}\mathbf{A}_{ii}^{-1}\mathbf{A}_{bi}^T\right]\hat{\mathbf{u}}_b = \hat{\mathbf{f}}_b - \mathbf{A}_{bi}\mathbf{A}_{ii}^{-1}\hat{\mathbf{f}}_i. \tag{2.29}$$

At the second step a series of small problems

$$\hat{\mathbf{u}}_i = \mathbf{A}_{ii}^{-1}(\hat{\mathbf{f}}_i - \mathbf{A}_{bi}^T\hat{\mathbf{u}}_b)$$

is solved. A system (2.29) can be solved directly or iteratively. The operator $\mathbf{A}_{sc}$ (called the *Shur complement*) is usually very large and sparse, what makes the iterative solution to become a method of choice.

2.6 Numerical properties of linear operators: Computational efficiency

In this section numerical aspects such as sparsity and eigenproperties of the linear operators constructed from the cartesian and barycentric bases are studied. First, the accuracy in solution of a projection and diffusion problems is verified. Second, the sparsity of the Projection and Laplacian operators are compared. Third, the eigenspectra and efficiency of iterative preconditioned conjugate solver in conjunction with different discretization is examined.

2.6.1 Accuracy verification

In order to verify accuracy of numerical solvers, based on both types of the expansion bases, the Projection and Diffusion problems defined in a compact form as $\mathbf{M}\hat{\mathbf{u}} = \hat{\mathbf{f}}_\mathbf{1}$ and $\mathbf{L}\hat{\mathbf{u}} = \hat{\mathbf{f}}_\mathbf{2}$, have been solved; here $\mathbf{u}$ are unknown coefficients of a spectral expansion, while $\mathbf{f}_1$ and $\mathbf{f}_2$ are suitable forcing terms. The numerical solution was evaluated at quadrature points, and compared to an exact one. As an exact solution we used $v(x, y) = \sin(\pi x)\sin(\pi y), \quad 0 \leq x, y \leq 2$. The numerical error was estimated using the following norms:

$$L_2 = \sqrt{\int_\Omega (u(x, y) - v(x, y))^2 dA}$$

$$L_\infty = MAX_\Omega |u(x, y) - v(x, y)|$$

The computational domain $\Omega = \{x, y | 0 \leq x, y \leq 2\}$ has been subdivided to 4, 8 and 16 triangular elements as presented in figure 2.7. The L_2 and L_∞ errors of a numerical solution, of a projection and diffusion problems, are presented in figures 2.8, 2.9. As expected the numerical error does not depend on a choice of an expansion bases, and spectral convergence is obtained in all cases.

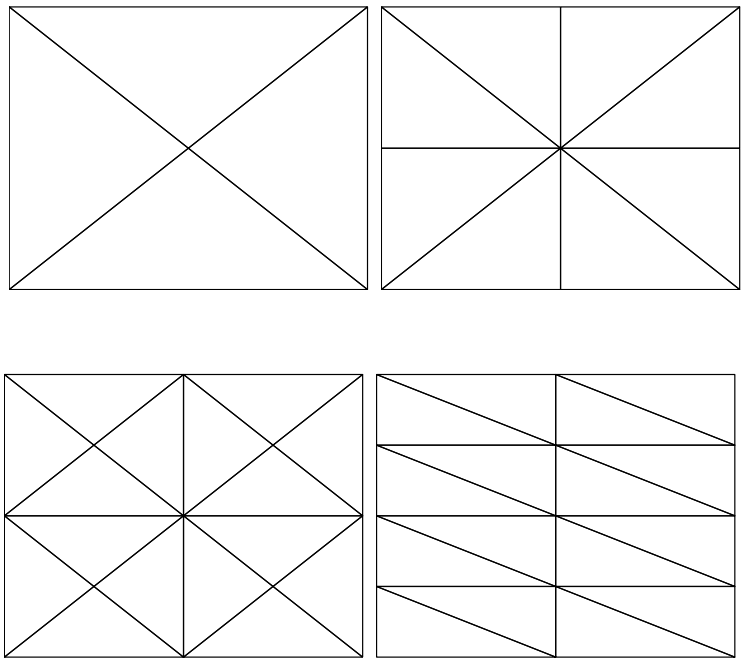


Figure 2.7: Triangulation of a computational domain Ω. Left top: four element mesh (mesh A). Right top: eight element mesh (mesh B). Left bottom: 16 element mesh (mesh C). Right bottom: 16 element mesh (mesh D).

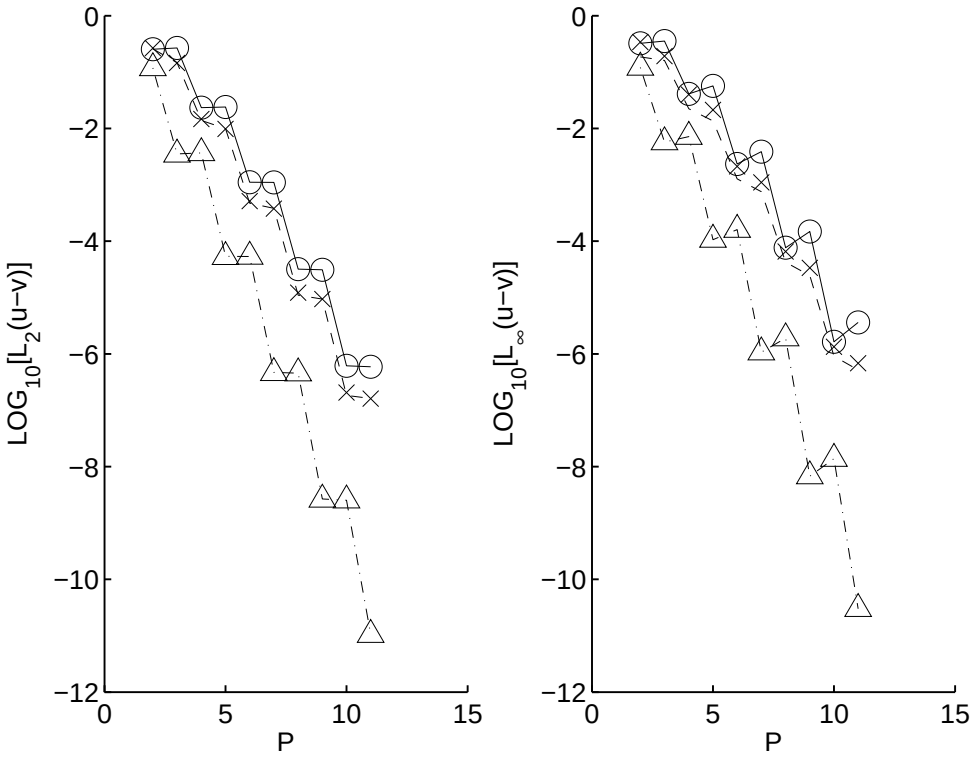


Figure 2.8: The L_2 and L_∞ errors in a solution of projection problem. Solid line - Mesh A, dash line - Mesh B, dash-dot line Mesh C. Lines correspond to $\mathbf{\Phi}$-type bases, markers to $\mathbf{\Psi}$-type bases.

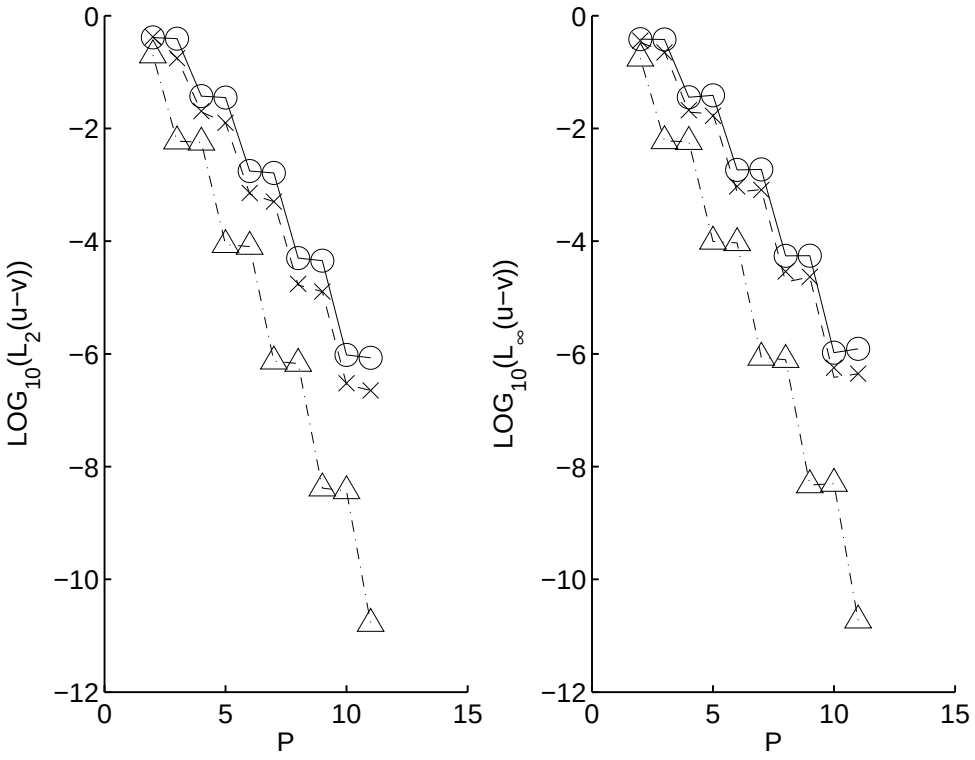


Figure 2.9: The L_2 and L_∞ errors in a solution of Diffusion problem. Solid line - Mesh A, dash line - Mesh B, dash-dot line Mesh C. Lines correspond to $\mathbf{\Phi}$-type bases, markers to $\mathbf{\Psi}$-type bases.

2.6.2 Sparsity of linear operators

In figures 2.10 the sparsity of components of global Mass and Laplacian operators corresponding to Cartesian and Barycentric bases is compared; the data corresponds to 24 elements mesh. The cartesian bases posses better sparsity patterns than the barycentric bases, particularly for higher-order polynomial expansion. Sparser operator requires less floating point operations potentially leading to computational savings.

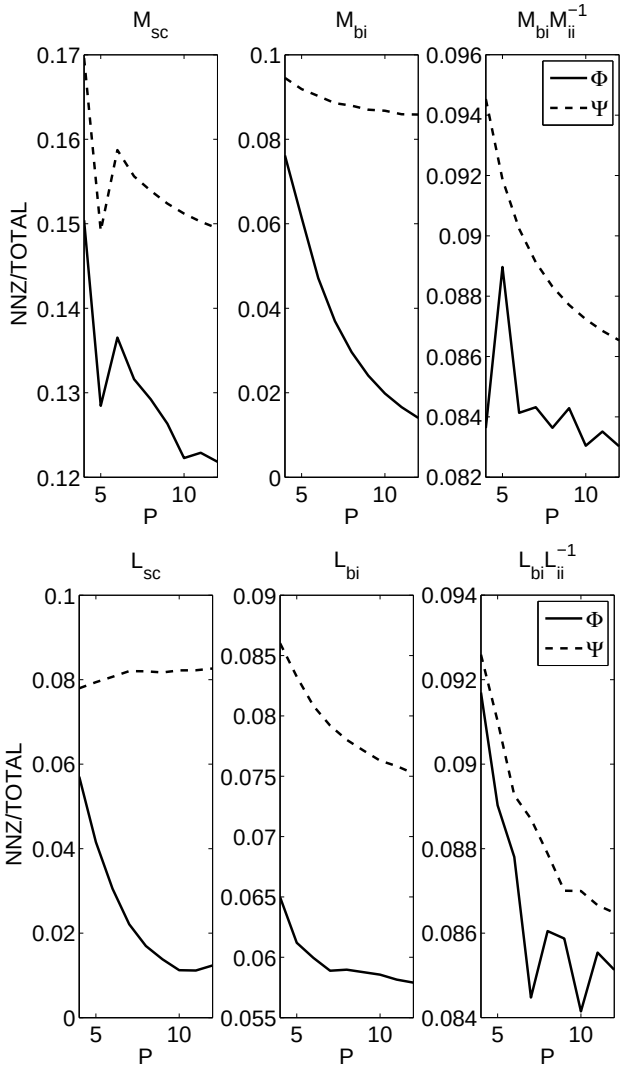


Figure 2.10: Components of Mass (top) and Laplacian (bottom) matrix: sparsity. NNZ/TOTAL - ratio of non-zero to total number of matrix entries.

2.6.3 Eigenproperties of linear operators

Mass Matrix

We start from presenting the maximal and minimal eigenvalues, λ, and the condition number, κ, of the unpreconditioned and preconditioned statically condensed Mass matrix for both types of bases. In figures 2.11 and 2.12 we compare the properties of the condensed

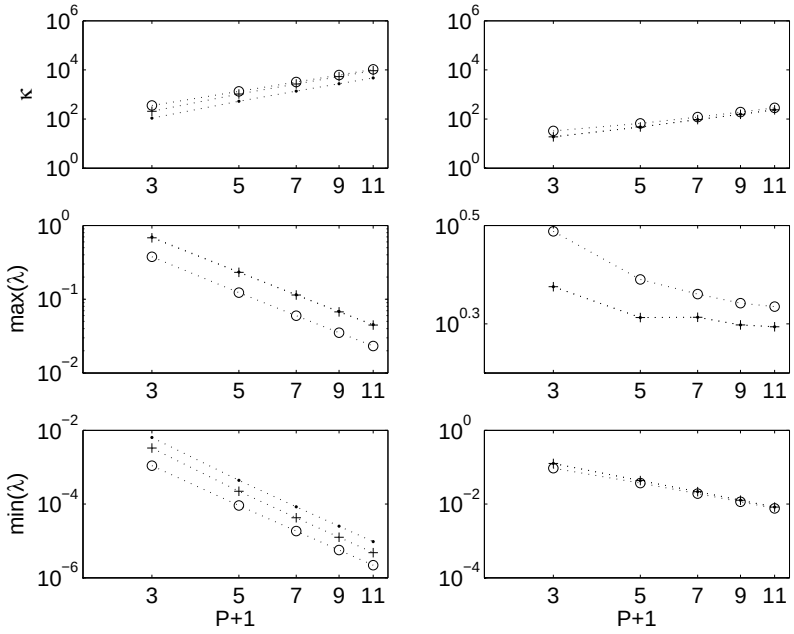


Figure 2.11: Spectral analysis of Schur complement of condensed Mass matrix. **Φ**- bases. Unpreconditioned (left) and Diagonally preconditioned (right) condensed Mass matrix. Results for meshes A,B and C are marked by • , + and *o* correspondingly.

Mass matrix based on different spectral bases. We observe the superiority of the **Φ**-type bases over **Ψ**-type bases in a conditioning and in a response to the diagonal preconditioning. We recall that the condensed Mass matrix based on the **Φ**-type bases is very sparse and most of its "mass" is concentrated on the diagonal, whereas the Mass matrix based on the **Ψ**-type bases is a full matrix. Thus the diagonal preconditioner of the $\mathbf{M}(\mathbf{\Psi})$ matrix is not as spectrally close as in a case of $\mathbf{M}(\mathbf{\Phi})$. As a result one of mentioned above requirements for the preconditioner is not held. The practical meaning of high condition number is a larger number of iterations needed in order to converge to an exact (or reasonable) solution.

Next we consider the Incomplete Cholesky Preconditioner (ICP). In order to evaluate the effectiveness of ICP, in conjunctions with the two bases, the projection problem defined

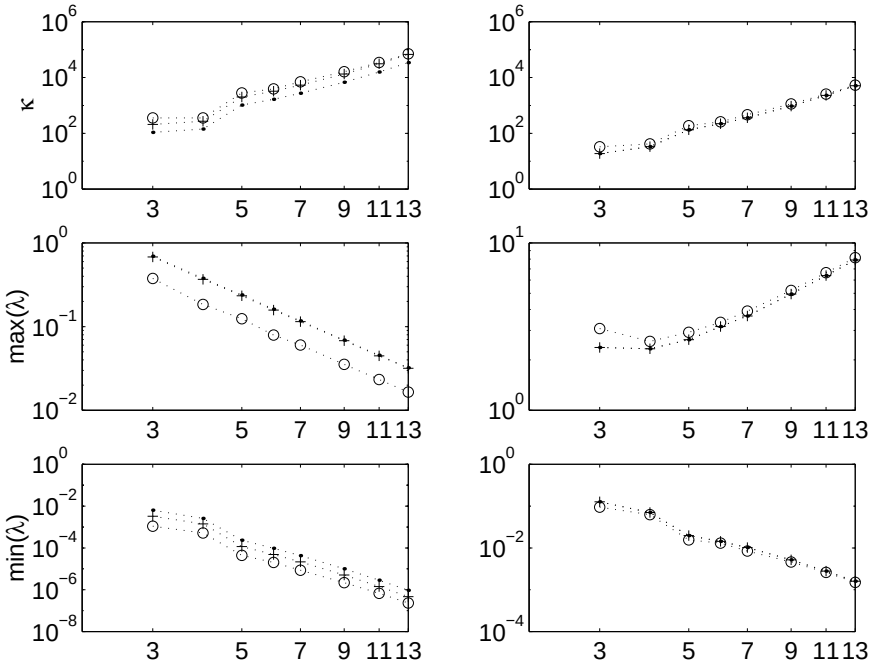


Figure 2.12: Spectral analysis of Schur complement of condensed Mass matrix. $\mathbf{\Psi}$- bases. Unpreconditioned (left) and Diagonally preconditioned (right) condensed Mass matrix. Results for meshes A,B and C are marked by $\bullet$, $+$ and o correspondingly.

on a domain Ω discretized into 16 uniform elements (figure 2.7(right bottom)) was solved. The size of the domain in $y-$ direction was repeatedly changed to reveal the influence of distortion of elements on the computational cost. We define a distortion parameter R as a ratio of the length of catheti of a triangular elements.

The effectiveness of ICP in terms of reduction in iteration count is presented in figures 2.13. We observe considerable reduction in number of iterations required, particularely in the case of $\mathbf{\Psi}$ bases. Specifically, we observe:

- The number of iterations and hence (the eigenspectrum of the condensed Schur complement of Mass matrix) is unaffected by distortion. This result is expected since all elements in the mesh have the same area, that is the same Jacobian, which scales both minimum and maximum eigenvalues so their ratio remains unchanged.

- The DP is more effective when applied on $\mathbf{M}_{sc}(\mathbf{\Phi})$.

- The ICP is more efficient than DP in both bases. Similar or slightly better efficiency can be obtained by applying ICP on a linear system based on the barycentric bases than on the cartesian bases.

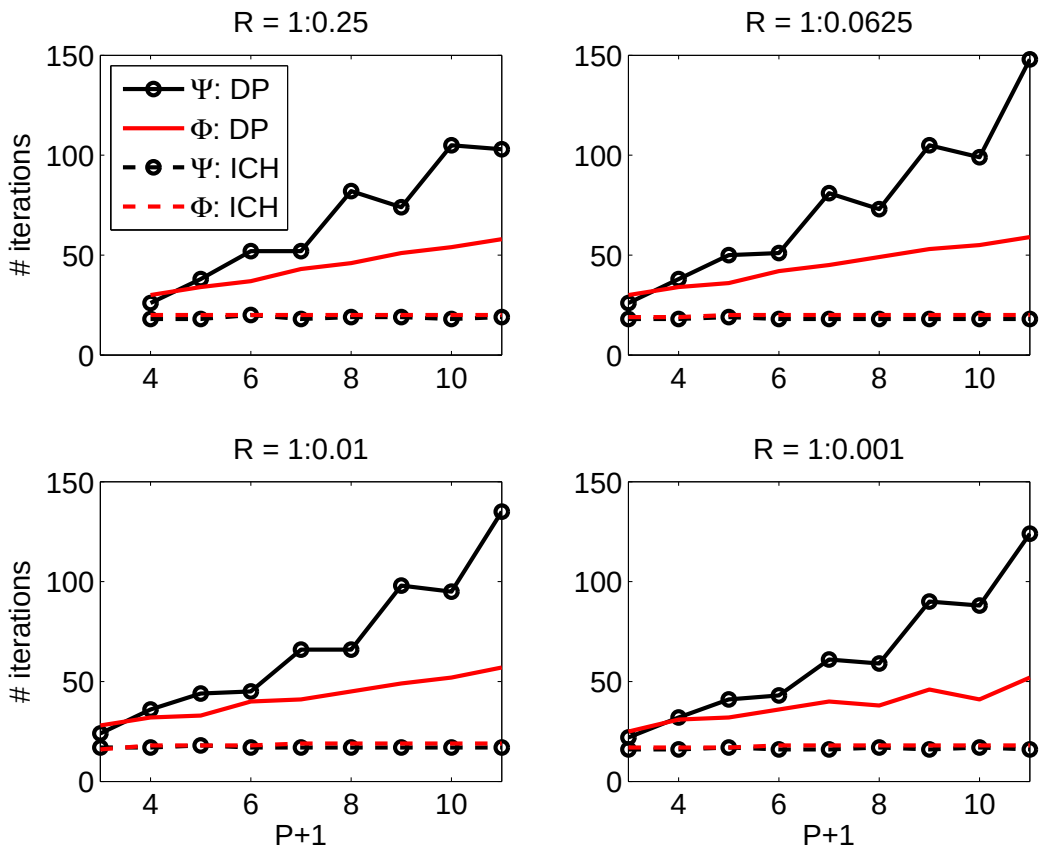


Figure 2.13: Solution of projection problem: Effectiveness of the Incomplete Cholesky Preconditioner (ICP) and Diagonal Preconditioner (DP). Solid line - DP, dash line - ICP. Circles mark data for the **Ψ**- type bases.

Laplacian operator

The eigenproperties of the statically condensed Laplacian operator (**L**) are presented in figure 2.14; the condition number, maximal and minimal eigenvalues of the unpreconditioned and preconditioned statically condensed matrix for both types of bases are compared. We observe that the condition number of **L**(**Ψ**) is considerably higher than of **L**(**Φ**). Moreover the diagonal preconditioner is noticeably more effective for the operator based on **Φ**- type bases.

Next, we a apply the Incomplete Cholesky Prerconditioner for iterative solution of diffusion problem. We monitor the number of iteration required by the PCGM which in part reflects the effectiveness of the preconditioning. We are also interested in the influence of the mesh deformation on the eigenproperties of the operators. The domain discretization is shown in figure 2.7(bottom right). As in a previous study the domain was repeatedly

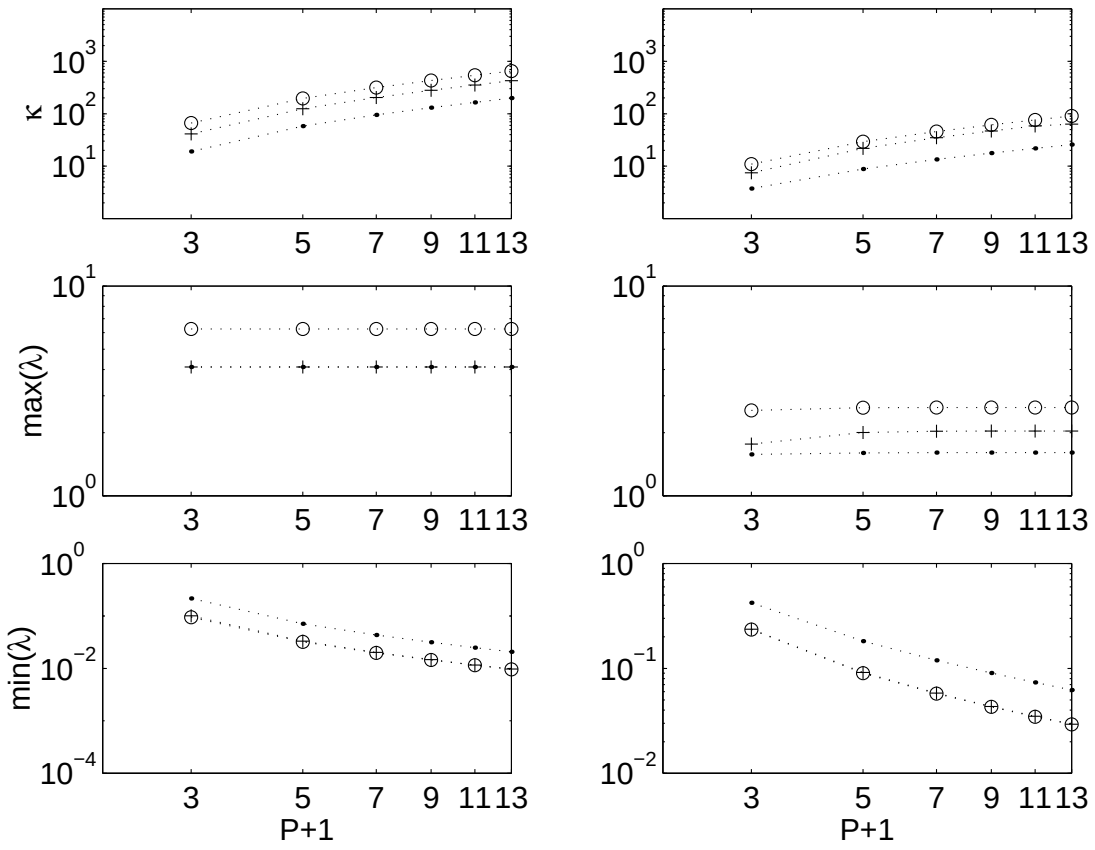


Figure 2.14: Spectral analysis of the unpreconditioned (left) and Diagonally preconditioned (right) condensed Schur complement of Laplacian matrix. Φ-type bases. Results for meshes A,B and C are marked by • , + and *o* correspondingly.

reduced in size in y- direction. Results are summarized in figure 2.16. Again, as in a case of the Mass matrix, we observe the benefits of using the ICP. In contrast to the Mass matrix, the Laplacian operator shows high sensitivity to the mesh distortion, since it incorporates the values of the derivatives of mapping functions. In the view of results presented in 2.16 we may conclude that the use of symmetric expansion bases can be advantageous, particularly for a very distorted mesh. However, we should not ignore the high computational cost in simulations with ICP and $\mathbf{\Psi}$ bases due to different sparsity patterns. It was observed that approximately same CPU-time is required for solution of diffusion problem defined on domain consisting from elements with large aspect ratio when the diagonal preconditioner is employed for $\mathbf{L}(\mathbf{\Phi})$ and ICP for $\mathbf{L}(\mathbf{\Psi})$.

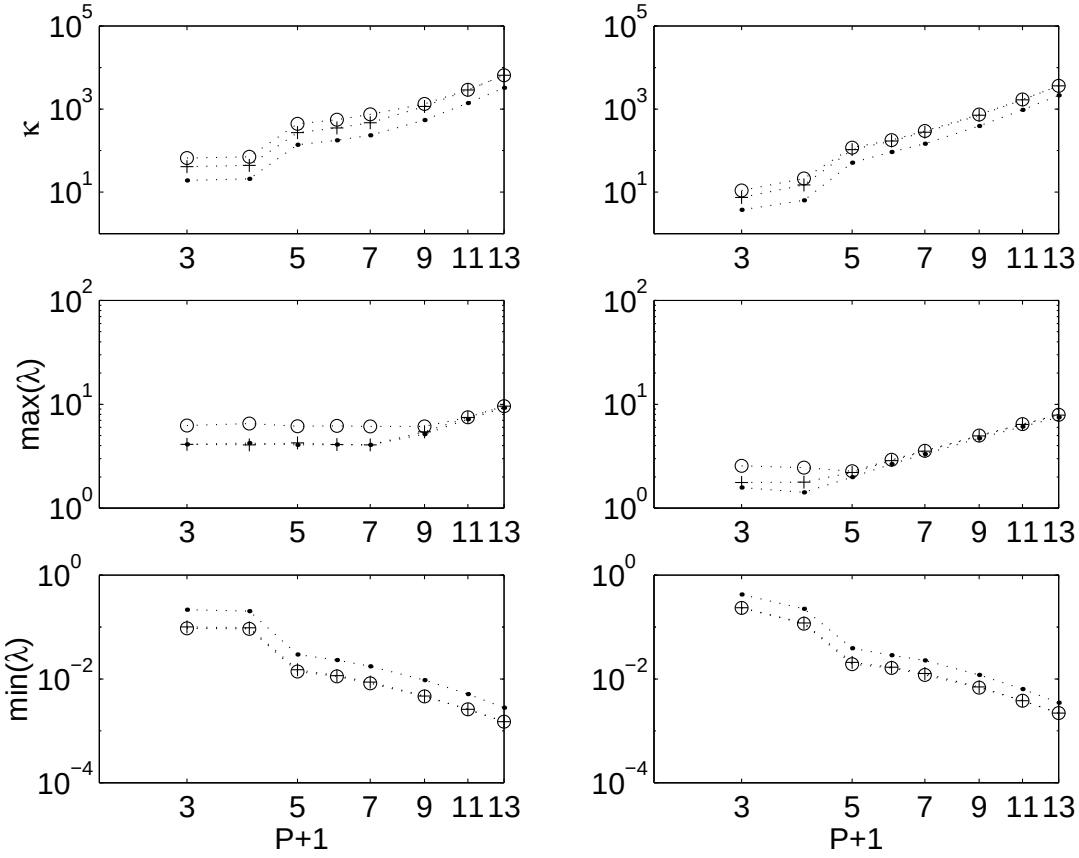


Figure 2.15: Spectral analysis of the unpreconditioned (left) and Diagonally preconditioned (right) condensed Schur complement of Laplacian matrix. Ψ-type bases. Results for meshes A,B and C are marked by • , + and *o* correspondingly.

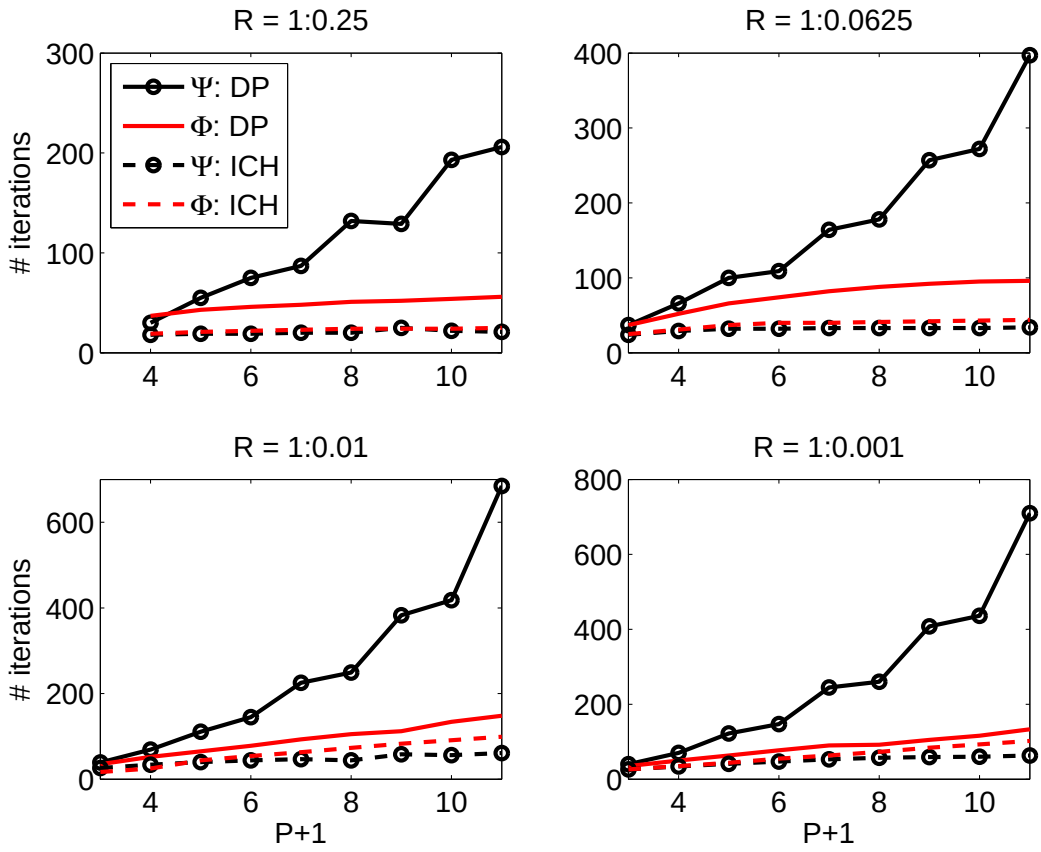


Figure 2.16: Solution of diffusion problem: Effectiveness of the Incomplete Cholesky Preconditioner (ICP) and Diagonal Preconditioner (DP). Solid line - DP, dash line - ICP. Circles mark data for the $\mathbf{\Psi}$- type bases.

Advection operator

In this section we solve the linear advection problem

$$\frac{\partial u}{\partial t} = U\frac{\partial u}{\partial x} + V\frac{\partial u}{\partial y} + f \tag{2.30}$$

which, in a compact form, reads

$$\frac{\partial}{\partial t}\hat{\mathbf{u}} = \mathbf{M}^{-1}(U\mathbf{D}\mathbf{x} + V\mathbf{D}\mathbf{y})\hat{\mathbf{u}} + \mathbf{M}^{-1}\int_{\Omega}\mathbf{\Lambda} f(t,x,y)dxdy, \tag{2.31}$$

where $\hat{\mathbf{u}}$ are the amplitudes of the expansion bases $\mathbf{\Lambda}$. We are interested in comparing spectral properties of the operator $\mathbf{M}^{-1}\mathbf{D} \equiv \mathbf{M}^{-1}(U\mathbf{D}\mathbf{x} + V\mathbf{D}\mathbf{y})$ with respect to different expansion bases. The spectral analysis of $\mathbf{M}^{-1}\mathbf{D}$ helps in selecting the appropriate time stepping scheme.

As an exact solution for the linear advection equation with periodic boundary condition we use

$$u = sin(\pi x + c_1 t)sin(\pi y + c_2 t), \tag{2.32}$$

where c_1 and c_2 are computed from the requirement $\sqrt{U^2 + V^2} = 1$

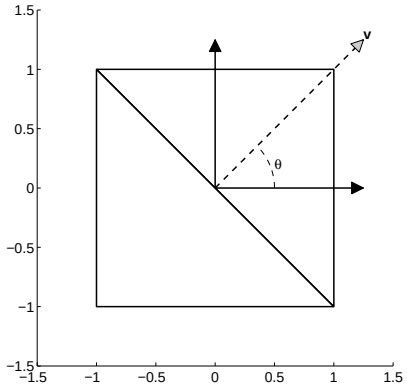


Figure 2.17: Two element domain for linear advection problem. Vector **v** shows the direction of the wave. The angle θ is computed from $tan^{-1}(V/U)$. The circles correspond to the $\mathbf{\Phi}$ bases, while the dots to the $\mathbf{\Psi}$ bases.

We start from a two element domain, as shown in figure 2.17. In figure 2.18 we present the maximal absolute value of the eigenvalues of $\mathbf{M}^{-1}\mathbf{D}$ for different combination of U and V (see Eq. 2.31) and $P = 7, 11$. It is clearly seen that the maximum value of $|\lambda(\mathbf{M}^{-1}\mathbf{D})|$ depends on a weights of its components, $\mathbf{D}\mathbf{x}$ and $\mathbf{D}\mathbf{y}$ or, in other worlds, on the wave course

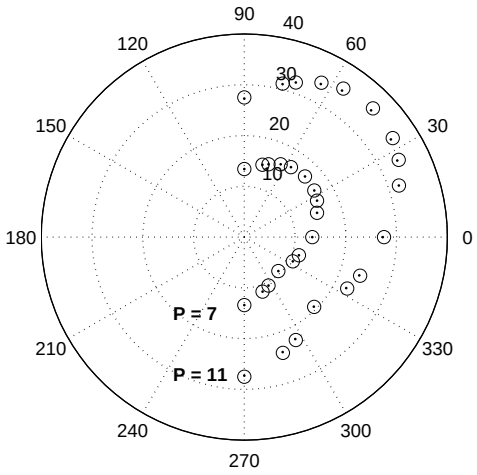


Figure 2.18: The maximal value of $|\lambda(\mathbf{M}^{-1}\mathbf{D})|$, $\theta = tan^{-1}(V/U)$.

with respect to the elements orientation. It is also noticeable that, considering the eigenproperties of the advection operator, there is no practical difference between implementing $\mathbf{\Phi}$- or $\mathbf{\Psi}$-type bases.

Next we investigate the influence of a distortion of elements composing the same domain Ω on the eigenproperties of the $\mathbf{M}^{-1}\mathbf{D}$ operator.

Consider a domain Ω with different discretization as shown in figure 2.19. The maximum eigenvalues of the operator $\mathbf{M}^{-1}\mathbf{D}$ for different values of θ are presented in figure 2.20. We observe that for both types of expansion bases we have practically the same maximum eigenvalues, thus we may conclude that the choice of the expansion bases does not affect the stability criteria for the linear advection operator. The numerical solution of (2.31) at $t = 1$ was compared to an exact one at arbitrary points $x_i, y_j \in \Omega$ which location was not affected by a domain discretization, the angle θ or the choice of an expansion bases. The $L_2 - error$ of the numerical solution is presented in figure 2.21. As we could expect the accuracy of the solution does not depend on the choice of the bases, since they span the same polynomial space.

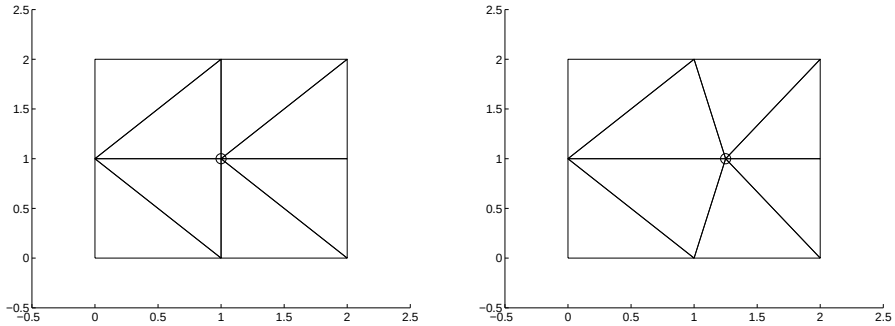
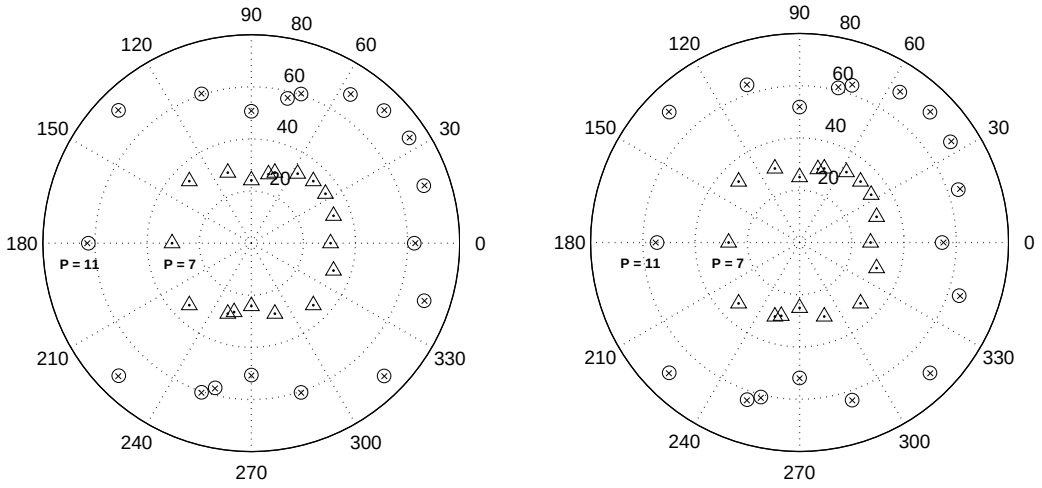
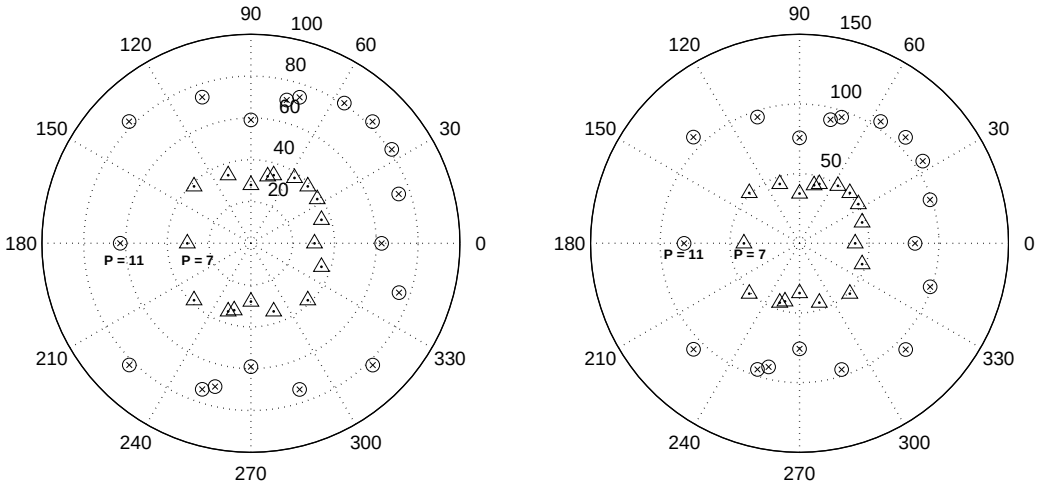


Figure 2.19: Discretization of computational domain Ω into eight triangular elements.

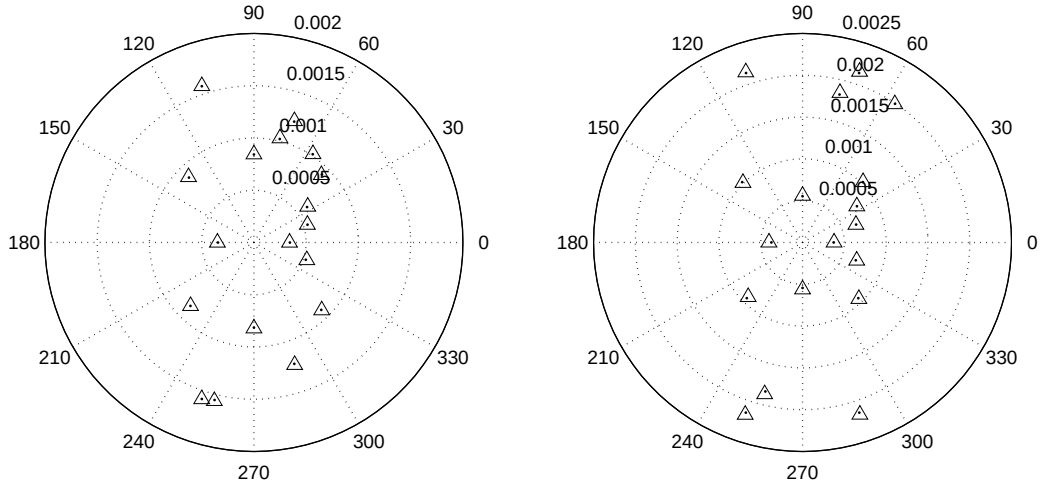


Mesh A’ Mesh B’



Mesh C’ Mesh D’

Figure 2.20: Maximum of $|\lambda(\mathbf{M}^{-1}\mathbf{D})|$. Data is presented in polar coordinate system, where the angle coordinate corresponds to wave direction θ. Data for $\mathbf{\Phi}$ bases is marked by circles and triangles, for $\mathbf{\Psi}$ bases by crosses and dots.



Mesh A' Mesh B'

Mesh C' Mesh D'

Figure 2.21: L_2-error of a numerical solution of (2.31). Data is presented in polar coordinate system, where the angle coordinate corresponds to wave direction θ. Data for $\mathbf{\Phi}$ bases is marked by circles and triangles, for $\mathbf{\Psi}$ bases by crosses and dots.

2.7 Solution of 2D Navier-Stokes problems with $P/P-k$ approach

Let us consider an incompressible flow, for which the governing equations are given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}, \tag{2.33a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.33b}$$

where $p(t, \mathbf{x})$ is the pressure, $\mathbf{u}(t, \mathbf{x})$ is the velocity field, ν is the kinematic viscosity, and it is assumed that the density is $\rho = 1$. The equations (2.33a,2.33b) are supplemented with appropriate boundary conditions. Here we consider $\mathbf{u} = 0$ on $\partial\Omega$, where $\partial\Omega$ defines the boundary of the domain. In a case of non-zero Dirichlet boundary condition we may "lift" the solution $\mathbf{u}$ by subtracting the know values for $\mathbf{u}$ at the boundaries to arrive to zero Dirichlet boundary conditions.

For solution of the Navier-Stokes (NS) equation we implement a) implicit-explicit scheme, and b) fully implicit scheme.

Implicit-explicit scheme: solution of the NS equations is performed with high-order time splitting scheme [48], that decouples the velocity and pressure fields. The non-linear term is integrated using high-order Adams-Bashforth integrator, while the viscous term is treated implicitly. The numerical scheme is based on the following discretization:

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k}}{\Delta t} = -\sum_{k=0}^{Je-1} \beta_k (\mathbf{u}^{n-k} \cdot \nabla)\mathbf{u}^{n-k} - \nabla p + \nu \nabla^2 \mathbf{u}^{n+1}. \tag{2.34a}$$

and the solution is performed in three stages:

Stage 1: auxiliary field $\mathbf{u}^*$ is computed explicitly using Adams-Bashforth integrator

$$\mathbf{u}^* = \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k} + \Delta t \left(\sum_{k=0}^{Je-1} \beta_k (\mathbf{NL})^{n-k} \right), \tag{2.34b}$$

where $\mathbf{NL} = (\mathbf{u} \cdot \nabla)\mathbf{u}$ is the non-linear term, γ_0 and α_k are coefficients of backward differentiation, β_k - coefficients of Adams-Bashforth integration schemes, Je is the integration (differentiation) order.

Stage 2: solution of Poisson equation for pressure

$$\nabla^2 p = (\Delta t)^{-1} \nabla \cdot \mathbf{u}^* \tag{2.34c}$$

supplemented with Dirichlet or Neumann boundary conditions. The Neumann boundary conditions are computed explicitly:

$$\frac{\partial p}{\partial n} = \left[-\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k}}{\Delta t} - \sum_{k=0}^{Je-1} \left(\beta_k [(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla \times \nabla \times \mathbf{u}]^{n-k} \right) \right] \cdot \hat{\mathbf{n}}. \tag{2.34d}$$

Stage 3: implicit solution of the Helmholtz problem

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\nabla p + \nu \nabla^2 \mathbf{u}^{n+1}. \tag{2.34e}$$

Note, that $O(\nu(\Delta t)^{Je})$ error is introduced in the Neumann pressure boundary conditions, the error appears due to explicit treatment of the viscous term.

Fully implicit scheme employs high-order backward differentiation for the time derivatives and has no splitting errors. However, it requires that the NS equations are solved repeatedly until convergence in numerical solution is obtained at each time step. Here we require convergence in the velocity field only. The numerical scheme is based on the following discretization:

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k}}{\Delta t} = -(\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^{n+1} - \nabla p + \nu \nabla^2 \mathbf{u}^{n+1}. \tag{2.35a}$$

The main difficulty in implicit solution is the treatment of non-linear term, to this end the following numerical scheme is implemented.

Stage 1:

$$\mathbf{u}^{n+1} \approx \tilde{\mathbf{u}}^{m=0} = \breve{\mathbf{u}}^{m=0} = \mathbf{u}^n, \tag{2.35b}$$

here index m denotes the $m-th$ subiteration.

Stage 2:

$$\mathbf{u}^* = \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k} - \Delta t (\tilde{\mathbf{u}}^m \cdot \nabla)\tilde{\mathbf{u}}^m. \tag{2.35c}$$

Stage 3:

$$\nabla^2 p = (\Delta t)^{-1} \nabla \cdot \mathbf{u}^* \tag{2.35d}$$

supplemented with Dirichlet or Neumann boundary conditions, The Neumann boundary conditions are computed from

$$\frac{\partial p}{\partial n} = -\left(\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{k=0}^{Je-1} \beta_k \mathbf{u}^{n-k}}{\Delta t} + [(\tilde{\mathbf{u}} \cdot \nabla)\tilde{\mathbf{u}}]^m + \nu \nabla \times \nabla \times \tilde{\mathbf{u}}^m \right) \cdot \hat{\mathbf{n}}. \tag{2.35e}$$

The $\frac{\partial p}{\partial n}$ is computed on Ω_δ, where the exact value of $\mathbf{u}^{n+1}$ term is given.
Stage 4: implicit solution of the Helmholtz problem

$$\frac{\gamma_0 \breve{\mathbf{u}}^{m+1} - \mathbf{u}^*}{\Delta t} = -\nabla p + \nu \nabla^2 \breve{\mathbf{u}}^{m+1}. \tag{2.35f}$$

Stage 5:

$$\tilde{\mathbf{u}}^{m+1} = \omega \breve{\mathbf{u}}^{m+1} + (1-\omega)\breve{\mathbf{u}}^m, \tag{2.35g}$$

where $0 < \omega < 1$ is a relaxation parameter. For $\omega = 0$ the semi-explicit first order scheme is obtained. An alternative formulation to (2.35g) is

$$\tilde{\mathbf{u}}^{m+1} = \omega \breve{\mathbf{u}}^{m+1} + (1-\omega)\tilde{\mathbf{u}}^m. \tag{2.35h}$$

It was observed that (2.35h) leads to convergence rate two to three times slower than (2.35g). At the first subiteration the two formulations are equivalent. According to formula (2.35g), for $m > 1$ the value of $\tilde{\mathbf{u}}^{m+1}$ is calculated from the two successive solutions of (2.35f). However, using (2.35h) the value of $\tilde{\mathbf{u}}^{m+1}$ includes the solution $\mathbf{u}^n$ premultiplied by weight $(1-\omega)^m$, which slows the convergence.
The Stages 2 to 5 are repeated till convergence in the field $\breve{\mathbf{u}}^m$ is achieved. To terminate the subiterations we employed the following stopping criteria $|\breve{\mathbf{u}}^{m+1} - \breve{\mathbf{u}}^m|_{L_\infty} < TOL$, where the values of modal coefficients (2.7) of $\tilde{\mathbf{u}}^m$ are compared. Upon convergence the solution at t^{n+1} is $\mathbf{u}^{n+1} \equiv \tilde{\mathbf{u}}^{m+1}$. Several strategies has been suggested to accelerate the convergence of iterative solution. For example, Aitkens acceleration technique with adaptive relaxation parameter was used in [25]. For relatively simple flows such as laminar flow around backward facing step, as considered in this study, the convergence is typically obtained in 2-4 subiterations when formula 2.35g is employed with relaxation parameter $\omega = 0.5$ and $TOL = 1E-7$.

To write (2.33a,2.33b) in the variational formulation, the appropriate spaces for the

velocity and pressure must be defined. The appropriate functional spaces are defined by the highest (spatial) derivatives of $\mathbf{u}$ and p :

$$\mathcal{U} \equiv H^1(\Omega)^d, \quad \mathcal{M} \equiv L^2(\Omega)$$

where $\mathcal{U}$ is the appropriate space for the velocity and $\mathcal{M}$ for the pressure, d is the spatial dimension.

The discrete spaces for velocity and pressure $(\mathcal{U}^\delta,\ \mathcal{M}^\delta)$ are formed as a sub-spaces of $(\mathcal{U},\ \mathcal{M})$ using polynomial bases functions. The velocity and pressure terms are coupled and hence can not be approximated independently. To satisfy a *compatibility* conditions also known as *inf-sup* conditions the pressure space has to be restricted to exclude spurious modes p^*, defined by $(\mathbf{w}, \nabla p^*) = 0$, where $\mathbf{w} \in \mathcal{U}$. Bernardi&Maday [20] suggested the so called $P/P-2$ approach where the velocity field is approximated by polynomial functions of degree $\leq P$, while the pressure term is approximated with polynomial functions of degree $\leq P-2$. The filtering of the spurious pressure modes using the $P/P-2$ approach is required in solution of *coupled* Stockes system. An alternative avenue to solve the Navier-Stokes equations (2.33a,2.33b) is to decouple the pressure and the velocity fields. The decoupling (splitting) leads to solution of the Poisson equation for the pressure, which along with the correct pressure boundary conditions leads to unique solution, that is solution without the spurious modes.

The objective here is to apply the $P/P-k,\ k=0,1,2$ approach for solution of NS equations with time-splitting scheme in order to evaluate numerically the robustness of the method. The following simulations have been performed. First, we apply the $P/P-k,\ k=0,1,2$ approach for solution of Navier-Stokes equations with semi-implicit time-splitting scheme. The test problem we solve numerically is the Kovaszany flow, for which analytical solution is known. The goal here is to compare the accuracy of the solutions obtained with $k=0,1,2$. Second, we consider the so called "step flow" problem, where discontinuity in the gradients of the velocity and pressure field appears. The semi-implicit scheme with the $P/P-k, k=0,1,2$ approach and the fully-implicit scheme with the P/P approach are applied.

$P/P-k$ approach for solution of 2D Navier-Stokes problem with operator splitting scheme: resolution study

To evaluate the accuracy of numerical solution obtained with the reduced space discretization for the pressure, we solve the Kovasznay flow problem. The problem is defined on a rectangular domain $\Omega = [-0.5\ 3] \times [-0.5\ 1.5]$ which is subdivided into $Nel = 24$ triangular elements. The computational domain is presented in figure 2.22. On each element solu-

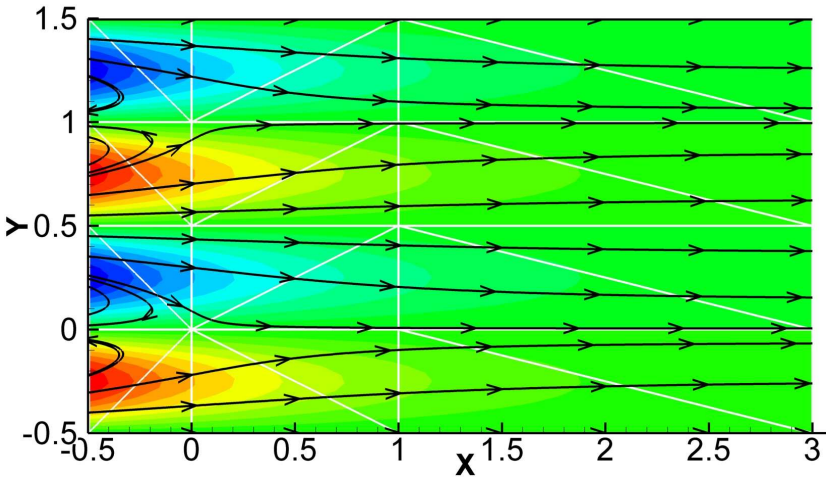


Figure 2.22: (in color) Kovasznay problem: computational domain and solution. Colors represent the vorticity, arrows represent the flow direction. Wide white lines depict the edges of spectral elements. Computational domain $\Omega = [-0.5\ 3] \times [-0.5\ 1.5]$, $Nel = 24$; $Re = 40$.

tion is approximated using Cartesian tensor product bases. At the domain boundaries Ω_δ Dirichlet boundary condition is applied for the velocity variables. For the pressure, Dirichlet boundary condition is applied at the boundary $x = 3$, and Neumann boundary conditions are set at other boundaries. The numerical solution is compared to the analytical one, given by

$$u = 1 - e^{\mu x}\cos(2\pi y), \quad v = \frac{\mu}{2\pi}\sin(2\pi y), \quad p = p_0 - \frac{1}{2}e^{2\mu x}, \tag{2.36}$$

where $\mu = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}$, and $Re = 1/\nu$. Note that $\mu < 0$, and for lower Re numbers the gradient $\partial u/\partial x$ is increasing in its magnitude. In figure 2.23 we compare the accuracy of numerical solution obtained with $P = 5, 7, 9, 11$ and $Re = 40$. It is noticeable that the accuracy of solutions obtained with the P/P and $P/P-1$ approaches is almost the same, while about an order of accuracy is lost when $P/P-2$ approach is implemented. In figure 2.24 we plot the convergence of numerical solution corresponding to $Re = 10$. Note that

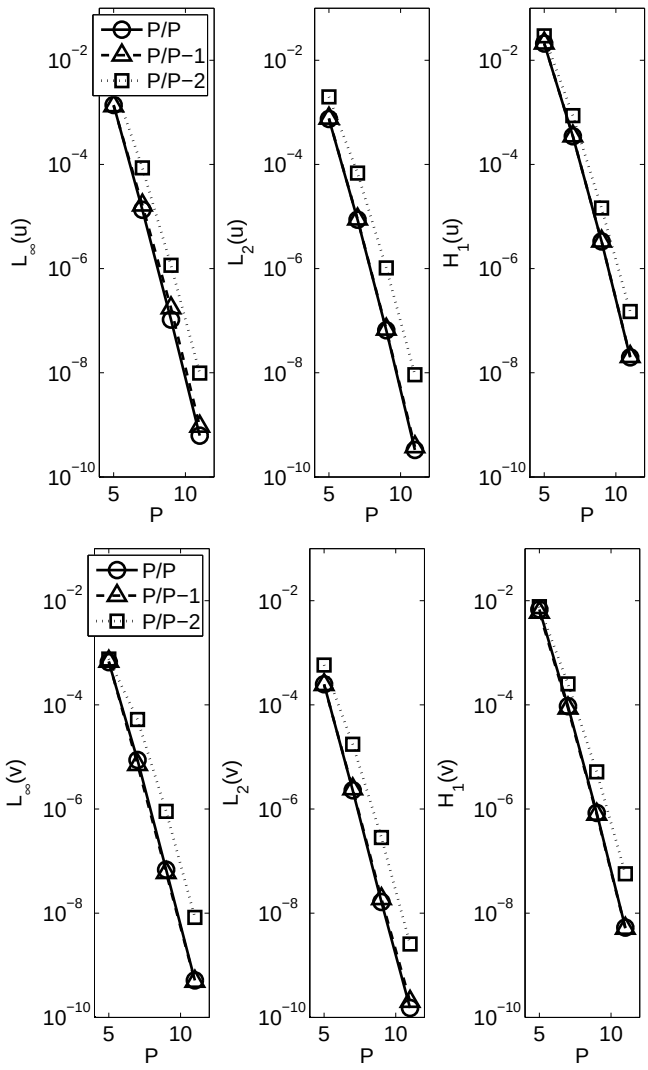


Figure 2.23: Solution of Kovasznay problem: convergence of velocity. Computational domain $\Omega = [-0.5\ 3] \times [-0.5\ 1.5]$, $Nel = 24$; $Re = 40$.

the Kovasznay problem with lower Re number requires higher spatial resolution than for high Re. In figure 2.25 the convergence of the pressure is plotted. Spectral convergence is observed for the three choices P/P, $P/P-1$ and $P/P-2$, however, superior convergence is obtained for the P/P discretization.

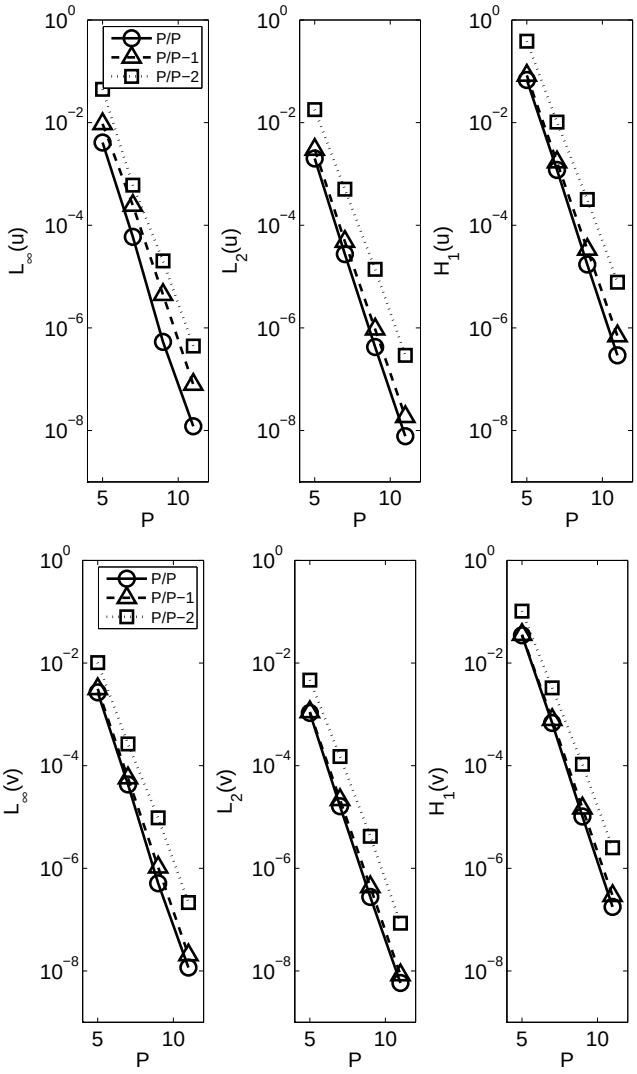


Figure 2.24: Solution of Kovasznay problem: convergence of velocity. Computational domain $\Omega = [-0.5\ 3] \times [-0.5\ 1.5]$, $Nel = 24$; $Re = 10$.

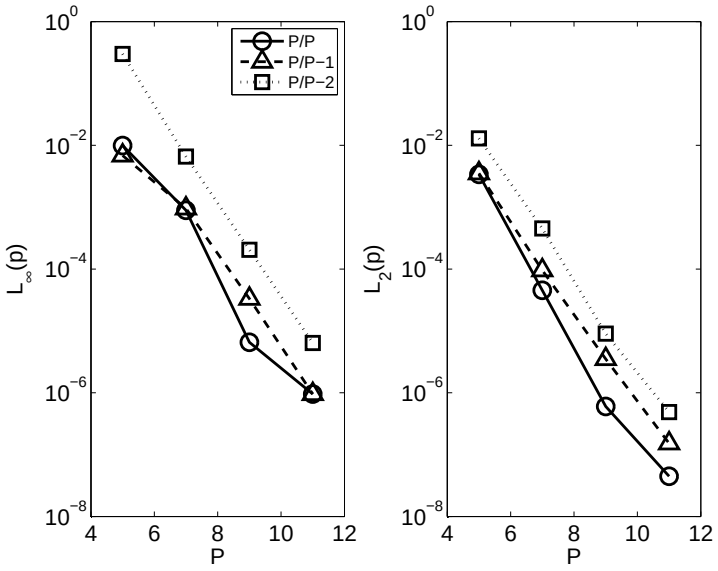


Figure 2.25: Solution of Kovasznay problem: convergence of pressure. Computational domain $\Omega = [-0.5\ 3] \times [-0.5\ 1.5]$, $Nel = 24$; $Re = 10$.

Next, we focus on solution of "step-flow" problem, the computational domain and finite element discretization are illustrated in figure 2.26. The following boundary conditions for

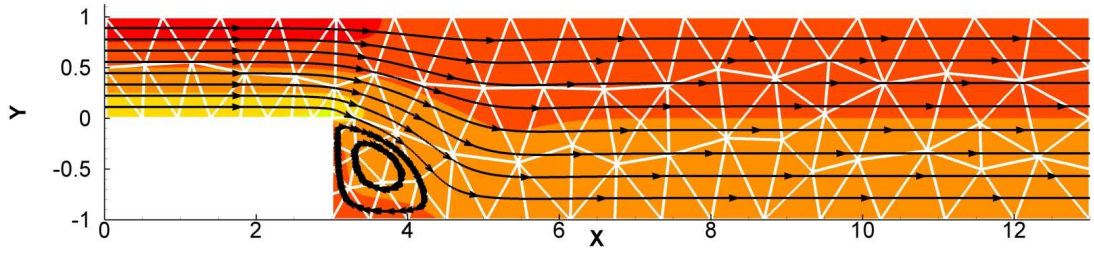


Figure 2.26: (in color) Step flow problem: computational domain and solution. Colors represent the vorticity, arrows represent the flow direction. Wide white lines depict the edges of spectral elements. Computational domain $\Omega = [0\ 13] \times [-1\ 1]$, $Nel = 124$; $Re = 25$.

the velocity variables are applied: at the inlet ($x = 0$) $u = y(1 - y)$, $v = 0$, at the outlet ($x = 13$) fully developed flow is assumed, and at other boundaries $u = v = 0$. Dirichlet boundary condition for the pressure ($p = 0$) is applied at the outlet and Neumann boundary condition at other boundaries. The simulations have been performed with $Re = H\bar{u}/\mu = 25$, where, $H = 1$ is the hight of the domain and $\bar{u} = 1/6$ is the mean velocity at the inlet. The initial solution was obtained from the solution of a Poisson problem for the velocity $\nabla^2\mathbf{u} = 0$, supplemented with the exact boundary conditions at all boundaries except $x = 13$, where fully developed flow was assumed. The solution was integrated till $T = 15$. To eliminate aliasing error the 3/2 (over-integration) rule was applied [51] in evaluation of the non-linear term.

It is well known that the flow around backward-facing step exhibits a singularity in the pressure and the velocity derivatives at the step corner, which may trigger erroneous oscillations. The objective here is to examine numerically the filtering of the high frequency components in the pressure by applying the P/P, $P/P - 1$ and $P/P - 2$ approaches in conjunction with the semi-implicit solver, and by solving the problem implicitly.

In figures 2.27 and 2.28 we present the pressure field in the vicinity of the step, erroneous pressure oscillations are observed, particularly in solution with lower-order polynomial approximation ($P = 5$) and P/P approach. Note that the oscillations appear not only in the elements adjacent to the corner but also propagate to the surrounding elements, mostly upstream the flow. Although, higher-order spatial resolution helps to suppress the oscillations (compare results obtained with $P = 5$ and $P = 9$) the reduced-space pressure approximation seems to be more efficient.

The simulations of step-flow have been performed with fully implicit scheme (2.35a). The main difference between the implicit and semi-implicit scheme (beside the superior stability) is in the solution of Poisson equation for the pressure, specifically, in calculating the Neumann pressure boundary conditions: the implicit scheme removes the splitting error. In figure 2.29 we compare the pressure fields computed with semi-implicit and fully implicit schemes and P/P approach with $P = 5, 9$. Clearly the implicit formulation is more robust than the semi-explicit one. The absence of the pressure oscillations in solution obtained with the implicit solver confirms that the origin of the oscillations is in the time-discretization (splitting) error, specifically in the explicit treatment of the Neumann pressure boundary conditions.

The implicit solution of the 2D step-flow problem lead to significant computational savings. The relaxation parameter implemented in this study was $\omega = 0.5$ and 2-4 subiterations at each time steps were required for solution with $P = 5$ and $4 - 6$ subiterations with $P = 9$. The first subiteration typically required about 77 conjugate gradient iterations for the pressure and about 25 for each velocity component (the diagonal preconditioner was employed). At the second subiteration 70 (20) conjugate gradient iterations were required for the pressure (velocity) solves, while, starting from the third subiterations, less than 50 iterations were required for the pressure solver and 1 to 8 iterations for the velocity solver. For the first subiteration the solution from the previous time step was used as initial condition for the conjugate gradient solvers, and subsequently solution from the previous subiteration was used as an initial condition. The stopping criteria for the conjugate iterations in solution of linear system $Ax = b$ was $r < 1.0e - 12$, where $r = ||Ax - b||$ is the residual. It is important to note that the overhead in the implicit solution of the NS equation is not only repeated solution of linear systems, but also calculation of the non-linear term. The calculation of the non-linear term is performed in physical space, i.e., transformation from the modal to physical space is necessary; once the non-linear term is computed the inner product of the non-linear term and the test function has to be taken, which is also relatively expensive operation. Computation of the Neumann pressure boundary conditions is also CPU-intensive due to calculation of $\nabla \times \nabla \times \mathbf{u}$. However, in the framework of parallel computing the aforementioned operations are embarasingly parallel. In the solution of the step-flow problem, considered here, the overall performance of the implicit solver was better than the semi-implicit one, in terms of both - the CPU-time and accuracy.

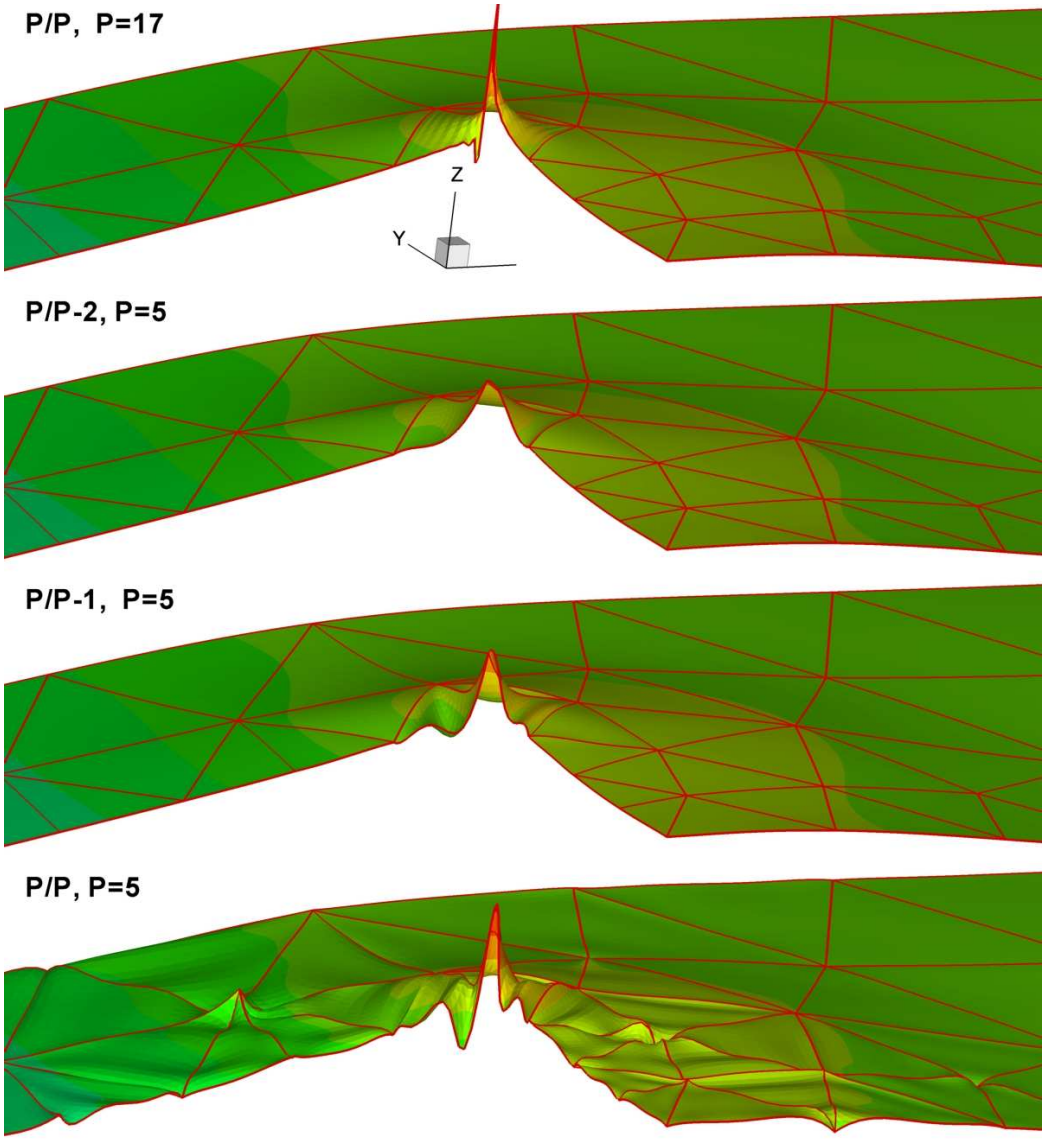


Figure 2.27: (in color) Step flow: Solution computed with different spatial resolutions: $P = 5$ and $P = 17$ (top plot), and different pressure discretization methods: P/P, $P/P-1$ and $P/P-2$. Top plot . z-axis - pressure (also displayed in color). Red lines depict the edges of spectral elements.

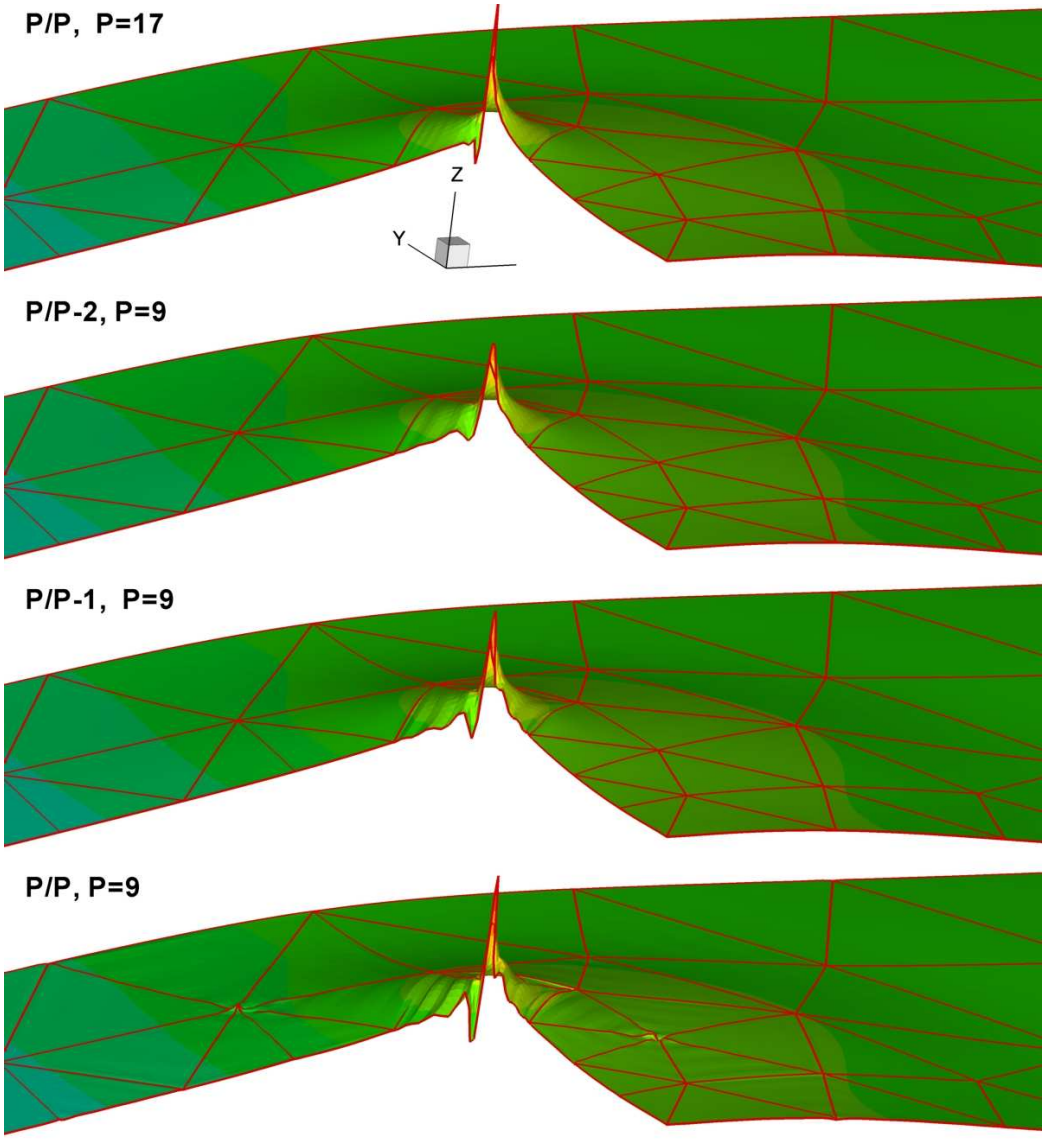


Figure 2.28: (in color) Step flow: Solution computed with different spatial resolutions: $P = 9$ and $P = 17$ (top plot), and different discretization approaches: P/P, $P/P - 1$ and $P/P - 2$. z-axis - pressure (also displayed in color). Red lines depict the edges of spectral elements. $Re = 25$, $Nel = 124$, $\Delta t = 0.001$, $Je = 2$.

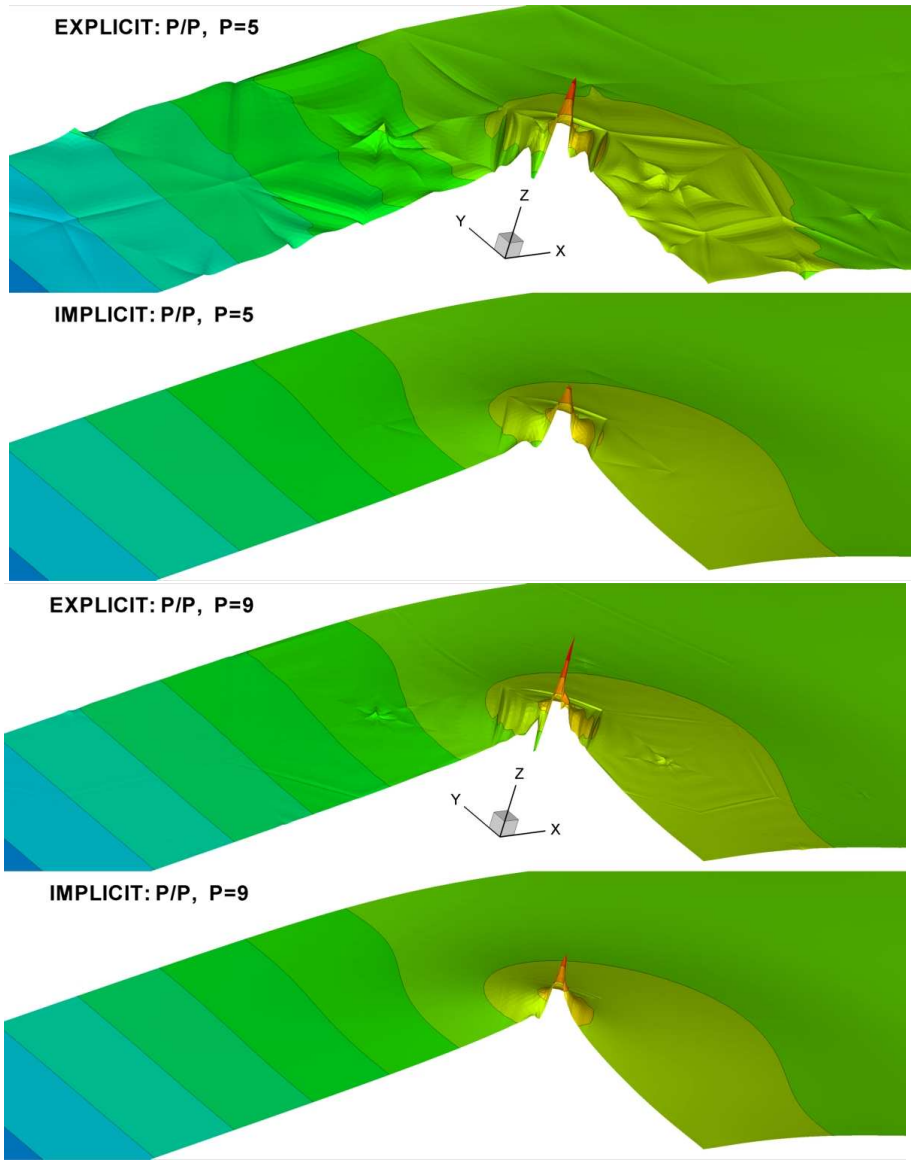


Figure 2.29: (in color) Step flow: Solution computed using explicit (2.34a) and fully implicit scheme (2.35a) with spatial resolutions: $P = 5$ and $P = 9$, and P/P approach. z-axis - pressure (also displayed in color). $Re = 25$, $Nel = 124$, $\Delta t = 0.001$, in semi-implicit formulation and $\Delta t = 0.01$ in implicit, $Je = 2$.

2.8 Hierarchical spectral basis and Galerkin formulation using barycentric and collapsed Cartesian quadrature grids in triangular elements

Here we focuses on *modal* spectral expansions and the Galerkin formulation for the divergence-free Navier-Stokes equations [78]. The weak formulation of the NS equations suggest that the solution is obtained in the modal space; however, the most efficient treatment of the non-linear term is to compute it in the physical space and then take its inner product with the test functions. Calculations being done in physical space require defining appropriate quadrature. In the past, the discrete system has been formulated by employing Gauss type quadratures on a cartesian grid that is mapped to a standard square domain. Here we attempt, to reformulate the discrete system based on barycentric quadrature grids formed by the nodal sets of points employed in the aforementioned nodal spectral expansions. Thus, the set of nodes which are used for interpolation for spectral collocation methods in the triangle will be used here for numerical integration and differentiation. We are particularly interested in examining the *quadrature crimes* [88] that may arise due to the nonlinear (advection) terms in the Navier-Stokes equations. By *quadrature crimes* we mean violations associated with quadrature rules for consistent numerical integration and differentiation of a polynomial function.

In the following, we first define the various quadrature grids and subsequently we analyze the three basic operations, namely integration, differentiation and projection. We then perform a comparison of accuracy and efficiency for the three different approaches for the Kovasznay flow and analyze the numerical stability of the corresponding discrete system.

2.8.1 Barycentric grids on triangle

We have introduced the cartesian and barycentric grids in section 2.3, and in this section we extend the discussion on the barycentric grid. Here we denote the collapsed-cartesian grid introduced in section 2.3 by CCG. In the present study we consider two types of barycentric grids. The first was developed by Blyth & Pozrikidis [23], and we will refer to this grid as a barycentric grid of a type A (BGA); a two-dimensional grid is generated from a one-dimensional *master grid*. Here we employ the Gauss grid in a non-tensorial coordinate system, whereas in [23] the gauss-Lobatto grid was used. The total number

of nodal points over a triangular element is $N = m(m+1)/2$, where m is the number of grid points in one dimension. The highest order of polynomials, P, that can be exactly integrated and differentiated is limited by $P = m - 1$. The main advantage of the BGA is that we obtain a *symmetric* distribution of quadrature points with respect to the vertices of a triangular element, also it is easy to compute the coordinates of the quadrature points and the associated weights for numerical integration.

The second type of barycentric grid was developed by Taylor *et al* [91] and we will refer to this grid as a barycentric grid of a type B (BGB). The number of quadrature points, N, is defined by the order of cardinal functions - polynomials of order P - as $N = (P+1)(P+2)/2$. The distribution of quadrature points over a triangular elements and their associated weights are optimized in order to obtain exact integration of polynomials of order higher than P. The distribution of quadrature points is not always symmetrical; however, even in a case of asymmetrical distribution, there is no clustering of quadrature points as in the case of the collapsed-cartesian grid. All quadrature points computed with the cardinal function of degree up to 14 are located inside the triangle and have positive weights. Integration on the BGB grid of polynomials up to order $P = 25$ is exact. We emphasize that the BGA grid was constructed for interpolation purposes, while the BGB grid was optimized for integration. In figure 2.30 we show typical distributions of cartesian and barycentric grid points over an equilateral triangle.

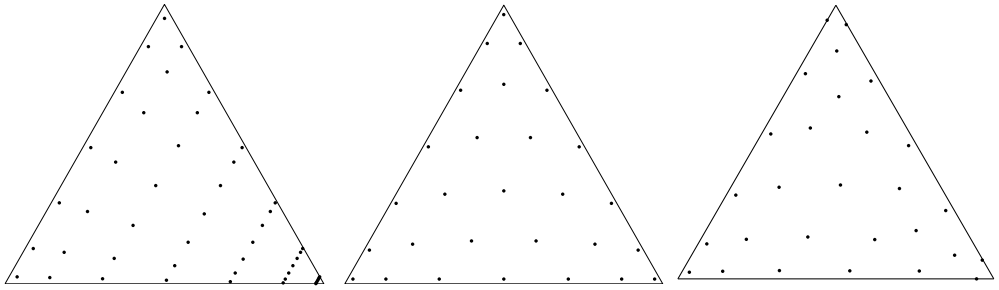


Figure 2.30: Quadrature grids: Left - CCG; Middle - BGA; Right - BGB. The total number of quadrature points, N, is defined as: $N = (P+1)^2$ for CCG, $N = (P+1)(P+2)/2$ for BGA and BGB; P = 6.

Another set of symmetrical nodes, optimized for integration on a triangular element was suggested by Wandzura & Xiao [99]. The highest order of polynomial function that can be exactly integrated using this set is limited by $P = 30$. It was argued in [91] that integration on BGB is generally less expensive than integration using the Wandzura & Xiao grid. The

Vandermonde matrix constructed from modal bases defined on a triangular element and Wandzura & Xiao points is not a square matrix, which provides additional complexity in numerical differentiation. In the current work we have not studied the Wandzura & Xiao points.

The integration on cartesian and barycentric grids was discussed in section 2.4.3. The suggested procedure to compute the weights for numerical integration on the barycentric grid was based on construction of the Vandermonde matrix. However, this method is appropriate for the BGA grid and not for the BGB. For BGB the procedure of computing the location of quadrature points and weights is more complicated; a detailed explanation can be found in [91].

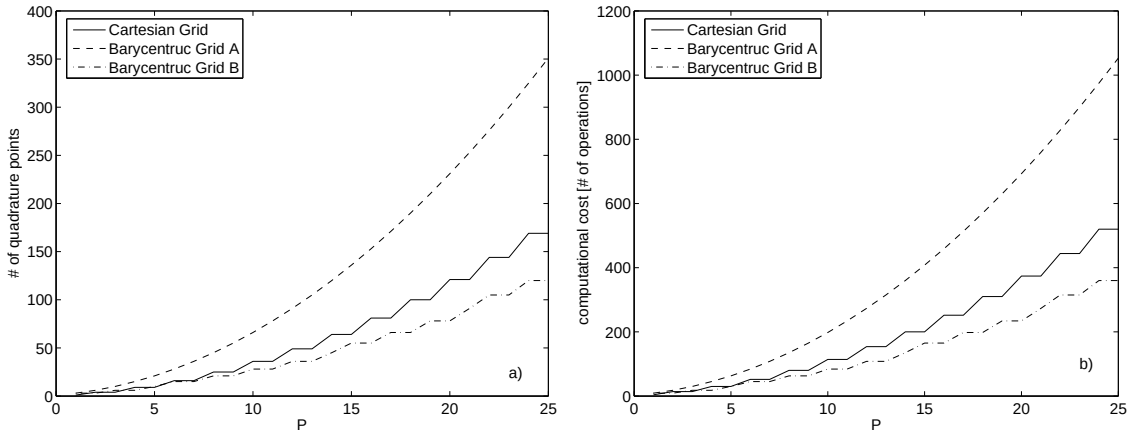


Figure 2.31: Left: Number of quadrature points for exact numerical integration of a polynomial of order P. Right: Corresponding computational cost.

The computational cost of the exact integration of polynomial of degree P on BGA is $(P+1)(P+2)$ summations and multiplications while the cost of function evaluations scales as $(P+1)(P+2)/2$. On the other hand, for the exactly same computational cost we can integrate polynomials of degree higher than P using BGB. We summarize this section by comparing the number of quadrature points and the number of numerical operations for integration of polynomial of degree P. In figures 2.31(a,b) we plot the results for the three types of quadrature. We assume that the computational cost for function evaluation at each quadrature point is one unit. Clearly, the quadrature grid BGB is more efficient than BGA.

The *projection* operator from *modal to physical space* is defined by

$$u(\xi_{1_i}, \xi_{2_j}) = \sum_{k=1}^{K} \hat{u}_k \Lambda_k(\xi_{1_i}, \xi_{2_j}),$$

here Λ represents an arbitrary spectral basis, however in this study we employ the cartesian bases $\mathbf{\Phi}$ only.

Thus, the computational cost of computing $u(\xi_{1_i}, \xi_{2_j})$ at a single quadrature point does not depend on the type of grid, but the overall cost depends on the number of quadrature points. This implies that using the barycentric grid with $N = (P+1)(P+2)/2$ quadrature points is almost half the cost of using CCG with $N = (P+1)^2$ quadrature points.

The projection from *physical to modal space* is performed in three steps. First, we compute the mass matrix $\mathbf{M}$; second, we integrate the function with respect to the test functions; third, we solve the algebraic system $\mathbf{M}\hat{\mathbf{u}} = \mathbf{f}$, where the unknowns $\hat{\mathbf{u}}$ are amplitudes of the expansion basis. The efficiency of the first and the second steps depends on the effectiveness of the numerical integration. However, we note that CCG is based on a tensor-product construction. For tensor-product bases we can employ the *sum-factorization* to effectively reduce the computational cost. On barycentric grids the sum-factorization is not possible [79]. We also note that the second step is repeated $(P+1)(P+2)/2$ times equal to the number of modes in a polynomial expansion. In figure 2.31b we demonstrate that the choice of BGB leads to the most efficient integration and thus the most efficient projection.

2.8.2 Application to a Navier-Stokes solver

This section consists of three parts. First, we overview the model problem and numerical scheme, specifically we focus on different formulations of the non-linear term. In the second part we analyze the numerical efficiency of a Navier-Stokes solver where different quadrature grids are employed. In the third part we analyze the effect of quadrature grids on the temporal stability of the Navier-Stokes solver.

The aforementioned three types of quadrature grids were employed in all operations in a 2D spectral/*hp*-element code for incompressible Navier-Stokes equations. As a prototype problem we consider the Kovasznay flow, already presented in section 2.7.

The numerical solution is obtained by solving the incompressible NS equations in terms

of the primitive variables (2.33a). In this study the semi-implicit NS solver (2.34a) is employed. The computational domain was decomposed into 24 triangular elements, as shown in figure 2.22. The Dirichlet boundary conditions are applied for solution of Helmholtz solver for the velocity field. For the pressure the Dirichlet boundary condition $p = 0$ is applied at $x = 3$ and Neumann boundary condition on other boundaries. The initial velocity field is obtained from the solution of $\nabla^2 \mathbf{u} = 0$ with exact boundary conditions. The size of the time step was fixed to $\Delta t = 1.0E - 4$. To estimate the numerical efficiency, we monitored the CPU-time required to obtain a solution at time $T = 6$ and the $L_2 - error$ of a solution at $T = 1, 2, 3, 4, 5, 6$ (steady state is achieved after $T = 4$.) Also, $P = 6$ and $P = 8$ were used for spectral polynomial order.

In (2.34a) the convective formulation of the nonlinear term is presented, however, the non-linear term $\mathbf{NL} = [NLx\ NLy]$ can be formulated in conservative, rotational, skew-symmetric and convective forms. We performed numerical experiments using all formulations. In this study we concentrate primarily on the conservative (2.37) and convective (2.38) forms.

$$NL_x = \frac{\partial (u)^2}{\partial x} + \frac{\partial uv}{\partial y}, \qquad NL_y = \frac{\partial uv}{\partial x} + \frac{\partial (v)^2}{\partial y}. \tag{2.37}$$

$$NL_x = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}, \qquad NL_y = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}. \tag{2.38}$$

The weak form of linear terms requires integration of polynomials from $\mathcal{V}^{2P}$, while for the nonlinear terms the integrated polynomials are in $\mathcal{V}^{3P-1}$. The calculation of $\mathbf{NL}$ involves several steps: On the first step, values of $\mathbf{u}$ are computed on N quadrature points by projection from modal to physical space. On the second step, the derivatives operator is applied to obtain $NL_x,\ NL_y$. Finally, NL_x and NL_y are integrated with respect to the test functions. We note that values of $NL_x,\ NL_y$ in physical space are also used for the evaluation of the right-hand side for the pressure equation, (2.34c). The divergence of the provisional field $\mathbf{u}^\star$ is, first, computed by applying the derivative operator on $\mathbf{u}^\star$ and, second, by integration of $\nabla \cdot \mathbf{u}^\star$ with respect to the test functions.

A *consistent* implementation of the weak form of the nonlinear terms implies that the number of quadrature points should satisfy the rules for *exact* numerical integration and *exact* differentiation of polynomials from $\mathcal{V}^{3P-1}$ and $\mathcal{V}^{2P}$, respectively. Our numerical experiments were divided into three categories:

(a) Number of quadrature points is sufficient for exact integration of polynomials from $\mathcal{V}^{2P}$.

(b) Number of quadrature points is sufficient for exact integration of polynomials from $\mathcal{V}^{3P-1}$.

(c) Number of quadrature points is sufficient for exact integration of polynomials from $\mathcal{V}^{3P-1}$ and exact differentiation of polynomials from $\mathcal{V}^{2P}$.

We will refer to the method in case (a) as inconsistent numerical integration; in case (b) as consistent numerical integration; and case (c) as fully consistent numerical integration and differentiation.

2.8.3 Numerical efficiency

Let P_I denote the highest order of polynomial function that can be exactly integrated on a given set of quadrature points and P_d the highest order of polynomial function that can be exactly differentiated. We also denote by N the total number of quadrature points. For each type of grid the number of quadrature points is chosen according to the following principles.

Collapsed-cartesian grid (CCG): The number of quadrature points is set according to rules for one-dimensional integration. Using Q Gauss quadrature points per direction we can integrate exactly polynomials of order $P \leq 2Q - 1 = P_I$ and differentiate exactly polynomials of order $P \leq Q - 1 = P_d$. The total number of grid points on a triangular element is $N = Q^2$.

Barycentric grid of type A (BGA): The highest order of polynomial that can be exactly integrated and differentiated is defined by the number of grid points of the one-dimensional master grid from which the two-dimensional one is constructed. In order to have exact numerical differentiation and integration we need $m = P_d + 1 = P_I + 1$ grid points in the one-dimensional grid. Thus, the total number of grid points in the triangular region is defined by $N = (P_d + 1)(P_d + 2)/2$.

Barycentric grid of type B (BGB): The number of quadrature points for consistent differentiation is defined similarly as for BGA, i.e., $N = (P_d + 1)(P_d + 2)/2$. However, the highest order of polynomial that can be exactly integrated using the same number of grid points is greater than P_d. Appropriate values of N for exact integration of polynomial function in $\mathcal{V}^{P_I}$ are provided in Table 7.1 of [91].

We recall that BGB was optimized for numerical integration, while BGA was not; moreover some of the weights, computed on BGA, for integration of polynomials with $P > 6$ are

negative.

In handling the nonlinear terms it is convenient to perform the basic numerical operations on the same grid, that is we use the same points for integration and differentiation. In nonlinear problems we have that $P_I > P_d$, which implies that it is not always possible to have N grid points and satisfy *exactly* the rules for accurate integration and differentiation without paying an extra cost by performing over-integration or differentiation (in a super-collocation fashion); see [52]. Also, in the discretization of the nonlinear terms with $\mathbf{u}^\delta \in \mathcal{V}^P$ we may specify the number of quadrature points for each type of grid in order to have (a) exact integration of the weak linear advection operator; (b) exact integration of the weak nonlinear advection operator; and (c) exact integration and differentiation of the weak nonlinear advection operator.

In Table 2.1 we summarize the values of N, P_I and P_d given the above considerations for the nonlinear problem with $\mathbf{u}^\delta \in \mathcal{V}^6$. The number of quadrature points on each grid was chosen such that we have: (a) inconsistent numerical integration, (b) consistent numerical integration, and (c) consistent numerical integration and differentiation. In our problem, the order of polynomial function that should be integrated is higher than the order for differentiation, thus consistent integration on BGA leads to consistent differentiation as well. For this reason the second and the third rows that correspond to BGA in Table 2.1 (also in Table 2.2) are the same. We note that for this particular problem the highest order of a polynomial function to be integrated is 17 and the highest order of polynomial function to be differentiated is 12, since we use the conservative formulation for the nonlinear terms.

	Collapsed-cartesian			Barycentric grid A			Barycentric grid B		
	N	P_I	P_d	N	P_I	P_d	N	P_I	P_d
(a)	49	13	6	91	12	12	36	13	7
(b)	81	17	8	171	17	17	66	18	10
(c)	169	25	12	171	17	17	120	21	12

Table 2.1: Number of quadrature points for (a) inconsistent numerical integration, (b) consistent numerical integration, and (c) consistent numerical integration and differentiation of a weak nonlinear operator; $\mathbf{u}^\delta \in \mathcal{V}^6$.

The Kovasznay flow problem was solved on the three different grids, with the number of grid points specified according to Table 2.1. In figure 2.32 we present the L_2-error as a function of time. The left plot illustrates the behavior of the error of the solution when

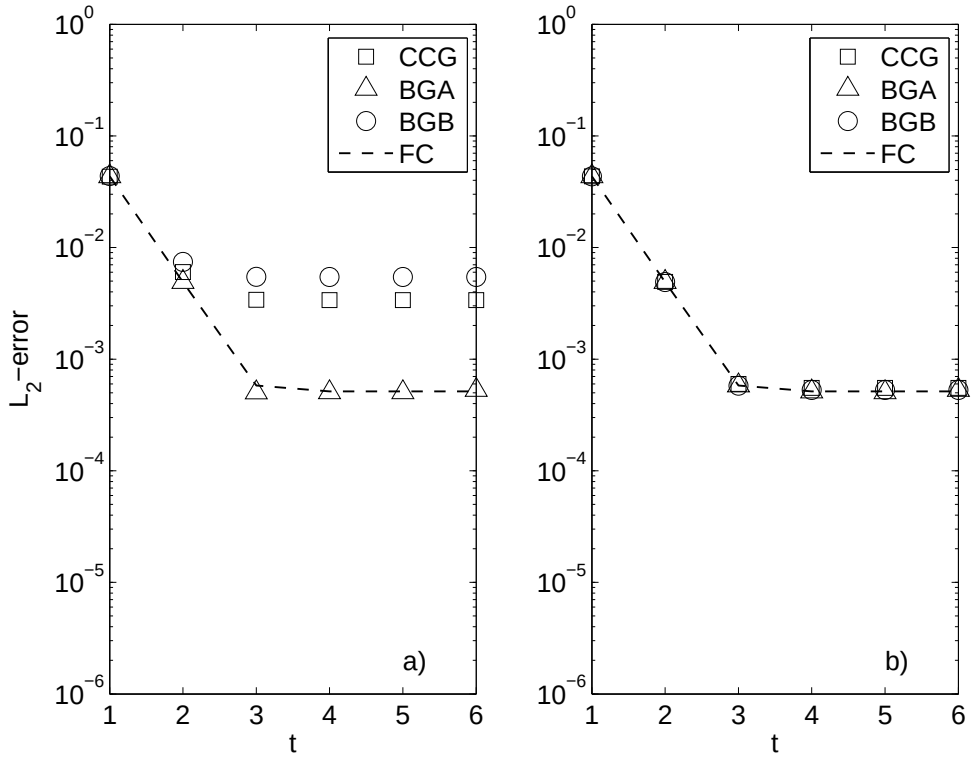


Figure 2.32: L_2 -error versus time for the collapsed-cartesian grid (CCG), barycentric grid of type A (BGA) and B (BGB) with (a) inconsistent numerical integration, and (b) exact numerical integration. The dash line (FC) depicts the fully consistent case; $\mathbf{u}^\delta \in \mathcal{V}^6$; conservative formulation of the nonlinaer term.

the number of quadrature points is sufficient to integrate a weak linear advection operator only. We note that for the collapsed-cartesian grid as well as for barycentric grid of type B this number of quadrature points is insufficient for exact numerical differentiation. On the other hand, for BGA the numerical differentiation is exact. The dash line depicts the fully-consistent case, i.e., when we have consistency in both integration and differentiation. The right plot shows the L_2-error for tests where the number of quadrature points is sufficient for consistent numerical integration. We observe that in terms of accuracy the choice of barycentric grid of type A is advantageous for the case of inconsistent numerical integration. Another observation is that insufficient resolution for numerical differentiation alone does not affect the overall accuracy, as shown in the right plot.

We performed the same test with rotational and skew-symmetric forms of the nonlinear term. The results were very similar to the results obtained with conservative formulation of the **NL**. The drop of accuracy was observed when *both* numerical integration and differentiation suffered from under-resolution.

A different situation was observed when the convective form of **NL** was used; using the convective form always leads to exact numerical differentiation. In figure 2.33 we present the numerical error of the solution obtained using the convective form and inconsistent numerical integration. We see no drop of accuracy due to numerical under-integration.

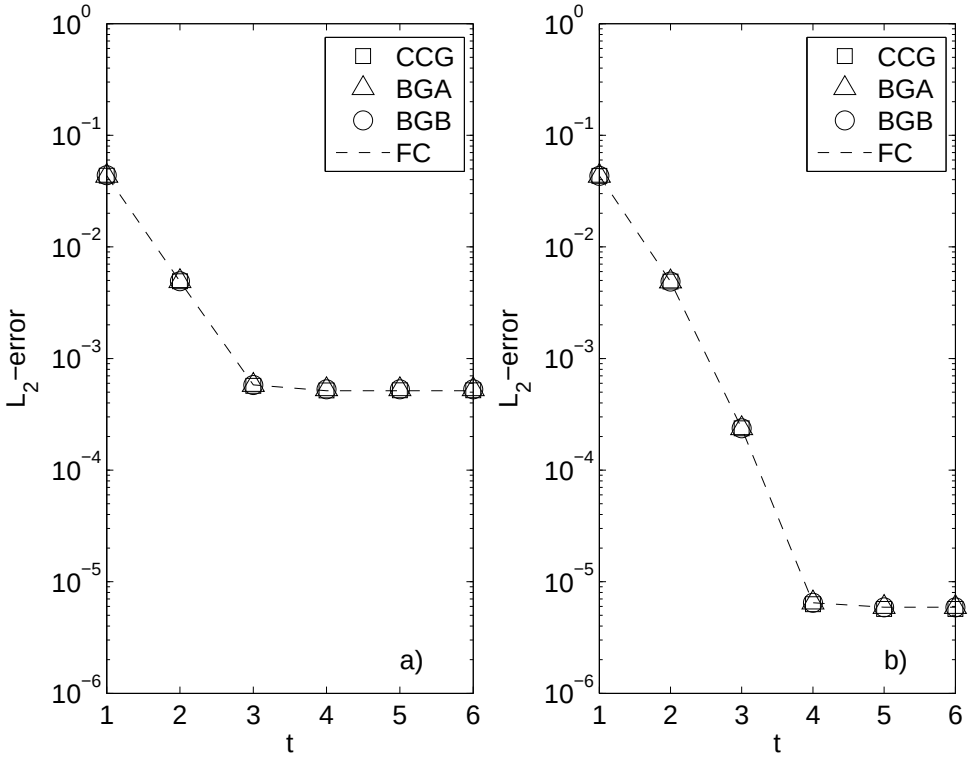


Figure 2.33: L_2 -error versus time for the collapsed-cartesian grid (CCG), barycentric grid of type A (BGA) and B (BGB) with inconsistent numerical integration and convective formulation of the nonlinear term; (a) $\mathbf{v}_P \in \mathcal{V}^6$; (b) $\mathbf{u}^\delta \in \mathcal{V}^8$ The dash line (FC) depicts the fully consistent case.

	Collapsed Cartesian			Barycentric grid A			Barycentric grid B		
	N	P_I	P_d	N	P_I	P_d	N	P_I	P_d
(a)	81	17	8	153	16	16	55	16	9
(b)	144	23	11	300	23	23	105	23	13
(c)	289	33	16	300	23	23	120	120	14

Table 2.2: Number of quadrature points for (a) inconsistent numerical integration, (b) consistent numerical integration, and (c) consistent numerical integration and differentiation of a weak nonlinear operator; $\mathbf{u}^\delta \in \mathcal{V}^8$.

The Kovasznay flow problem was also solved with higher accuracy, corresponding to $\mathbf{u}^\delta \in \mathcal{V}^8$. In Table 2.2 we provide a summary of number of quadrature points and P_I, P_d for each simulation. We note that in case of $\mathbf{u}^\delta \in \mathcal{V}^8$ the highest order of polynomial function to

be integrated is 23 while the highest order of polynomial function to be differentiated is 16. For BGB the maximum number of quadrature points is 120 which corresponds to cardinal function of a degree 14; for this reason the $max(P_d) = 14$. The results of convergence of the error in the L_2 norm are presented in figure 2.34; we observe similar behavior as in the case with $\mathbf{u}^\delta \in \mathcal{V}^6$.

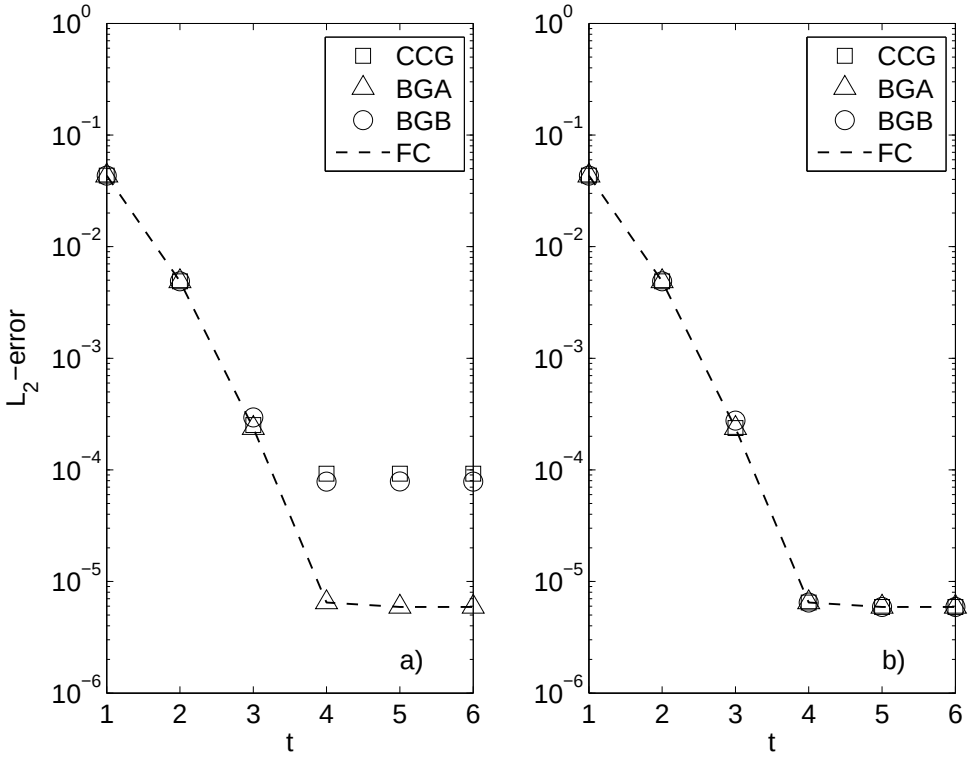


Figure 2.34: L_2 -error versus time for the collapsed-cartesian grid (CCG), barycentric grid of type A (BGA) and B (BGB) with (a) inconsistent numerical integration, and (b) exact numerical integration. The dash line (FC) depicts the fully consistent case; $\mathbf{u}^\delta \in \mathcal{V}^8$.

In figures 2.35 and 2.36 we show the dependence of the L_2-error for the different methods with respect to CPU-time. Here the CPU-time is measured in seconds and reflects the elapsed time from the beginning of simulation (first time step, t=0) to the intermediate time ($t = 1, 2, 3, 4, 5$) and final time $T = 6$. The CPU-time does not reflect the time consumed for construction of the linear operators, which is done only once at the beginning of the simulation. We recall that for all tests we use the same numerical scheme, the same initial condition, and the same size of a time step. However, the nonlinear term is computed using different grids, which results in distinct performance of the solver. In this test, the sum-factorization technique for numerical integration on CCG (2.9) was not

implemented. For solutions obtained on the cartesian grid and on the barycentric grid of type B we present the CPU-time for (a) inconsistent numerical integration and (b) consistent numerical integration. For solution on barycentric grid of a type A we show only one case, with the number of quadrature points consistent for differentiation but not sufficient for consistent integration of the nonlinear terms. The results in both figures show that at least for the Kovasznay flow we consider here the collapsed-cartesian grid is the most efficient, even without implementing the sum-factorization technique.

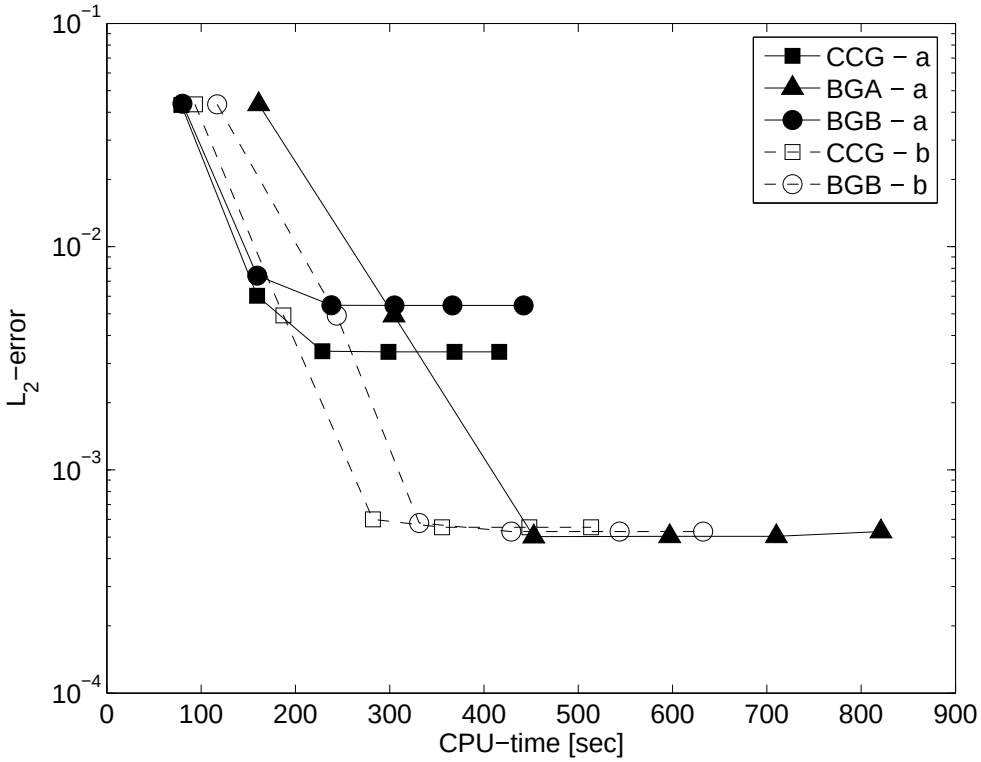


Figure 2.35: $L_2 - error$ vs. CPU-time: numerical solution of Kovasznay problem on collapsed-cartesian grid (CCG), barycentric grid of a type A (BGA) and B (BGB); inconsistent numerical integration of nonlinear term - solid line (a) and consistent numerical integration - dash line (b); $\mathbf{u}^\delta \in \mathcal{V}^6$. The computations were performed on an Intel(R) Xeon(TM) CPU 3.06GHz and 2GB of memory.

2.8.4 Stability

In this section we show that the stability criteria are practically independent of the grid type where the basic numerical operations are performed. However, when the size of timestep approaches its critical value, the BGA grid exhibits some advantages.

The Kovasznay problem was solved again using different spatial accuracy and variable

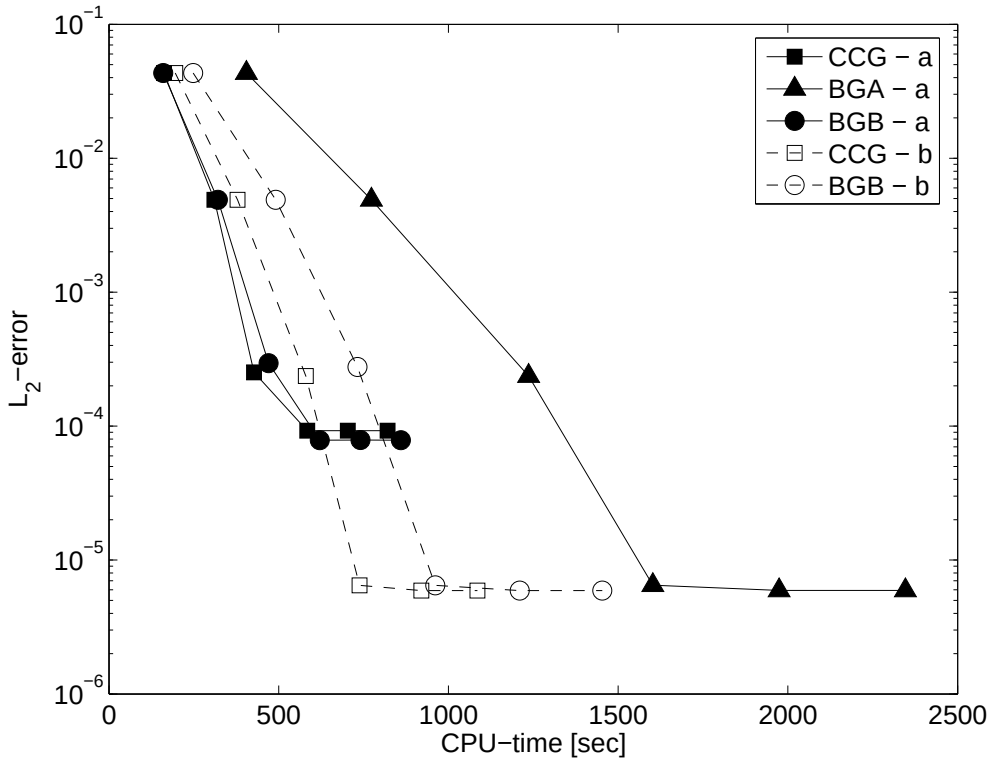


Figure 2.36: $L_2 - error$ vs. CPU-time: numerical solution of Kovasznay problem on collapsed-cartesian grid (CCG), barycentric grid of a type A (BGA) and B (BGB); inconsistent numerical integration of nonlinear term - solid line (a) and consistent numerical integration - dash line (b); $\mathbf{u}^\delta \in \mathcal{V}^8$. The computations were performed on an Intel(R) Xeon(TM) CPU 3.06GHz and 2GB of memory.

timestep. The conservative formulation of the nonlinear term was employed. The critical timestep for stability (Δt_{stable}) was numerically evaluated; in Table 2.3 we summarize the results in terms of the L_2-error in the numerical solution at $T = 6$ (steady state), obtained using different grids. In the second column we show the numerical error obtained using Δt_{stable} - one can see that the numerical error is not affected by a choice of grid and it is solely determined by the order of polynomial expansion, P, as expected. In the third column we present results obtained using marginally stable Δt; here the choice of a grid does affect the results. Columns four and five present results of simulation with slightly large size of Δt.

The relation between P and the critical timestep, Δt_{stable}, was obtained heuristically in the following way: For fixed order of polynomial expansion, P, the timestep was continuously increased until an instability in the numerical solution appeared; the instability was characterized by a *sudden* loss of accuracy. Then, Δt_{stable} was defined as the max-

P = 5

Grid Δt	2.500E-3	2.550E-3	2.600E-3	2.800E-3
CCG	5E-3	**4E-2**	1E+0	unstable
BGA	5E-3	**5E-3**	unstable	unstable
BGB	5E-3	**1E-2**	1E+0	unstable

P = 6

Grid Δt	1.730E-3	1.740E-3	1.750E-3	2.000E-3
CCG	4E-4	**2E-1**	5E-1	2E+0
BGA	3E-4	**3E-4**	unstable	unstable
BGB	4E-4	**4E-4**	6E-3	unstable

P = 7

Grid Δt	1.270E-3	1.275E-3	1.280E-3	1.290E-3
CCG	4E-5	**3E-1**	5E-1	unstable
BGA	4E-5	**4E-2**	5E-1	unstable
BGB	4E-5	**4E-2**	3E-1	unstable

P = 8

Grid Δt	9.735E-4	9.749E-4	10.00E-4	12.00E-4
CCG	5E-6	**2E-3**	1E+0	unstable
BGA	4E-6	**5E-6**	1E+0	unstable
BGB	4E-6	**6E-5**	1E+0	5E+1

P = 9

Grid Δt	7.600E-4	7.790E-4	8.000E-4	8.200E-3
CCG	3E-7	**3E-4**	1E+0	unstable
BGA	3E-7	**4E-7**	1E+0	unstable
BGB	3E-7	**9E-5**	2E+0	1E+1

Table 2.3: L_2-error for different timesteps. The number of grid points for CCG and BGB satisfies consistent numerical integration only but for BGA it satisfies consistent numerical differentiation only. $\mathbf{u}^\delta \in \mathcal{V}^i, i = 5, 6, 7, 8$. For $\mathbf{u}^\delta \in \mathcal{V}^9$ the number of quadrature points for BGB is sufficient for exact numerical integration of polynomials of order up to 25.

imum timestep for which the stable solution was obtained. The test was performed for $P = 5, 6, 7, 8, 9$ and was repeated for Adams-Bashforth time-integration of the nonlinear terms with first-, second- and third-order ($AB1$, $AB2$ and $AB3$, respectively). The dependence of Δt_{stable} on P for the three Adams-Bashforth methods is presented in Fig. 2.37. It is clear that AB1 (Euler-Forward) method with $\Delta t_{stable} \propto P^{-1.7}$ is the most stable (this will become clear from analyzing the eigenspectra, see below). Second, we observe that the stability region for barycentric grids and collapsed-cartesian grid is not exactly the same. This is due to the different *quadrature crimes* we commit using inconsistent integration or differentiation. In Table 2.4 we compare the L_2-error of a solution at $T = 6$ with consistent integration and unresolved/resolved differentiation (CCG) to the error of a solution with

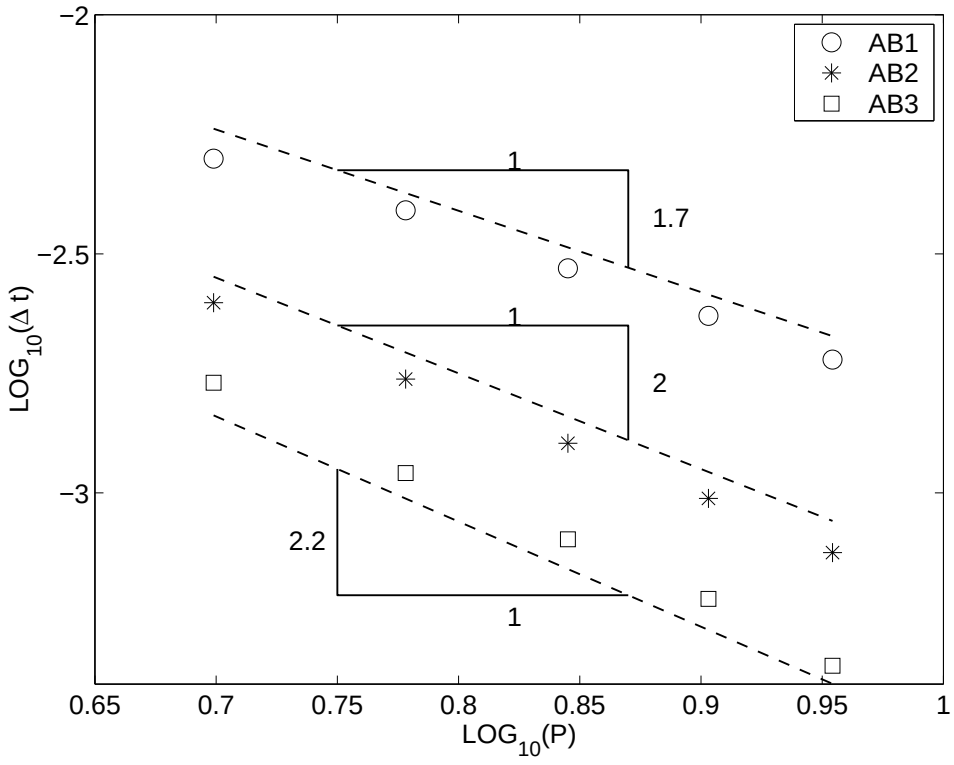


Figure 2.37: Stable Δt_{stable} versus P for solution of the Kovasznay problem. The nonlinear terms, fomulated in conservative form, were computed with Adams Bashforth method of order 1,2 and 3.

inconsistent integration but consistent differentiation (BGA); the improvement in L_2-error is clear.

The estimate of the numerical error in computing the nonlinear advection operator on a sparse grid can be obtained from the following. First, we project the exact solution of Kovasznay problem onto the space spanned by the basis functions. Second, we use *consistent* numerical differentiation to compute NL_x, NL_y (2.37). Third, we compute *exactly* the inner product of polynomial NL_x, NL_y terms with a test function, ϕ_k:

$$I_{exact} = (NL, \phi_k).$$

These three steps lead to an accurate projection of the nonlinear terms onto a modal space. Next, we compute the approximate inner product I_ϵ. The nonlinear terms are evaluated and projected using the CCG, BGA and BGB grids with consistent integration on the first two and consistent differentiation on the last one. We define the numerical error in evaluating

P = 6, $\Delta t = 1.74E-3$

	a) $P_d(CCG) < P_d(BGA)$	b) $P_d(CCG) = P_d(BGA)$
CCG	2E-1	3E-4
BGA	3E-4	3E-4

P = 8, $\Delta t = 9.749E-4$

	a) $P_d(CCG) < P_d(BGA)$	b) $P_d(CCG) = P_d(BGA)$
CCG	2E-3	6E-6
BGA	5E-6	5E-6

P = 9, $\Delta t = 7.79E-4$

	a) $P_d(CCG) < P_d(BGA)$	b) $P_d(CCG) = P_d(BGA)$
CCG	3E-4	4E-7
BGA	4E-7	4E-7

Table 2.4: L_2-error in Kovasznay solution computed on CCG and BGA. The number of quadrature points for BGA satisfies consistent numerical differentiation only. The number of quadrature points for CCG satisfies consistent numerical integration and (a) unresolved numerical differentiation ($P_d(CCG) < P_d(BGA)$), (b) resolved numerical differentiation ($P_d(CCG) = P_d(BGA)$). $\mathbf{u}^\delta \in \mathcal{V}^i, i = 6, 8, 9$.

the nonlinear advection operator as

$$\alpha_{NL} = MAX_{j,\ k}(I_{exact} - I_\epsilon), \quad j = 1, .., Nel, \ k = 1, ..., K$$

where Nel is a number of elements in the computational domain and $K = (P+1)(P+2)/2$ is the number of test functions. In Table 2.5 we present the values of α_{NL}; note that unresolved differentiation on CCG grid leads to the highest error.

	N	α_{NL}
CCG	144	8E-8
BGA	300	8E-10
BGB	105	5E-9

Table 2.5: Error in the nonlinear advection operator; $\mathbf{u}^\delta \in \mathcal{V}^8$

In the semi-implicit time-stepping we have employed, the stability of the three schemes is dictated by the eigenspectra of the advection operator and, of course, the type of the time-stepping method. Let us denote the advection operator by A and its eigenspectra by $\Lambda(A)$. Then, the perturbed advection operator, $A + \Delta A$, has $\Lambda(A + \Delta A) = \Lambda_\epsilon(A)$ eigenspectra, which is also known as *epsilon-pseudospectra* of A [92]. The definition of pseudospectra of a matrix A is given by

$$\Lambda_\epsilon(A) = z \in \mathcal{C} : ||(A - zI)v|| \leq \epsilon$$

for some $v \in \mathcal{C}^n$ with $||v|| = 1$. If some of eigenvalues of A are located in the vicinity of the region of instability, then the *epsilon-pseudo-eigenvalues* might be located inside the unstable region. The distance between the eigenvalues and the *epsilon-pseudo-eigenvalues* depends on $||\Delta A||$. In our case, $||\Delta A||$ is a result of inconsistent numerical evaluation of the advection operator.

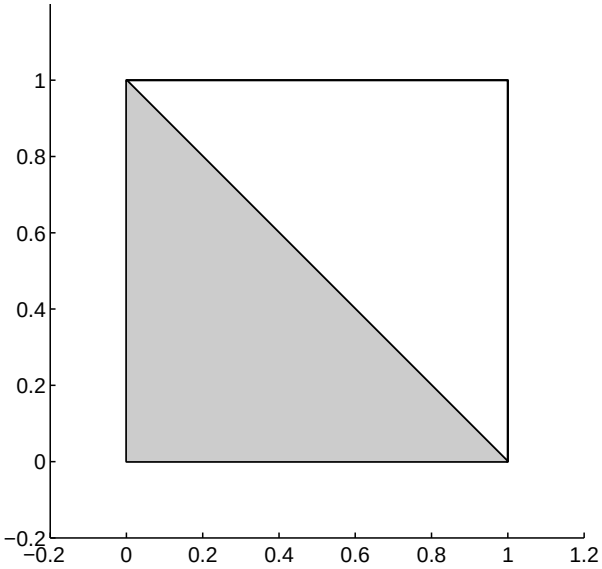


Figure 2.38: Two-element computational domain for solution of Kovasznay problem.

In order to analyze the eigenspectrum of the advection operator we simplify the problem by solving the Navier-Stokes equations on a computational domain with two elements as shown in figure 2.38. We use AB1 method for the nonliner terms. By gradual increase of Δt we experimentally obtain the critical timestep, Δt_{stable}, for which a stable solution is obtained. Then, we assemble the condensed global linear advection operator $A_G = M^{-1}D$, where

$$D = (U\frac{\partial u}{\partial x} + V\frac{\partial u}{\partial y}, \phi),$$

and the scalar coefficients U and V denote the maximum values of the corresponding velocity field, $\mathbf{u}^\delta$; M is a mass matrix. The local, elemental, advection operator is extracted from the global one and its eigenspectra is analyzed. We choose the element where U and V have their largest values. In Fig. 2.39 we present the scaled (by Δt_{stable}) eigenspectra of the *linearized* local advection operator. The dash line depicts the edge of a stability region for the AB1 time-stepping scheme. We note that the values of Δt_{stable} were obtained from the solution

of the *nonlinear* problem, with the advection terms computed with $O(\Delta t)$. For all numerical experiments the scaled eigenvalues fit exactly the region of stability, which suggests that we can analyze stability of the *nonlinear* problem using properly scaled eigensectra which correspond to the *linear* problem.

In Fig. 2.40 we present the pseudospectra of the linearized local advection operator A. Note that one of the eigenvalues is located in the neighborhood of the unstable region. The eigenspectra of the *linear* advection operator do not depend on the type of grid, since the number of grid points suffices to compute the operator exactly. However, the "effective" eigenspectra of the *nonlinear* operator is grid dependent due to different *quadrature crimes* we commit. The larger the numerical error associated with incorrect integration and differentiation, the larger the distance between eigenvalues of the perturbed operator and those of the exact operator. As seen in Table 2.5, the error in computing the nonlinear terms on the BGA grid is smaller than the error of the nonlinear terms computed on the CCG grid. The nonlinear advection operator is computed every timestep, which introduces a random shift to the eigenvalues of the perturbed operator from those of the exact one. It appears that the BGA grid minimizes these shifts and provides a more accurate bound on $\Delta t\Lambda(A + \Delta A)$. Thus, the transition from a stable region to an unstable one depends on Δt and less on ΔA as in the case of CCG.

2.9 Conclusions

In the first part of Chapter 2 (sections 2.1 - 2.6) the robustness of two tensor product spectral bases has been investigated. The first set of basis function (cartesian tensor product bases, denoted here by $\mathbf{\Phi}$) suggested by Sherwin&Karniadakis [79] has been studied in depth and desribed in the literature, while the second type of basis functions (barycentric tensor product bases, denoted here by $\mathbf{\Psi}$) proposed by Bittencourt in [22] is relatively new and many of its properties have not been investigated so far. The main fundings are:

- Both bases can be effectively implemented in the framework of spectral element/hp disctetization.
- The cost of construction projection operator using the barycentric bases can be significantly reduced due to the rotational symmetry of the bases.

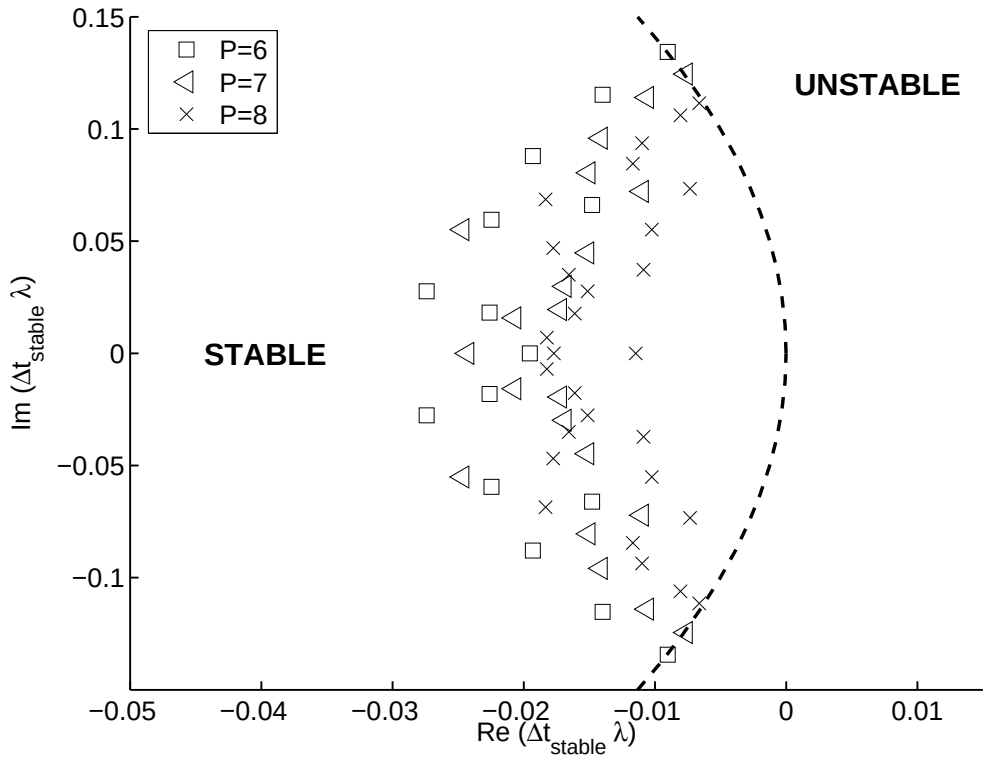


Figure 2.39: Eigenspectrum of linearized local advection operator. $\mathbf{u}^{\delta} \in \mathcal{V}^{i},\ i = 6, 7, 8.$

- The stability criteria in explicit solution of advection problem does not depend on a choice of basis.
- The linear operators constructed from the cartesian product bases have lower condition number and better sparsity, which is favorably reflected on the computational cost in iterative solution of partial differential equations.
- The projection and diffusion operators constructed from both bases and preconditioned by the Incomplete Cholesky Preconditioner (ICP) show similar eigenproperties. In iterative solution of diffusion problem with conjugate gradient solver lower iteration count can be achieved by employing the barycentric bases and ICP, particularly in the case of very distorted triangular elements.

In section 2.7 the $P/P-k$ approach for solution of Navier-Stokes problem has been investigated. Specifically, we focused on the numerical accuracy and filtering of high-frequency pressure oscillations arising due to singularities and explicit treatment of pressure bound-

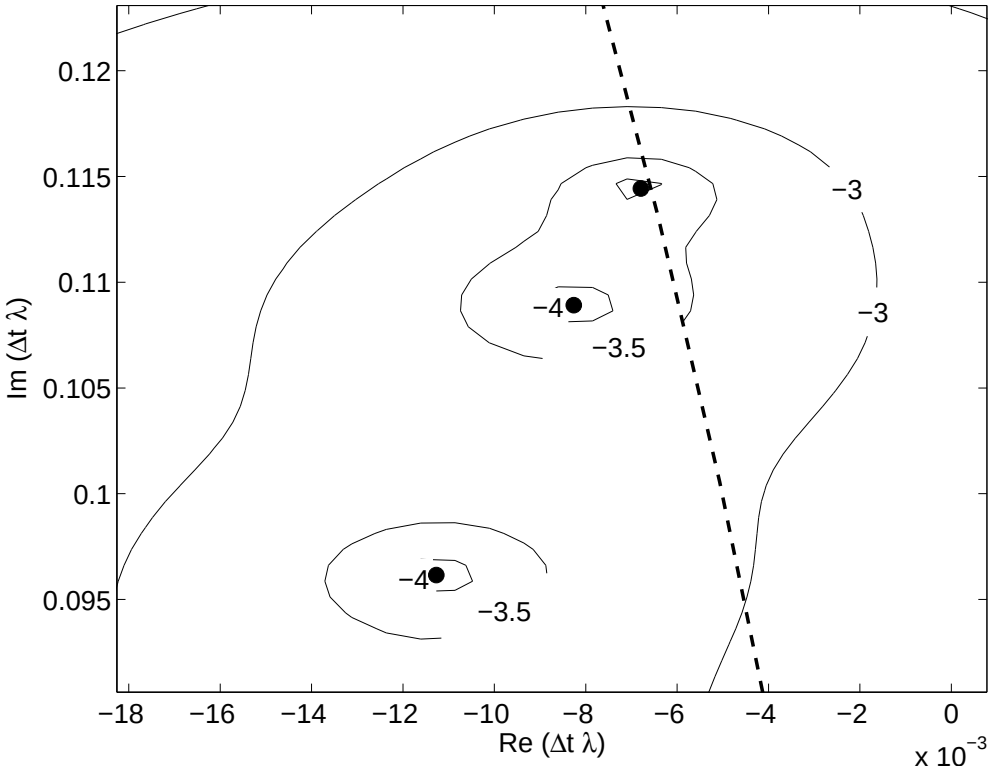


Figure 2.40: Pseudospectra of a linear local advection operator, A, computed on a triangular element: dots - represent the eigenvalues of A, solid lines - contours of Log_{10} of *epsilon-pseudo-spectra* for $\epsilon = 10^{-4}, 10^{-3.5}, 10^{-3}$, dash line - edge of stability region; $\mathbf{u}^{\delta} \in \mathcal{V}^8$

ary conditions. Robustness of a semi-implicit and implicit numerical schemes have been compared. The main fundings are:

- Solution of Navier-Stokes problem with $P/P-k,\ k>0$ discretization suffers from lower accuracy than solution with P/P discretization.

- In solution of problems with singularities the use of lower order polynomial expansion for discretization of the pressure variable than for discretization of the velocity variables, indeed leads to filtering high-frequency pressure oscillations. It was observed that the pressure oscillations appear due to the time-splitting error corresponding to semi-implicit scheme, and specifically due to explicit treatment of Neumann boundary condition for the pressure. Implicit numerical scheme does not posses any time-splitting error and pressure field computed with the implicit solver does not exhibit erroneous high-frequency oscillations. Filtering high-frequency pressure oscillations might have positive effect in solution of fluid-structure interaction (FSI) problems where the main challenge is rather stability than accuracy. In FSI solvers based on

weak coupling between the fluid and the structure domain, pressure computed at the boundaries of fluid domain is incorporated in boundary conditions for the structure solver.

In section 2.8 we have employed collapsed cartesian and barycentric grids, typically used in collocation spectral methods on a triangle, to perform numerical integration, differentiation and projection in the Galerkin spectral/*hp* element method. We have examined the individual operations separately as well as in the context of solving the Navier-Stokes equations for incompressible flows. The main findings of this study are the following:

- All three quadrature grids investigated can be used in constructing the discrete operators.
- The choice of the grid, (e.g., symmetric versus non-symmetric) does not affect the accuracy of the numerical solution as long as a sufficient number of quadrature points is used.
- BGB leads to high efficiency but the limited number of grid points defined for multivariate quadrature restricts the highest order of polynomial expansion.
- In solution of nonlinear problems with Galerkin projection the loss of accuracy can be a result of underresolution in *both* numerical integration and differentiation. However, in the solution of the model equation chosen in this paper, if at least one of these numerical operations is consistent the overall accuracy is maintained.
- The stability properties are nearly independent of the grid types. The performance of barycentric grids is slightly better when the size of a timestep approaches its critical value, while the performance of the collapsed-cartesian grid is the worst due to the *quadrature crimes* we commit due to unresolved differentiation of the nonlinear terms when a conservative form is used.
- Using different grids deviation of several orders of accuracy in the error of a stable numerical solution may occur. This happens when the size of the time step approaches its critical value in terms of stability. Quadrature crimes we commit in differentiation and integration practically shrink the stability region.

- The collapsed-cartesian grid is overall the most efficient, particularly for high order of polynomial expansions.

Chapter 3

Parallel performance of iterative solver for high-order spectral/hp element method

3.1 Introduction

Efficiency in solving linear system of equations $\mathbf{Ax} = \mathbf{b}$ iteratively depends primarely on three factors: a) a choice of numerical method, e.g., GMRES, Conjugate Gradient, etc.; b) condition number of the operator $\mathbf{A}$; and c) a distance between the initial guess $\mathbf{x}^0$ and the solution $\mathbf{x}$. The choice of the numerical method and a proper preconditioner are directly related to the properties of the operator $\mathbf{A}$. Techniques for minimization $|\mathbf{x}^0 - \mathbf{x}|$ are derived from the solution properties. In this study we consider a high-order spectral element discretization, leading to a positive definite symmetric operator $\mathbf{A}$ and solution of a linear system with a preconditioned Conjugate Gradient method. Our goal is to design *fast* and *scalable* numerical solver, for solution of multi-million (or even billion) degrees of freedom problem. We focus here on two topics: a) parallel preconditioner designed to improve *parallel efficiency* of a conjugate gradient solver, and b) methods for accurate prediction of the initial guess (that reduces the number of conjugate gradient iterations) while imposing negligible computational overhead.

The spectral/hp element code $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ is employed in our studies [79]. The computational domain consists of structured or unstructured grids or a combination of both,

similar to those employed in standard finite element and finite volume methods. In each element the solution is approximated in terms of hierarchical mixed order Jacobi polynomial expansions. This provides a way of hierarchically refining the numerical solution by increasing the order of the expansion (p-refinement) within every element without the need to regenerate the mesh, thus avoiding a potential significant overhead cost associated with remeshing. Domain partitioning, required by the parallel solver, is done by Metis [3]. $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ employs up to third-order accurate semi-implicit time integration scheme (2.34a). A high-order splitting scheme [48] is adopted, that decouples the velocity and pressure fields requiring only the inversion of three Helmholtz operators for the velocity components (in three-dimensions) and one Poisson operator for the pressure. Due to the matrix sparsity arising from symmetric linear operators, iterative solver based on Preconditioned Conjugate Gradient (PCG) is preferred. The *effectiveness* of preconditioner is normally estimated by reduction in iteration count. However, the *parallel efficiency* of a given preconditioner is also strongly affected by the additional computational cost associated with the preconditioning step and by the volume of communication involved. In this study we measure the effectiveness of preconditioner by monitoring the reduction in the average CPU-time required for one time step in large 3D simulations of flow in arterial geometries. The effectiveness of initial guess prediction is measured by reduction of the initial residual $r^0 = ||\mathbf{A}\mathbf{x}^0 - \mathbf{b}||$ and corresponding reduction in number of iterations.

The following computational platforms were used for the benchmarking and code developing:

- *IBM Blue Gene of San Diego Supercomputing Center (SDSC).* This computer is housed in three racks with 3,072 compute nodes. Each node has two PowerPC processors that run at 700 MHz and share 512 MB of memory. All compute nodes are connected by two high-speed networks: a 3-D torus for point-to-point message passing and a global tree for collective message passing.

- *IBM Power4+/Federation of SDSC.* This computer has 8-way P655+ 1.5 GHz and 32-way P690 1.7GHz compute nodes with 32 and 128 GB of memory, respectively. In our study we used the P655 compute nodes.

- *Cray XT3 MPP system of Pittsburgh Supercomputing Center (PSC)* with 2068 compute nodes linked by a custom-designed interconnect. Each compute node has two

2.6 GHz AMD Opteron processors with its own cache, and shared 2 GB of memory and the network connection. The nodes are connected in a three-dimensional torus using a HyperTransport link to a dedicated Cray SeaStar communications engine.

- *Cray XT3 MPP of U.S. Army Engineer Research and Development Center (ERDC).* The computer has 4,160 nodes, each containing one 2.6-GHz AMD Opteron 64-bit dual-core processor and 4GB of shared memory. The nodes are connected in a three-dimensional torus using a HyperTransport link to a dedicated Cray SeaStar communications engine.

The Chapter is organized as follows: In section 3.2 we review the existing scalable parallel solvers and preconditioners and specify the objectives for the current study. In section 3.3 we overview the Low Energy Bases Preconditioner. In section 3.4 we focus on the coarse space linear vertex solver, specifically we discuss the algorithmic aspects of parallel implementation of the solver. In section 3.5 we discuss two acceleration techniques designed to predict better initial state for iterative solvers. In section 3.6 we conclude with a brief summary. In Appendices A and B we provide additional information on construction of LEBP and communication paradigms implemented for parallel matrix-vector product in $\mathcal{NEKTAR}$.

List of Symbols/Notations

Ω - computational domain.

Ω_e - computational domain corresponding to spectral element e.

$\mathbf{x}$ - cartesian coordinate system or solution space, depends on context.

ξ - coordinates of cartesian system, defined on Ω_e.

η - collapsed coordinate system.

P - order of polynomial expansion.

$\mathbf{\Phi}$ - global polynomial basis.

ϕ^e - polynomial bases defined on Ω_e.

$\mathbf{\Psi}$ - POD spatial basis.

$\mathbf{u} = [u\ v\ w]$ - velocity field.

$\mathbf{C}$ - correlation matrix.

$\mathbf{\Lambda} = [\lambda_1, ..., \lambda_Q]$ - eigenvalues of $\mathbf{C}$.

3.2 Low Energy Bases Preconditioner and Coarse Space Linear Vertex Solver: Introduction

Scalability of highly effective preconditioners on thousands of processors is an open problem that had no satisfactory solution for many matrix problems arising from the discretisation of partial differential equations. As the effectiveness of the preconditioner increases so does the global interdependencies reflecting in a sense the multi-scale nature of the problem and hence the poor scaling. To the best of our knowledge, there are currently no effective preconditioners for the new generation of *modal* spectral/*hp* element discretisations that scale well on more than one thousand processors.

Several effective preconditioners for spectral element Navier-Stokes solvers can be found in the literature, but very few of these preconditioners are also scalable. Tufo and Fischer [93] presented a scalable parallel solver for the incompressible Navier-Stokes equation, for the first (*nodal*) generation of spectral elements. Performance of the solver was evaluated on the Intel ASCI-Red, CRAY T3E-600 and SGI ASCI-Blue computers with 333MHz processors. Good scalability was observed for simulations with relatively high order polynomial approximation. Bergen *et al.* [18] used Hierarchical Hybrid Grid (HHG) to solve efficiently large scale linear finite element problem. The HHG approach is essentially a geometric

multigrid method. Good scalability and also good solution time was achieved for solution of a large problem on up to 1024 SGI Atlix^{TM}3700 CPUs (1.6 GHz Itanium 2 processors). Lottes and Fischer [57] studied performance of the variations of multigrid method applied to spectral element *nodal* discretizations of the Helmholtz equation. Several methods considered in their work resulted in convergence rates comparable to regular (Cartesian) grid based multigrid methods. Although effective for *non-deformed* spectral elements, geometric multigrid is essentially a sequential approach and can not lead to a high parallel efficiency on petaflop computers. Pavarino *et. al.* [69] developed overlapping Shwarz methods for nodal triangular and quadrilateral spectral elements. Their results show that it is possible to obtain convergence rate independent of the order of polynomial expansion and number of elements.

A Low Energy Basis Preconditioner (LEBP) for elliptic substructured solvers was proposed by Bica [21] and later implemented by Sherwin and Casarin [76] for modal spectral/*hp* elements. In this work we present a parallel implementation of the LEBP appropriate for a large number of processors and investigate its performance. Specifically, we discuss in detail implementation of a parallel coarse space linear vertex solver. Our results show that LEBP is very effective in simulations on thousands of processors while it exhibits similar scalability to a Diagonal Preconditioner. The goal of this project was to develop a *scalable* and *efficient* coarse space linear vertex solver required by a LEBP for elliptic substructured solver. Guided by the information obtained with profiling tools on IBM Blue Gene/L and DataStar (Power4+/Federation) computer at SDSC we were able to make optimizations by redesigning the communication in several bottleneck routines. Profiling code simultaneously on several computational platforms allowed us to perform differential optimization. As a result, the code implemented with Low Energy Basis Preconditioner and the optimized coarse space linear vertex solver now scales well on thousands of processors.

3.3 Low Energy Basis Preconditioner for Spectral/*hp* Elements

The computational domain in $\mathcal{NEKTAR}$ consists of tetrahedra, hexahedra, prisms, pyramids or a combination of these. Within each element the solution is approximated in terms of hierarchical, mixed order, semi-orthogonal Jacobi polynomial expansions [79]. It is hierarchical in a sense that the modes are separated into vertex (linear term), edge, face and bubble (interior) modes. In figure 3.1 we provide an illustration of the domain decomposition and polynomial basis employed in $\mathcal{NEKTAR}$.

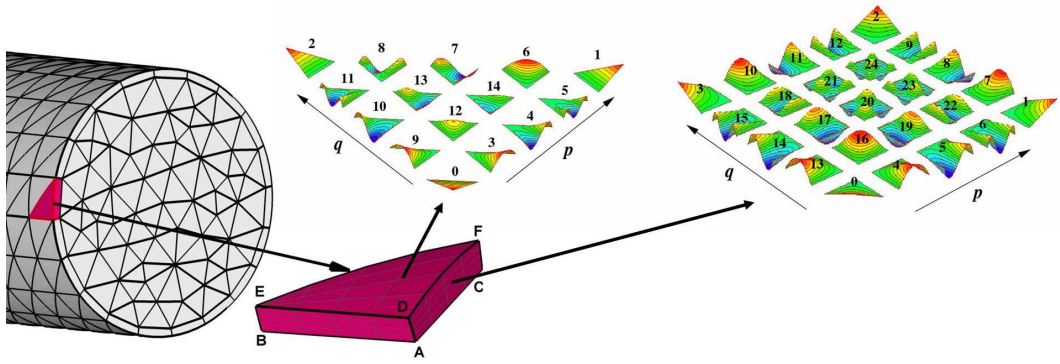


Figure 3.1: Illustration of the unstructured surface grid and the polynomial basis employed in $\mathcal{NEKTAR}$. The solution domain is decomposed into nonoverlaping elements. Within each element the solution is approximated by vertex, edge, face and (in 3D) interior modes. The shape functions associated with the vertex, edge and face modes for fourth-order polynomial expansion defined on triangular and quadrilateral elements are shown in color.

The polynomial expansion basis within each element is decomposed into interior and boundary modes (vertex, edge and face) to help construct a global C^0-continuity. The interior modes have zero support on the elemental boundaries, thus the boundary and interior degrees of freedom can be numerically decoupled through a technique known as *substructuring* where the Schur complement of the boundary system is constructed. The boundary degrees of freedom, corresponding to adjacent elements, are coupled due the requirement of C^0-continuity. Moreover, there is a strong coupling between the vertex, edge and face modes within each element. It is this strong coupling between the boundary degrees of freedom that leads to solution of large linear system with high condition number.

The dependence of the condition number of the preconditioned Schur complement of the Laplacian matrix on the order of polynomial approximation was reported in [76]. The

condition number of 3D Laplacian matrix scales with the order of polynomial expansion P as $\kappa \propto P^{3.3}$ and $\kappa \propto P^{3.2}$ for tetrahedral and prismatic meshes with fixed number of elements reciprocally, whereas the condition number of the preconditioned (with LEBP) Schur complement scales as $\kappa \propto (1 + lg(P))^2$. Bica [21] and Pavarino & Widlund [68] presented a theoretical proof and numerical verification for that polylogarithmic growth of κ for solution of 3D problems with the p-version finite element method based on continuous, piecewise polynomial expansions.

As we will see in the next section, the idea of the LEBP is to *weaken the coupling* of the boundary modes, enhance the diagonal dominance of the matrix system and then to apply a block-preconditioning technique for the vertex, edge and face degrees of freedom.

3.3.1 Low Energy Basis Preconditioner: formulation

Consider the elliptic boundary value problem

$$\nabla^2 u(\mathbf{x}) + \lambda u(\mathbf{x}) = f(\mathbf{x}); \qquad \lambda \leq 0, \tag{3.1}$$

defined on computational domain Ω which is discretized into N_{el} non-overlapping spectral elements; the computational sub-domain, associated with a particular element, is denoted by Ω_e. A standard spectral/*hp* element [79] is defined on a *local*, to this element, system of coordinates ξ, which can be mapped to the global coordinate system $\mathbf{x} = \mathbf{x}_e(\xi)$. Then, a standard spectral/*hp* element [79] spatial approximation of u is given by

$$u^\delta(\mathbf{x}) = \sum_{i=1}^{N_{dof}} \hat{u}_i \Phi_i(\mathbf{x}) = \sum_{e=1}^{N_{el}} \sum_{i=1}^{dim(\mathcal{V}^\delta)} \hat{u}_i^e \phi_i^e(\mathbf{x_e}(\xi)), \tag{3.2}$$

where N_{dof} is a total number of degrees of freedom and $\phi_i(\mathbf{x}(\xi))$ are polynomials defined in a space $(\mathcal{V}^\delta)$ of order P, which when pieced together under the mapping $\mathbf{x}(\xi)$ make a C^0 continuous (global) expansion $\Phi(\mathbf{x})$. The superscript δ emphasize that we use a finite (truncated) space.

Using Galerkin discretization of (3.1), i.e. find $u^\delta \in \mathcal{V}^\delta$ such that

$$\mathcal{L}(v, u) = \int_\Omega \nabla v^\delta \cdot \nabla u^\delta + \lambda v^\delta u^\delta d(\mathbf{x}) = \int_\Omega v^\delta \, f d(\mathbf{x}) \qquad \forall v^\delta \in \mathcal{V}^\delta \tag{3.3}$$

we obtain the weak formulation of (3.1). Following the standard Galerkin formulation adopted in finite element methods and letting $v^\delta = \Phi_i (i = 1, \ldots, dim(\mathcal{V}^\delta))$, problem (3.1) can be recast into a matrix equation

$$\mathbf{H}\hat{\mathbf{u}} = \mathbf{f},$$

where $\mathbf{H}(i,j) = \int_\Omega \nabla\Phi_i(\mathbf{x})\nabla\Phi_j(\mathbf{x}) + \lambda\Phi_i(\mathbf{x})\Phi_j(\mathbf{x})d(\mathbf{x})$ denotes the Helmholtz operator and $\mathbf{f}(i) = \int_\Omega \Phi_i(\mathbf{x}) f \, d(\mathbf{x})$.

In terms of implementation, the global matrix $\mathbf{H}$ is typically assembled from elemental contributions $\mathbf{H}^e$ where

$$\mathbf{H}^e(i,j) = \int_{\Omega^e} \nabla\phi_i^e(\mathbf{x})\nabla\phi_j^e(\mathbf{x}) + \lambda\phi_i^e(\mathbf{x})\phi_j^e(\mathbf{x})\, d(\mathbf{x})$$

and $\phi_i^e(\mathbf{x})$ denote the elemental representations of the elemental expansion which when assembled globally make $\Phi_j(\mathbf{x})$ [79]. The elemental approximation, u_e^δ, can be expressed by different sets of polynomial expansion from the same space ($\mathcal{V}^\delta$)

$$u_e^\delta(\mathbf{x}(\xi)) = \sum_{i=1}^{dim(\mathcal{V}^\delta)} \hat{u}_{1_i}\phi_{1i}^e(\xi) \equiv \sum_{i=1}^{dim(\mathcal{V}^\delta)} \hat{u}_{2_i}\phi_{2i}^e(\xi), \tag{3.4}$$

and therefore it is possible to define a matrix transform $\mathbf{C}$ from basis ϕ_1^e to ϕ_2^e, i.e.,

$$\phi_2^e = \mathbf{C}\phi_1^e,$$

and this process is possible even if a close form expression for ϕ_2^e is not available. Due to the linearity of the operation, $\mathbf{C}$ can be applied to transform the operator $\mathbf{H}^e$ computed with different expansion bases, i.e.,

$$\mathbf{H}_2^e = \mathbf{C}\mathbf{H}_1^e\mathbf{C}^T, \quad \mathbf{H}_1^e = \mathbf{C}^{-1}\mathbf{H}_2^e(\mathbf{C}^T)^{-1}.$$

Analogous to the elemental decomposition of $\mathbf{H}$ into components $\mathbf{H}^e$, we introduce elemental contributions to $\hat{\mathbf{u}}$ and $\mathbf{f}$ denoted as $\hat{\mathbf{u}}^\mathbf{e}$ and $\mathbf{f}^\mathbf{e}$. Further decomposing $\hat{\mathbf{u}}^\mathbf{e}$ and $\mathbf{f}^\mathbf{e}$ into contributions associated with the boundary ($\hat{\mathbf{u}}_\mathbf{b}^\mathbf{e}$) and interior ($\hat{\mathbf{u}}_\mathbf{i}^\mathbf{e}$) modes we obtain:

$$\begin{bmatrix} \mathbf{H}_{\mathbf{bb}}^\mathbf{e} & \mathbf{H}_{\mathbf{bi}}^\mathbf{e} \\ \mathbf{H}_{\mathbf{ib}}^\mathbf{e} & \mathbf{H}_{\mathbf{ii}}^\mathbf{e} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_\mathbf{b}^\mathbf{e} \\ \hat{\mathbf{u}}_\mathbf{i}^\mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\mathbf{b}^\mathbf{e} \\ \mathbf{f}_\mathbf{i}^\mathbf{e} \end{bmatrix}. \tag{3.5}$$

Due to non-overlapping support of the interior modes, the contributions of $\mathbf{H_{ii}^e}$ to $\mathbf{H}$ are decoupled from each other. It is therefore an excepted practice to construct the Schur complement $\mathbf{S^e} = \mathbf{H_{bb}^e} - \mathbf{H_{bi}^e}[\mathbf{H_{ii}^e}]^{-1}\mathbf{H_{ib}^e}$ and solve directly for the boundary degrees of freedom, which when known can be used to recover the interior degrees of freedom.

We therefore restrict our discussion to preconditioning only the Schur complement system and define a transformation matrix $\mathbf{C}$ as

$$\mathbf{C} = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{I} \end{bmatrix},$$

then, after substructuring the matrix system $\mathbf{H_2^e} = \mathbf{C}\mathbf{H_1^e}\mathbf{C^T}$, we obtain that the Schur complement of $\mathbf{H_2^e}$ is related to the Schur complement of $\mathbf{H_1^e}$ by

$$\mathbf{S_2^e} = \mathbf{R}\mathbf{S_1^e}\mathbf{R^T},$$

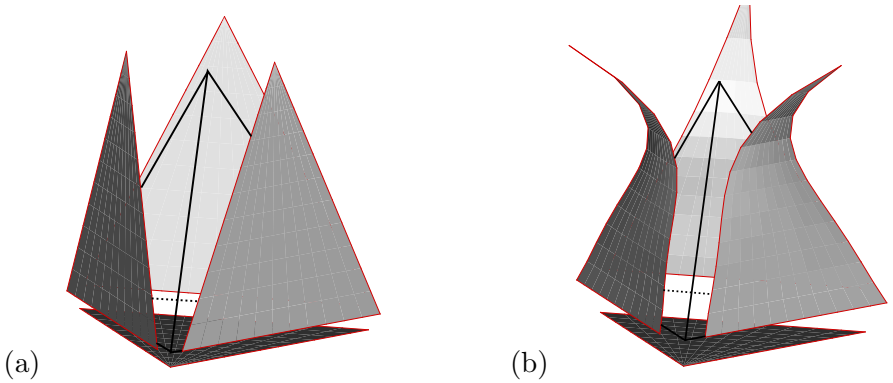
where $\mathbf{S_1^e}, \mathbf{S_2^e}$ are the Schur complements of $\mathbf{H_1^e}$ and $\mathbf{H_2^e}$, respectively. Our task is now to define an appropriate transformation matrix $\mathbf{R}$ in order to obtain $\mathbf{S_2^e}$ with a predominantly diagonal structure. Details on the structure and numerical construction of the matrix $\mathbf{R}$ can be found in Appendix A and in [76].

We note that in order to reduce significantly the number of iterations, we want to design a preconditioner whose eigenspectrum is close to the one of the original operator. However, the efficiency of a parallel solver is also strongly affected by the additional computational and communication cost associated with the preconditioning. Therefore, the *block-diagonal* dominance of the modified Schur complement is essential.

The LEBP is developed by numerically constructing a "low energy" basis,
(i.e., $\int \phi_{2i}^e \phi_{2j}^e d\Omega_e < \int \phi_{1i}^e \phi_{1j}^e d\Omega_e$, particularly for $i \neq j$) related to the transformation matrix $\mathbf{R}$. This is achieved by considering the elemental matrices arising for a given spectral/hp problem within a standardized region and *numerically constructing* a new low energy basis, where coupling between each vertex mode with respect to the edge and face modes, and also the edge modes with respect to the face modes is minimized. Due to the influence of local elemental mappings $\mathbf{x}_e(\xi)$ the transformed global matrix $\mathbf{H}$ will not maintain the orthogonality introduced by the low energy basis on $\mathbf{H}^e$. However, the transformed system is now diagonally dominant and so block diagonal preconditioning of this transformed matrix

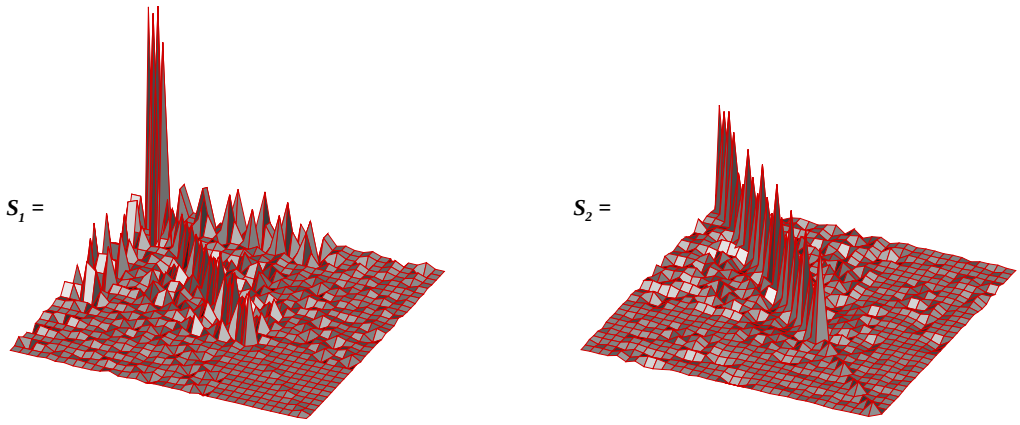
leads to a polylogarithmic scaling of the condition number with respect to polynomial order P for reasonably well behaved mappings. Since the number of iterations of the conjugate gradient methods scales as a square root of the condition number this approach leads to an effective $p-type$ preconditioner. Further, the *local* communication associated with edges and faces means that this preconditioner is very suitable for parallelization.

An example of numerically-derived basis for a low energy vertex mode is shown in figure 3.2 [76]. The standard linear finite element vertex mode is shown in figure 3.2(a) and the numerically constructed low-energy vertex mode is shown in figure 3.2(b). As depicted in figure 3.3 the transformed Schur complement system $\mathbf{S_2^e}$ has much more diagonal-dominant structure than the original $\mathbf{S_1^e}$.



(a) (b)

Figure 3.2: Projected mode shape for the vertex mode in a $P = 5$ polynomial expansion (a) original basis and (b) low energy.



(a) (b)

Figure 3.3: Scatter plot of Schur complement matrices of a $P = 5$ polynomial expansion: (a) Original Basis (b) Low Energy Basis (scaled by a factor of 4).

When constructing the global Schur complement $\mathbf{S}$ from $\mathbf{S}^e$, we now choose to design a

preconditioner which inverts blocks corresponding to degrees of freedom along each global edge and face since these are relatively small and easily inverted.

The $h-scaling$ is most effectively handled by directly inverting the coarse linear finite element space block associated with the spectral/hp discretization; here we define the coarse linear finite element space as a space of the *vertex* degrees of freedom. The definition of the coarse linear finite element space block is given in 3.4.1. Thus, the additive Schwartz preconditioner is a combination of the coarse space linear vertex block and the block-diagonal LEBP, as described in section 2.6 of [76]. Since in these types of methods we typically have quite coarse grids compared to classical finite element descritization, a direct inversion is still tractable from a memory standpoint (we should note that methods for direct solution of the equations on a linear finite element mesh may be also based on approaches we describe in this study). However, extra algorithmic work is required to make this coarse space linear vertex solve scale from a parallelization/communication point of view. In section 3.4 we discuss the implementation of parallel direct solver associated with the coarse grid preconditioning.

3.3.2 Low Energy Basis Preconditioner for prismatic elements

For many applications, e.g. in boundary layers, we use a layer of prismatic elements to capture the boundary layer of the flow. The interior space is then filled with tetrahedrons. In this section we present an improvement in constructing LEBP for prismatic elements used in this type of meshing strategy. In the original work of [76] the construction of LEBP was based on considering a standard prismatic element, which has two equilateral triangular surfaces connected by three quadratic faces which all have edges of a similar length. The equilateral shape of the triangular face was to ensure the global continuity of the Low Energy shape functions between the prismatic and tetrahedral elements.

In the following, we modify the aspect ratio of the standard prismatic element by varying the distance between the triangular faces (thickness of the prismatic element). The reshaping of element results in a better fit of the standard element to that used in the physical mesh. We denote by α the parameter by which we scale the distance between triangular faces of the standard prismatic element. In general, not all elements in the original mesh have the same aspect ratio, which makes adjustment of α not trivial.

To check the effect of α on the of convergence rate we performed a steady flow simulation

in the domain shown in figure 3.1. The mesh is constructed from one layer of prismatic elements while the rest are tetrahedral elements. The dimensions of the prismatic element are summarized in table 3.1. In figure 3.4 we present the number of iterations required by the Poisson solver for the pressure and Helmholtz solver for the streamwise (w) velocity component. In figure 3.5 we show the number of iterations required by the linear solvers

Edge:	AB	AC	AD	DE	DF
Length (original):	0.717	0.694	0.133	0.717	0.738
Length (standard):	1	1	α	1	1

Table 3.1: Dimensions of prismatic element shown in figure 3.1.

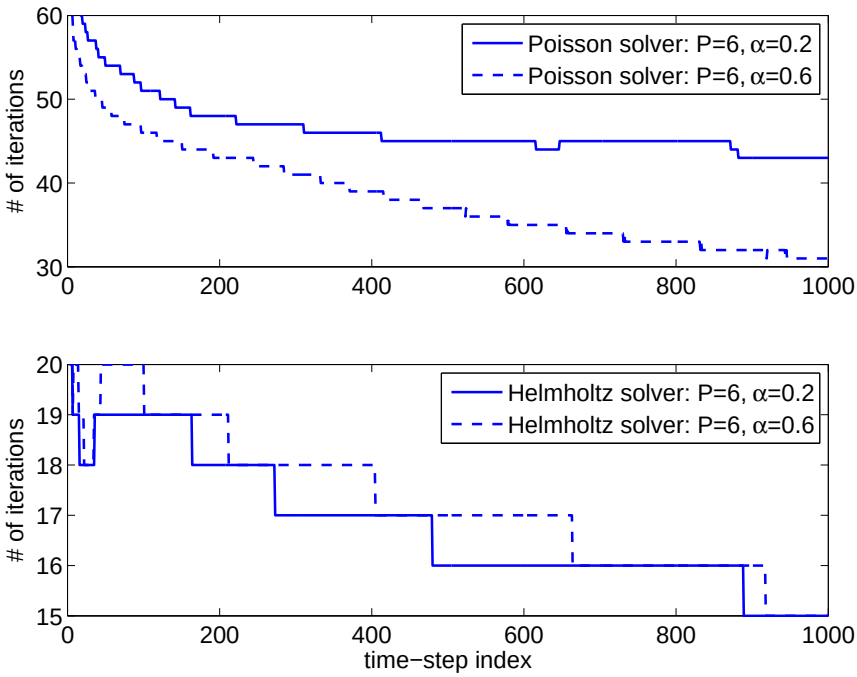


Figure 3.4: Hybrid mesh: performance of LEBP versus time step as a function of parameter α for Poisson (upper) and Helmholtz (lower) solvers. Simulation of a steady flow in a domain presented in figure 3.1.

with respect to polynomial order and parameter α. In a case of elemental Mass matrix, the factor α scales the Jacobian of the standard element only, which is equivalent to scaling the Mass matrix with a constant. This scaling does not modify the ratio of the largest to smallest eigenvalues $\lambda_{MAX}/\lambda_{MIN}$. If all elements in the mesh have the same Jacobian and the same scaling factor α, then the condition number of the global, statically condensed Mass matrix, will not be affected. In the case of Stiffness matrix, the parameter α appears not only in the Jacobian but its inverse appears also in the derivatives taken along the

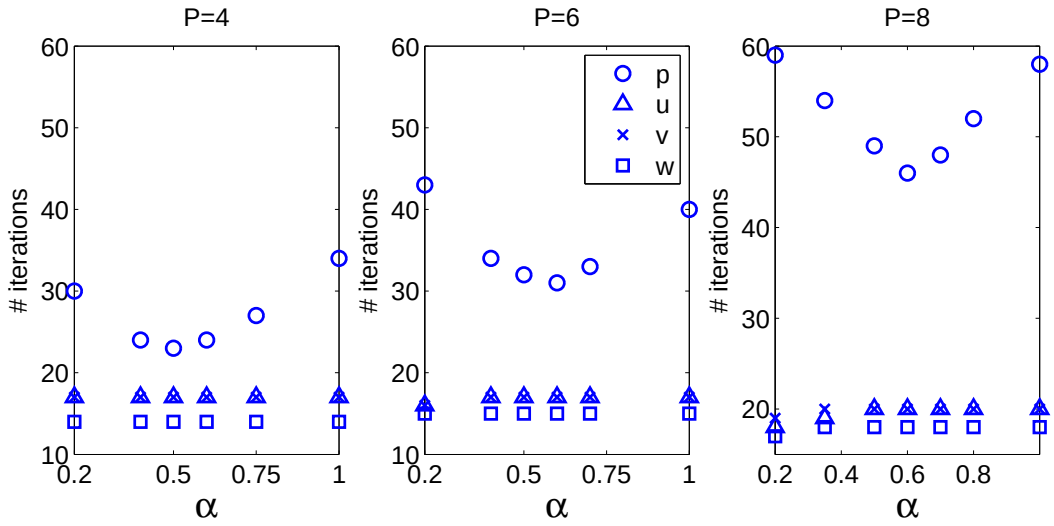


Figure 3.5: Hybrid mesh: performance of LEBP as a function of parameter α. Simulation of a steady flow in a domain presented in figure 3.1. Mean number of iterations required by the last ten time-steps of figure 3.1.

AD direction (see figure 3.1), thus it affects the $\lambda_{MAX}/\lambda_{MIN}$ ratio. In a hybrid mesh, where the Jacobian of all prismatic elements are scaled by α while the Jacobian of the tetrahedral elements remains unmodified, the condition number of the global Mass and Stiffness matrices changes. The Helmholtz operator is a weighted linear combination of the Mass and the Stiffness operators, hence its condition number will depend on both matrices.

As we observe in figure 3.5 the number of iterations required to solve the Helmholtz equations for the three velocity components is not very sensitive to α, while the number of iterations required by the Poisson solver strongly depends on α. For these types of meshes, the thickness of the standard prismatic element leads to a better approximation of the shape of the original elements and, as a result, to a lower iteration count, ultimately leading to computational savings.

3.4 Parallel Coarse Space Linear Vertex Solver

In this section we first discuss the partitioning of global linear vertex degrees of freedom and formulation of the Schur complement for the coarse vertex solve. Second, we compare several numerical approaches to tackle the solution of a linear system required by the coarse vertex solve. Third, we provide details on the parallel construction of the Schur complement for the coarse vertex solve during the preprocessing stage. Finally, we discuss different algorithms for parallel matrix-vertex multiplication and examine the load-balancing with respect to different implementations of communication required by the multiplication.

3.4.1 Formulation

Parallel implementation of the coarse space vertex degrees of freedom solver requires partitioning the global domain into non-overlapping groups of elements. To minimize the volume of communication between different groups it is preferable to form a partition of one or few clusters of adjacent elements; in our code this task is performed by Metis [3]. Given such compact partitioning of elements, we can identify another boundary-interior decomposition with respect to the vertex degrees of freedoms. This decomposition is highlighted in figure 3.6 where we consider an illustrative example of a partitioning of a 2D triangular computational domain into four partitions; an analogous extension to 3D domain is reasonably straight-forward. In this figure the interfaces between partitions are identified by wide lines. Vertices shared by adjacent elements are sub-divided into two groups: (a) boundary-boundary vertices (shown as squares), located on the interfaces between partitions, and (b) interior-interior vertices (shown as circles), located inside each partition.

Our aim is to precondition the global Schur complement system $\mathbf{S}$. We can order $\mathbf{S}$ by vertex, edge and face degrees of freedom so as to obtain a matrix structure of the form

$$\mathbf{S} = \begin{bmatrix} \mathbf{S_{vv}} & \mathbf{S_{ve}} & \mathbf{S_{vf}} \\ \mathbf{S_{ve}^T} & \mathbf{S_{ee}} & \mathbf{S_{ef}} \\ \mathbf{S_{vf}^T} & \mathbf{S_{ef}^T} & \mathbf{S_{ff}} \end{bmatrix},$$

where the subscripts v, e and f refer to vertex, edge and face degrees of freedom, respectively. As described in section 3.3.1, the p-type (polynomial) scaling of the system is preconditioned through the numerical change of basis associated with the low energy basis (block diagonal)

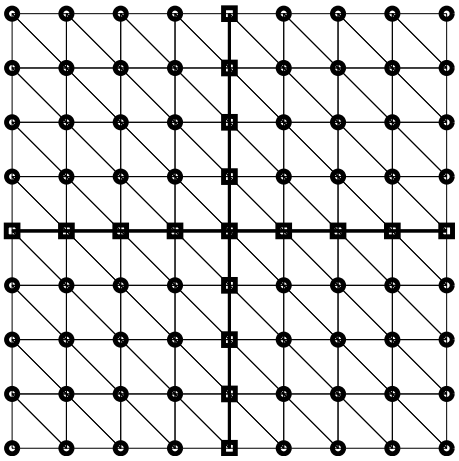


Figure 3.6: Partitioning of 2D domain consisting of triangular elements. The domain is subdivided into four partitions. Boundary-boundary vertices shared by partitions are marked by squares; interior-interior vertices are marked by circles.

preconditioner. To take account of the h-type (elemental) scaling in our preconditioning strategy an appropriate solution [59] is to invert the vertex space sub-matrix of the Schur complement, i.e., $\mathbf{S_{vv}}$. As we mentioned in section 3.3.1, we use an additive Schwartz preconditioner, which includes two parts: the coarse space linear vertex block *and* the Low Energy preconditioner constructed by the numerical orthogonalization of the basis [76], i.e.,

$$\begin{bmatrix} \mathbf{S_{vv}}^{-1} & & \\ & 0 & \\ & & 0 \end{bmatrix} + \mathbf{R^T} \begin{bmatrix} Diag[(\mathbf{S}_2)_{vv}] & & \\ & (\mathbf{S}_2)_{eb} & \\ & & (\mathbf{S}_2)_{fb} \end{bmatrix}^{-1} \mathbf{R},$$

where $Diag[(\mathbf{S}_2)_{vv}]$ is the diagonal of $\mathbf{S}_2$ vertex modes, $(\mathbf{S}_2)_{eb}$ ($(\mathbf{S}_2)_{fb}$) is the block diagonal of the edge (face) components [76]. Note that the Schur complement $\mathbf{S}$ has already been orthogonalized with respect to the bubble modes of the 3D expansion and in this sense might be considered as having low energy within the interior of an element. However, the vertices will not have low energy within the "wire-basket" space of the tessellated elements.

We can now further decompose the coarse space linear vertex system $\mathbf{S}_{vv}$ into degrees of freedom on the boundary of the partitions (denoted again with a subscript b) and those degrees of freedom within the partitions (denoted with a subscript i) to obtain

$$\mathbf{S_{vv}} = \begin{bmatrix} \mathbf{V_{bb}} & \mathbf{V_{bi}} \\ \mathbf{V_{ib}} & \mathbf{V_{ii}} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{v}}_\mathbf{b} \\ \widehat{\mathbf{v}}_\mathbf{i} \end{bmatrix} = \begin{bmatrix} \mathbf{g_b} \\ \mathbf{g_i} \end{bmatrix}, \tag{3.6}$$

where we have used $\widehat{\mathbf{v}}$ and $\mathbf{g}$ to denote the solution and the forcing vectors corresponding to the vertex components; the $\mathbf{V_{bb}}$ submatrix corresponds to the boundary-boundary modes; $\mathbf{V_{ii}}$ correspond to the interior-interior modes, and $\mathbf{V_{ib}} = \mathbf{V_{bi}^T}$ is the submatrix which couples these two systems. Analogously, $\widehat{\mathbf{v}}_\mathbf{b}$ and $\widehat{\mathbf{v}}_\mathbf{i}$ are the solution coefficients of the boundary and the interior degrees of freedom, marked in illustration of figure 3.6 by squares and circles respectively. Once again, we can apply substructuring to decouple the interior and boundary degrees of freedom, i.e.,

$$\begin{bmatrix} \mathbf{V_{bb}} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{V_{ib}} & 0 \\ \mathbf{V_{ib}} & \mathbf{V_{ii}} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{v}}_\mathbf{b} \\ \widehat{\mathbf{v}}_\mathbf{i} \end{bmatrix} = \begin{bmatrix} \mathbf{g_b} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{g_i} \\ \mathbf{g_i} \end{bmatrix}. \tag{3.7}$$

The matrix $\mathbf{V_{bb}} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{V_{ib}}$ is the Schur complement matrix of $\mathbf{S_{vv}}$ based on degrees of freedom along the partition. Further, by construction $\mathbf{V_{ii}}$ has a block-diagonal structure that can be readily inverted, where each block has a size corresponding to the number of interior vertices on a given partition. For a fixed mesh size, the number of boundary-boundary degrees of freedom depends on the number of partitions, and therefore the more partitions that are generated the larger the size of $\mathbf{V_{bb}}$ and the number of blocks in $\mathbf{V_{ii}}$. In contrast, the size of $\mathbf{V_{ii}}$ decreases as the number of partitions grows. Solution for $\widehat{\mathbf{v}}_\mathbf{i}$ can be computed locally within each partition once the values of $\widehat{\mathbf{v}}_\mathbf{b}$ have been determined.

The solution for $\widehat{\mathbf{v}}_\mathbf{b}$ can be computed locally by constructing the global system on each processor or using a parallel approach. For the relatively small amount of elements one typically uses in a spectral/*hp* element discretization as compared to a classical finite element methods such a local approach has been successfully applied for up to a few tens of processors. However, this approach will clearly saturate when considering larger number of processors and the number of elements typically required by large-scale computation. In the following sections, we discuss alternative approaches for solving the boundary-boundary vertex system, and then provide details on assembling the boundary-boundary system $\mathbf{V_{SC}} = \mathbf{V_{bb}} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{V_{ib}}$ in parallel.

3.4.2 Algorithms for solution of boundary-boundary system

We considered two approaches for solving the boundary-boundary system

$$[\mathbf{V_{SC}}]\widehat{\mathbf{v}}_\mathbf{b} = \mathbf{g_b} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{g_i}. \tag{3.8}$$

The first approach is based on the LU decomposition of the operator $\mathbf{V_{SC}}$, and the second is directly inverting the $\mathbf{V_{SC}}$ operator. Each of the two methods can be executed in serial or in parallel. In the serial approach the LU decomposition of operator $\mathbf{V_{SC}}$ (or its inverse) is replicated on each processor, while in the parallel one it is distributed over all processors.

As we mentioned before, when the number of partitions is small and consequently the rank of $\mathbf{V_{SC}}$ is low, it might not be beneficial to parallelize the solution of the boundary-boundary system. We implemented the serial and the parallel approaches for solving the system by inverting the $\mathbf{V_{SC}}$ operator (in the preprocessing only) and performing matrix-vector multiplication at every iteration; in figure 3.7 we compare the *overall* parallel efficiency of our solver. The rank of $\mathbf{V_{SC}}$ operator for the Poisson and Helmholtz solvers is presented in table 3.2. Clearly, the serial solver for system (3.8) is not scalable on more then 64 processors.

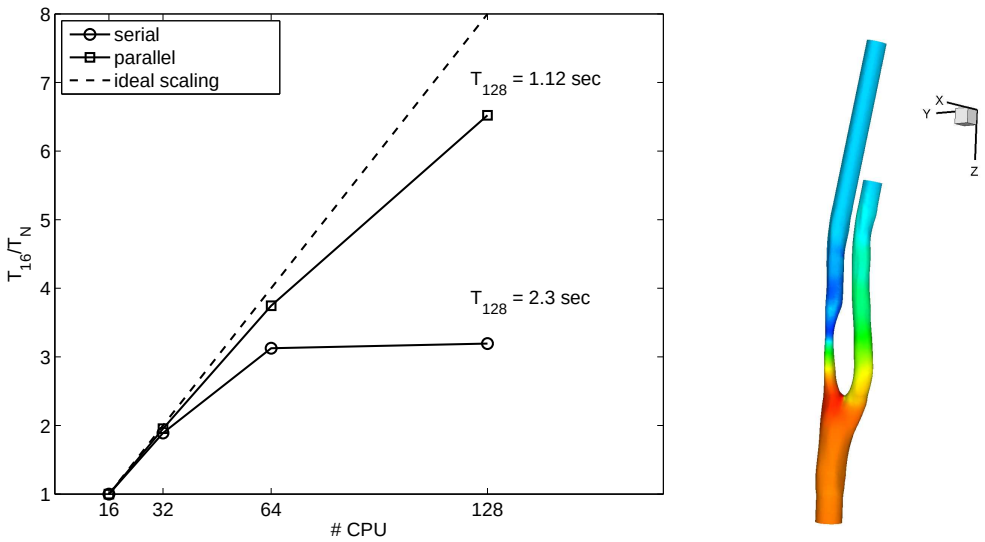


Figure 3.7: Blue Gene: Parallel efficiency of $\mathcal{Nektar}$ with serial and optimized parallel implementation of coarse space linear vertex solver. T_N - mean CPU-time required for one time step in computation with N CPUs. Simulation of unsteady flow in stenotic carotid artery (shown in the right plot). Problem size: 19270 tetrahedral elements, fifth-order spectral polynomial approximation. Rank of $\mathbf{V_{SC}}$ is presented in table 3.2. Simulation performed at SDSC.

Next we compare the performance of our solver where the system (3.8) is solved in parallel by means of the LU decomposition or with the inversion of the operator $\mathbf{V_{SC}}$; we used functions *pdgetrs* and *pdgemv* from ScaLapack library [4] for the parallel LU solve

# of CPUs	16	32	64	128
Poisson: Rank ($\mathbf{V_{SC}}$)	530	980	1619	2323
Helmholtz: Rank ($\mathbf{V_{SC}}$)	282	515	848	1226

Table 3.2: Rank of $\mathbf{V_{SC}}$ operator of the Poisson and Helmholtz solvers. Computational domain of stenotic carotid artery (see figure 3.7). Problem size: 19270 tetrahedral elements.

and for the matrix-vector multiplication, respectively. In both cases, the LU decomposition and inversion of $\mathbf{V_{SC}}$ was done during preprocessing and the time of the preprocessing was excluded; we plot the results in figure 3.8. The poor efficiency of the LU based solver is due to high volume of communication together with relatively high number of floating point operations. ScaLapack does not implement sparse algebra operators, and the LU decomposition is stored as a full dense matrix in 2D block cyclic format, hence the high number of floating point operations.

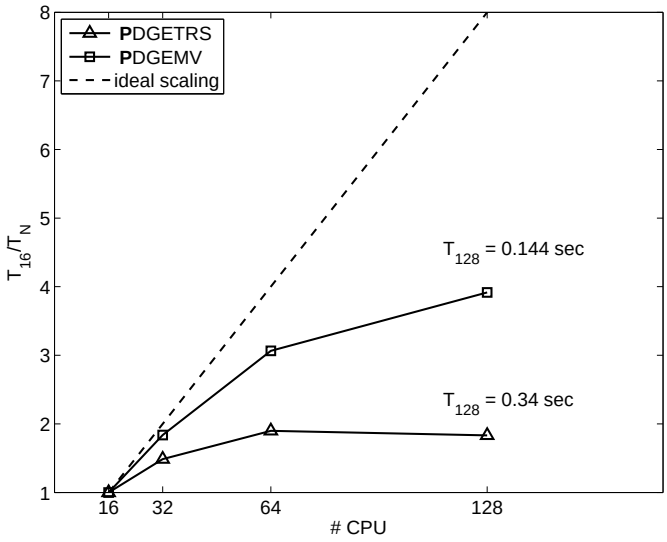


Figure 3.8: XT3: Parallel efficiency of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ with different implementations of coarse space linear vertex solver. T_N - mean CPU-time required for one time step in computation with N CPUs. Simulation of unsteady flow in stenotic carotid artery (shown in figure 3.7). Problem size: 19270 tetrahedral elements, third-order spectral polynomial approximation. Rank of $\mathbf{V_{SC}}$ is presented in table 3.2. Simulation performed at PSC.

In preliminary tests the parallel version of SuperLU library (version 2.0) [5] for solution of system (3.8) was implemented. SuperLU uses a sparse algebra and is efficient for solution of linear systems with very sparse operators. Although the number of non-zero values (nz) of $\mathbf{V_{SC}}$ is small (see figure 3.9), they are not packed in a way which will maximize the efficiency of memory access to the values of LU. Compared to ScaLapack-based solver, the

SuperLU-based solver, applied to (3.8), performed about five times slower on CRAY XT3 and showed poor scalability.

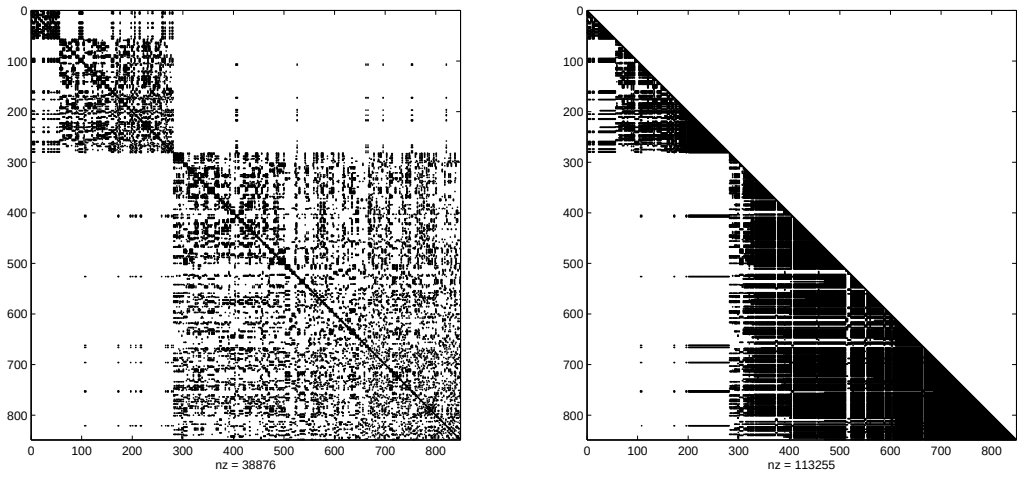


Figure 3.9: Velocity solver: Sparsity pattern of the $\mathbf{V_{SC}}$ operator (left) and its LU decomposition (right). Only the L part is plotted. Problem size: 19270 tetrahedral elements. Computational domain of carotid artery subdivided into 64 partitions.

Based on the performed tests we concluded that the most efficient way to solve the system (3.8) is to directly invert the operator and to perform a parallel matrix-vector multiplication. Our next tasks are to develop a technique to assemble the $\mathbf{V_{SC}}$ in parallel while minimizing memory requirements and find the most efficient procedure for the parallel matrix-vector multiplication. In the following section, we concentrate on the construction and inversion of $\mathbf{V_{SC}}$.

3.4.3 Construction of coarse space linear vertex operator

The parallel construction of the coarse space linear vertex operator is performed during the preprocessing stage in three steps:

1. Values of $\mathbf{V_{SC}}$ corresponding to each partition are computed locally.

2. Processors perform a ring-type communication to redistribute locally computed values of $\mathbf{V_{SC}}$.

3. LU decomposition and inversion of $\mathbf{V_{SC}}$ is performed in parallel with subsequent redistribution from 2D block cyclic format to standard block decomposition.

In the following we discuss these steps in detail.

The *first step* of $\mathbf{V_{SC}}$ construction consists of computing the local contribution, namely

$\mathbf{V}^{\mathbf{k}}_{\mathbf{SC}}(i,j)$, where i and j are global indices and k is the partition ID. At this point, our assumption is that the values $\mathbf{V}_{\mathbf{SC}}(i,j)$ may have random distribution across processors and any amount of overlap is allowed. Indeed, the distribution of $\mathbf{V}_{\mathbf{SC}}(i,j)$ depends on the global numbering of the boundary-boundary degrees of freedom and on the partitioning of the computational domain. In figure 3.10 we sketch distribution of values of $\mathbf{V}_{\mathbf{SC}}(i,j)$ computed for the velocity system, in the computational domain of carotid artery (see figure 3.7(right)) partitioned into four parts. To minimize the memory requirement, values of

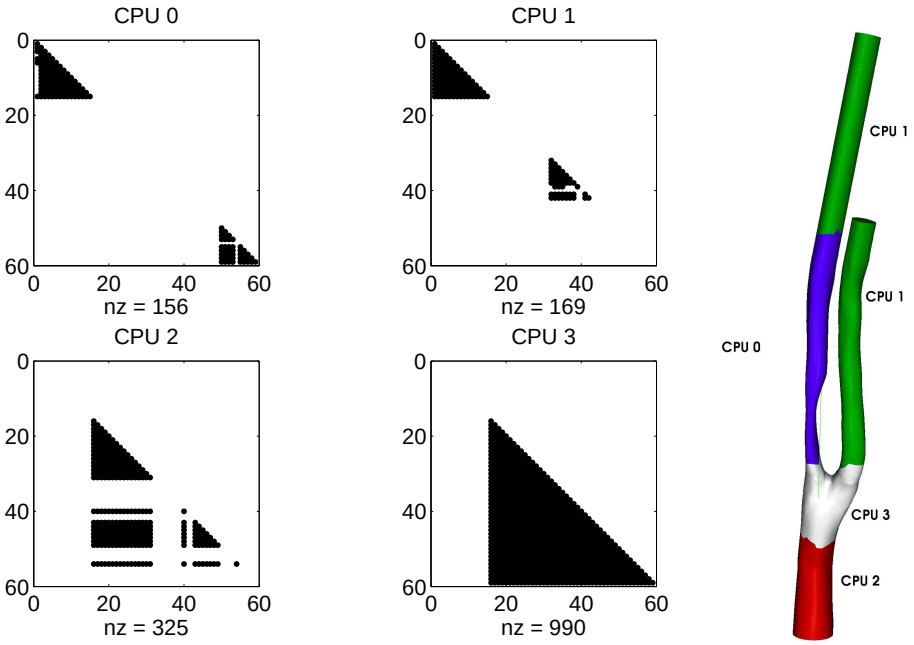


Figure 3.10: (in color) Initial distribution of $\mathbf{V}^{\mathbf{k}}_{\mathbf{SC}}(i,j)$ computed for the velocity system. Computational domain of carotid artery (see figure 3.7(right)) is sub-divided into four partitions. Due to the symmetry of the operator only low triangular terms are stored; nz - number of non-zero components.

$\mathbf{V}^{\mathbf{k}}_{\mathbf{SC}}(i,j)$ are stored in sparse matrix using the coordinate format [72]; moreover, the upper triangular part of symmetric operator $\mathbf{V}_{\mathbf{SC}}$ is not stored. The ordering of $\mathbf{V}^{\mathbf{k}}_{\mathbf{SC}}(i,j)$ stored in sparse matrix format is not important. Each vertex can be shared by several elements within partition. If a vertex contribution with the same i,j indices is already stored we add the value of $\mathbf{V}^{\mathbf{k}}_{\mathbf{SC}}(i,j)$ to the existing one, otherwise we increase the size of the matrix and append a new value with corresponding indices. It is not necessary to compute the exact number of values stored in the sparse matrix prior to its construction, since we can use dynamic memory reallocation to store new values. We note that the dynamic memory reallocation for the matrix is an expensive computation task; one way to minimize the

computational cost of resizing the matrix is to allocate some extra memory in advance. However, the discussed procedure is performed only in the preprocessing stage and typically requires less than one second.

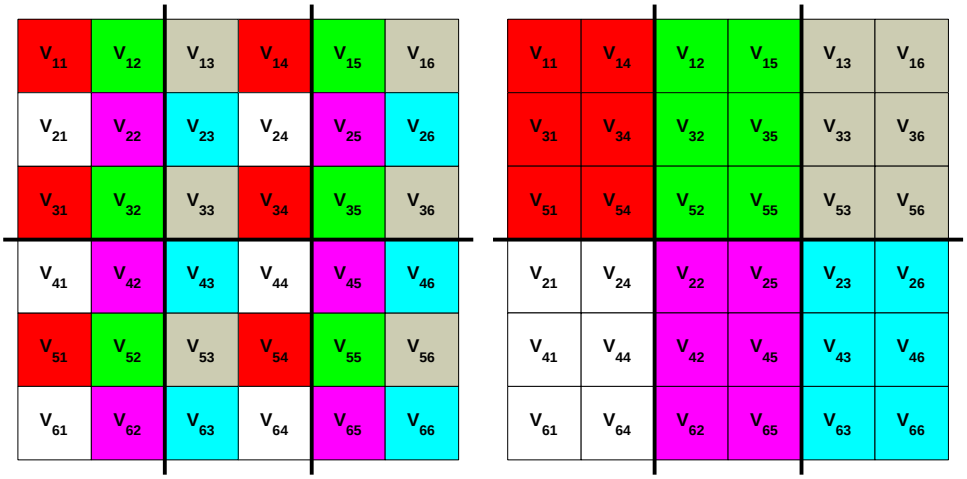


Figure 3.11: Parallel construction of $\mathbf{V_{SC}}$. Matrix is distributed onto 6 partitions on 3×2 two-dimensional processor grid. Wide lines represent decomposition of the matrix to six partitions. $\mathbf{V_{SCrc}}$ - block of rank N_{BS}, r and c are the row and the column index of each block. Left: standard decomposition; Right: two-dimensional block cyclic decomposition used in ScaLapack. Colors represent standard blocks (sub-matrices of a rank N_{BS}), which under 2D block cyclic mapping are grouped on the same processors.

The *second step* in the parallel construction of $\mathbf{V_{SC}}$ is to redistribute the locally computed values between partitions. In $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ we implement two matrix partitioning methods. In figure 3.11 we present decompositions of a matrix $\mathbf{V}$ onto 6 partitions. The left plot shows the standard block decomposition. The right plot depicts the corresponding two-dimensional block cyclic distribution. Each block $\mathbf{V_{rc}}$ is a square matrix of rank N_{BS}, $r = [1\ N_r]$ and $c = [1\ N_c]$ - are the block row and column index. If $mod(rank(\mathbf{V_{SC}}), N_{BS}) \neq 0$ then the blocks $\mathbf{V_{N_{rc}}}$ and $\mathbf{V_{rN_c}}$ are not square matrices; although, from algorithmic point of view this is perfectly acceptable, it alters the computational load balance. ScaLapack assumes matrices are laid out in a two-dimensional block cyclic decomposition. We use a basic block of size $N_{BS} = 4, ..., 8$. This size is not very efficient for LU decomposition and subsequent matrix inversion (performed only during preprocessing) where the optimal size of a block is between 30-80. However, the small size of the basic block leads to better load balancing in parallel matrix-vector multiplication, performed (by ScaLapack) at each iteration. As for the 2D processor grid layout, our

numerical experiments indicate that better performance is achieved when the number of processors-columns is greater or equal to the number of processors-rows.

Given the alignment of processors where $\mathbf{V_{SC}}$ is to be distributed, the size of the base block and considering the rank of $\mathbf{V_{SC}}$, we allocate only the necessary memory for sub-matrix of $\mathbf{V_{SC}}$ on each processor. To map $\mathbf{V_{SC}}$ into two-dimensional blocks with cyclic distribution required by ScaLapack, sparse matrices with values and indices of $\mathbf{V^k_{SC}}(i,j)$ are circulated between processors (employing ring-type communication) where the local part of two dimensional block cyclic matrix $\mathbf{V_{SC2D}}$ is updated with relevant values from $\mathbf{V^k_{SC}}(i,j)$.

At the *third step* we use ScaLapack for parallel LU decomposition and then parallel matrix inversion. As we already mentioned, the size of a basic block in 2D cyclic distributed matrix is not optimized for parallel LU decomposition and matrix inversion, however these two operations require only 0.5-5 seconds. After the parallel matrix inversion is performed, we can use the $(\mathbf{V_{SC2D}})^{-1}$ distributed over all processors in 2D block cyclic format for parallel matrix-vector multiplication to compute $\widehat{\mathbf{v}}_\mathbf{b} = \mathbf{\Pi^{-1}}\left[(\mathbf{V_{SC2D}})^{-1}\mathbf{\Pi}(\mathbf{g_b} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{g_i})\right]$ using the ScaLapack *pdgemv* function. The operator $\mathbf{\Pi}$ is permutation operator which maps a vector into block cyclic format. Alternatively, we can map $(\mathbf{V_{SC2D}})^{-1}$ to form $(\mathbf{V_{SC}})^{-1}$, which is distributed in standard blocks over all processors, as illustrated in figure 3.11. We found that the latter approach is advantageous due to additional optimization we can apply for redistributing vectors $\widehat{\mathbf{v}}_\mathbf{b}$ and $(\mathbf{g_b} - \mathbf{V_{bi}}[\mathbf{V_{ii}}]^{-1}\mathbf{g_i})$ over processors. We provide details of gathering and scattering of these vectors in section 3.4.5.

3.4.4 Load balancing in parallel matrix vector multiplication

The efficiency of parallel matrix-vector multiplication depends strongly on two factors: (a) effectiveness of communication between processors, and (b) serial performance of matrix vector multiplication. In the original ScaLapack *pdgemv* function and modified for $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ *pdgemv* function the serial part of matrix-vector product is based on the same BLAS library. However, the communication patterns are different. In figure 3.12 we compare communication balance during the matrix vector multiplication performed by ScaLapack *pdgemv* function and by the $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ *pdgemv* function. In this simulation 128 processors laid out in 8×16 process grid were used. The first 16 processors aligned on the row zero, the next 16 (processor IDs 16 to 31) on the first row, and so on. Processors ID on the zero column of the 2D process grid can be computed from $ID_{(row,col=0)} = row*16,\ row = 0,1,...,7$.

As shown in figure 3.12 (upper plot), a severe load balance problem occurs when MPI_Bcast is called. Additionally, the processors from column zero spend more time in MPI_Allreduce than those from other columns. This situation is due to two factors: (a) ScaLapack *pdgemv* function expects that the vector, which multiplies the distributed matrix, is supplied to the column zero of process grid only in cyclic block-*row* distribution, then it is redistributed among other processors in cyclic block-*column* distribution; and (b) ScaLapack *pdgemv* function returns the result in cyclic row format to the processors of the column zero only. Thus, the result should be reordered and then broadcast from the column zero to the rest. While the result of matrix-vector multiplication is reordered on the column zero (this operation also includes one communication between processors of column zero), processors on other columns are waiting.

By modifying the ScaLapack *pdgemv* function we could eliminate unnecessary data transfer from one processor to another and optimize redistribution of multiplication result to all processors. As a result we, achieved better load balancing and also reduced the total execution time by 10 to 30 percent as presented in Table 3.3.

ScaLapack *pdgemv*	$\mathcal{NEKTAR}$ *pdgemv*
0.55 sec	0.37 sec

Table 3.3: Blue Gene: Mean CPU-time per time step for simulation with Low Energy preconditioner using ScaLapack *pdgemv* and optimized $\mathcal{NEKTAR}$ *pdgemv* function. Problem size: 19270 tetrahedral elements with fourth-order polynomial approximation. Ranks of $\mathbf{V_{SC}}$ for Poisson and Helmholtz solvers are 2981 and 1570. Computation performed on 256 computer nodes of Blue Gene with 2 processors per node.

3.4.5 Implementation of parallel matrix-vector multiplication in $\mathcal{NEKTAR}$

In figure 3.13 we sketch the distributed matrix $(\mathbf{V_{SC}})^{-1}$. For illustration purposes, the matrix is distributed over a 6-processor mesh, aligned in two rows and three columns. In the following for simplicity of notations we denote by $\mathbf{x}$ and $\mathbf{y}$ the solution vector and forcing term. We consider the parallel matrix-vector multiplication as a three-step procedure:

step 1 - parallel assembling of the vector $\mathbf{y}$;

step 2 - local sub-matrix - sub-vector multiplication;

step 3 - redistributing the results of the last operation between partitions.

On *step 1*, local contribution to vector $\mathbf{y}$ is computed on each processor and then redis-

tributed. We note that we use the spectral element code where domain decomposition is done element-wise by Metis [3], while the dense matrix $(\mathbf{V_{SC}})^{-1}$ is distributed regardless of the element-wise domain decomposition. The latter requires to develop an efficient and balanced procedure for parallel construction of the vector $\mathbf{y}$ and later for distribution of the result of matrix-vector product stored in $\mathbf{x}$. For communication between processors we split the default MPI communicator global MPI_COMM_WORLD into MPI_COMM_ROW and MPI_COMM_COLUMN sub-communicators in order to perform message passing within the scope of each row or column of the process grid.

Redistribution of the vector $\mathbf{y}$ is done in three stages:

a) On each processor we compute the local contribution to the vector $\mathbf{y}$.

b) On every row of the processor mesh we distribute values of $\mathbf{y}$ such that on processors of each column we compute $\sum_{k=0}^{N_{col}-1} y_{j,k}$, where N_{col} is number of columns in 2D grid of processors, j is the global column index in the range $J_s <= j <= J_e$. Here J_s and J_e are the global index of the first and the last column of the local block of $(\mathbf{V_{SC}})^{-1}$.

c) Finally, values of $y_j,\ J_s <= j <= J_e$ are summed on each column of the processor mesh using MPI_Allreduce and MPI_COMM_COLUMN communicator.

In figure 3.14 we provide an illustration of the elements redistribution within each row of processors. In the illustration the vector $\mathbf{y}$ has a length of 10; on the processors of column zero values of $y_j,\ j = 0, ..., 3$ are required, on the processors of column one values $y_j,\ j = 4, ..., 7$ are required, and on the processors of the last column values of $y_j,\ j = 8, 9$ are required for matrix-vector multiplication. The shaded fields correspond to values of y_j computed in each partition according to element-wise domain decomposition.

We considered several MPI implementations to perform step 1. In the following we provide the description of four different methods we implemented. For clarity we provide a pseudocode for these four methods in Appendix B.

Our first version (V1) uses MPI_Allreduce within MPI_COMM_WORLD for a global reduction operation over the entire length of the vector $\mathbf{y}$. This operation transmits more data than is strictly necessary, since each processor only needs to get data for y_j with j bounded by $J_s <= j <= J_e$. For the example of figure 3.14 the length of message will be 10. In addition, the number of arithmetic operations in this case is higher, since *all* values of vector $\mathbf{y}$ are passed and summed on each processor.

In the second version (V2) we employ the aforementioned sub-communicators

MPI_COMM_ROW and MPI_COMM_COLUMN. First, MPI_Allreduce operation is done within each row to perform MPI_SUM of the vector **y** over its entire length; then MPI_Allreduce is done within each column. This version communicates significantly less data than V1, yet there are still some redundant data (zeros) being transmitted in a scope of a row.

In the third version (V3) we employ MPI_Alltoallv instead of MPI_Allreduce in rows, while the second MPI_Allreduce is the same as in V2. The MPI_Alltoallv is a blocking function, however, performed over the row communicator MPI_COMM_ROW
it synchronizes processors within a scope of each row only. This version does not communicate unnecessary data in a scope of a row.

In the fourth version (V4) we replace the blocking call MPI_Alltoallv with point-to-point, non-blocking sends and receives (MPI_Isend and MPI_Irecv) operations, followed by MPI_Waitany; while the MPI_Allreduce operation within columns is the same as in V2.

These four versions were studied in detail on three architectures: IBM Blue Gene/L, Cray XT3 and IBM Power 4. We found that on sufficiently large partitions of *Blue Gene* (over 2048 CPUs) versions V1 and V3 perform with almost the same speed and are better than the other two versions. The explanation of this behavior has several components:
1. Global MPI_Allreduce is done over the tree interconnect of Blue Gene, which is specialized for collective communication, while collective calls over subcommunicators are done using the 3D torus interconnect.
2. There is only one MPI call with associated latency instead of two calls as in V2 and several calls in the V4.
3. The reason that V4 version does not perform as well as V3 most likely has to do with the fact that vendor-implemented MPI_Alltoallv optimizes network traffic to avoid hotspots.
On *IBM Power4* (Datastar) the cost of global MPI_Allreduce was more than twice higher than row-wise and column-wise MPI_Allreduce. For example, using 512 processors of IBM Power 4 construction of the vector **y** with a global MPI_Allreduce (V1) call was accomplished in 0.093 seconds, while using the alternative approach (V2) 0.040 seconds where required. This behavior is more intuitively understandable since in the latter approach less data are sent. On *CRAY XT3* we likewise saw a behavior different than the one on Blue Gene: V4 performed faster than V1, V2 and V3. This demonstrates the different characteristics of interconnects of different platforms and the necessity to target algorithm development correspondingly. In the benchmark study presented in the following section we focus on the

best algorithm on a given platform.

The *step 2* of parallel matrix-vector multiplication requires only one call to level 2 BLAS function only and no communication is performed. On each processor we multiply the local part of $(\mathbf{V_{SC}})^{-1}$ by the local part of $\mathbf{y}$ to obtain local part of $\mathbf{x}$.

On the *step 3*, to redistribute the result of the parallel matrix vector product we implement the following procedure: First, we sum the result in each row of the processors concurrently using MPI_Allreduce and MPI_COMM_ROW communicator. The size of the message here is equal to the number of $(\mathbf{V_{SC}})^{-1}$ rows stored on each row of the process grid. Second, within the scope of each column we implement non-blocking communication to exchange values of $\mathbf{x}$ between processors, the algorithm is similar to V4 used for gathering values of the vector $\mathbf{y}$. We note that the processor in row i receives only the required values of $\mathbf{x}$ from other processors in its column, according to the original element-wise domain decomposition.

3.4.6 Parallel Performance Results

In this section we investigate the performance of Low Energy Basis and Diagonal Preconditioner used in $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$. We compare the *scalability* and *effectiveness* of the two preconditioners. The Schur complement of the Mass and the Stiffness operators constructed with the polynomial basis used in $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ are diagonally dominant, and the values on their diagonals are varying, and this makes the Diagonal Preconditioning quite effective, unlike in other methods. In table 3.4 we compare the number of iterations required for the solution of Helmholtz equation $\nabla^2 U - \lambda U = -(\lambda + 6\pi^2)\sin(\pi x)\cos(\pi y)\cos(2\pi z)$ with Conjugate Gradient solver and three types of preconditioners:

a) the Diagonal Preconditioner, defined as the inverse of the main diagonal of the Schur complement S.

b) the Block Preconditioner, defined as

$$\begin{bmatrix} \mathbf{S_{vv}}^{-1} & & \\ & \mathbf{S_{ee}}^{-1} & \\ & & \mathbf{S_{ff}}^{-1} \end{bmatrix},$$

where $\mathbf{S_{ee}}^{-1}$ and $\mathbf{S_{ff}}^{-1}$ are constructed by inverting the blocks of S_{ee} (S_{ff}) corresponding to coupling of the edge (face) i with itself only, i.e., omitting the coupling of the edge (face) i with the edge (face) j.

c) the Coarse Space Linear Vertex Block Preconditioner, defined as

$$\begin{bmatrix} \mathbf{S_{vv}}^{-1} & & \\ & I & \\ & & I \end{bmatrix}.$$

For the discretization we use 96 tetrahedral elements, and the iterations are terminated when the residual is less then $10E-6$. We observe that significant reduction in the iteration count is achieved with the simple Diagonal Preconditioner.

P	No Prec.	Diag. Prec.	Block Prec.	Vertex Block Prec.
2	3	3	1	1
4	162	36	36	125
6	400	71	67	312

Table 3.4: Iteration count: solution of Helmholtz equation with Conjugate Solver and different preconditioners. Problem size: 96 tetrahedral elements.

Results presented in this section show that the computational cost of simulations of unsteady flows is considerably lower with LEBP than with the standard Diagonal Preconditioner. Our results also show that scalability of the more effective LEBP is practically the same as of Diagonal Preconditioner. In all our tests we terminate the conjugate gradient iterations when the relative error in the residual is less then 1E-8.

Simulations of unsteady flow typically require integration over $10^5 - 10^6$ time steps and at each time step 20 to 100 iterations are executed on average. Thus, the cost of solution of preconditioning outweighs significantly the cost of construction of the LEBP, hence it is not included in the performance study.

3.4.6.1 Performance of Low Energy Basis and Diagonal Preconditioner

First we compare the iteration count using the Diagonal Preconditioner and LEBP. Our computational domain has a shape of cylindrical pipe and is constructed of 67456 tetrahedral elements; the initial condition for velocity fields is $\mathbf{v} = 0$. At the inlet of the domain we prescribe a Poiseuille velocity profile, while at the outlet fully developed flow is assumed.

We monitor the number of iterations during the first 1000 time steps; we observe (see figure 3.15) that the number of iterations at the beginning of simulation is higher than after some transient period, thus the performance of numerical solver with the two preconditioners is compared after a few hundreds of time steps.

In figure 3.16 and Table 3.5 we compare the mean CPU-time per time step required for 3D flow simulation with LEBP and with Diagonal Preconditioner; almost an 8-fold speed-up in the execution time is observed. In this simulation the solution was integrated 1000 time-steps and the mean CPU-time per time step was measured as an average of the CPU-time required for the last 10 time steps of simulation. In the last column of Table 3.5 we present the Rank of $\mathbf{V_{SC}}$. Note that with increasing number of processors the size of the local block of $\mathbf{V_{SC}}$ is decreased, which means that the computation versus communication ratio is decreasing as well.

# of CPUs	Diag. Prec.	Low Energy Prec.	Rank of $\mathbf{V_{SC}}$: Velocity (Pressure)
512	24.1 sec	2.88 sec	5449 (8596)
1024	12.43 sec	1.61 sec	6625 (10577)
2048	6.69 sec	0.98 sec	7552 (12192)
3072	N/A	0.76 sec	8018 (13072)

Table 3.5: Blue Gene: Mean CPU-time for simulation with Diagonal and Low Energy Basis Preconditioner. Problem size: 67456 Tetrahedral elements with eight-order polynomial approximation. Computation performed using 1 processor per node.

Performance of $\mathcal{NEKTAR}$ with LEBP was also tested on CRAY XT3 computer of ERDC, which has dual-core AMD Opterons 2.6GHz processors and 2GB memory per core (but only 1.8GB is available for applications). Achieving good scalability on a cluster of processors with relatively high clock-speed requires very good optimization of communication algorithms since the time spent on computation is usually 2 to 4 times less than on computers with relatively slow processors such as Blue Gene. In figure 3.17 and Table 3.6 we show the scalability of $\mathcal{NEKTAR}$ achieved on the CRAY XT3 computer. The data presented in figure 3.17 are based on a simulation in a domain of 120,813 tetrahedral elements with 6th, 8th and 10th polynomial approximation. In each simulation the solution was integrated for 500 time steps and the mean CPU-time was computed based on the last 10 time steps.

In figure 3.18 we plot the CPU-time balance monitored during the first 40 iterations required by the Poisson solver in simulation of unsteady flow in a domain carotid artery

Number of CPUs	P=6	P=8	P=10
512	0.64 sec	1.3 sec	2.5 sec
1024	0.39 sec	0.74 sec	1.37 sec
2048	0.28 sec	0.46 sec	0.78 sec
4096*			0.50 sec
N_{points}	69,588,288	132,894,300	226,161,936

Table 3.6: CRAY XT3 (ERDC): Mean CPU-time for simulation with LEBP. Problem size: 120,813 Tetrahedral elements with 6th, 8th and 10th order polynomial approximation. N_{points} is a total number of quadrature points for each unknown. The result for 4096 CPUs was obtained on CRAY XT3 of the Pittsburgh Super Computing center; in our experience this computer is typically 5-10% slower than CRAY XT3 of ERDC.

(figure 3.7). For this simulation we used 128 processors of CRAY XT3. The plots show that about half of the CPU-time is consumed by preconditioning. Only 15% of the CPU-time consumed by preconditioning is spent on solving system (3.8); this time includes the three steps of the parallel matrix-vector multiplication as discussed in section 3.4.5. The diagonal preconditioning requires only one vector-vector multiplication and no communication; thus the time spent on the preconditioning is negligible, and the total CPU-time required for one iteration in the last simulation would be of order $4E-3$ to $5E-3$ sec.

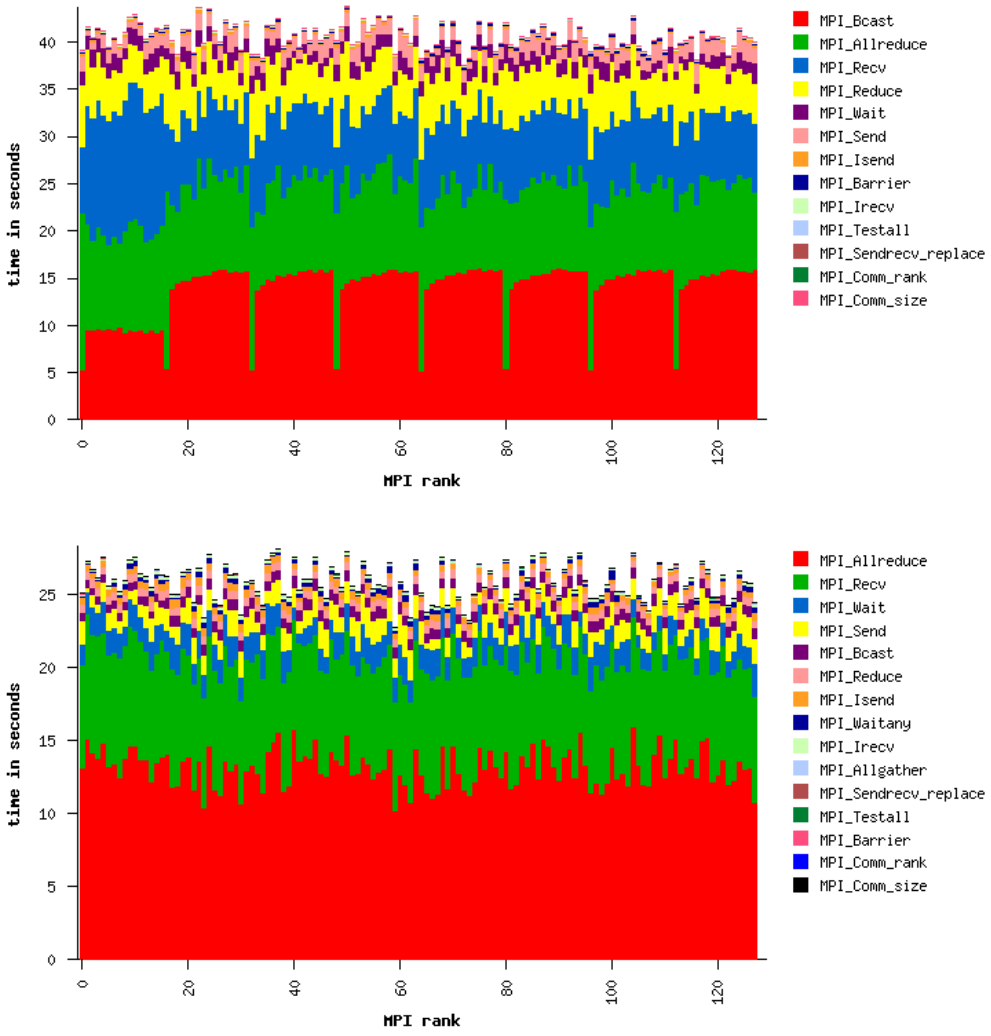


Figure 3.12: (in color) Blue Gene: Load imbalance in preconditioning stage. Upper plot - simulation with ScaLapack *pdgemv* function. Lower plot - simulation with $\mathcal{NEKTAR}$ pdgemv function. Problem size: 19270 tetrahedral elements, fifth-order polynomial approximation. Simulation performed on IBM Blue Gene supercomputer using 128 CPUs. Performance was monitoring with IPM tool [1].

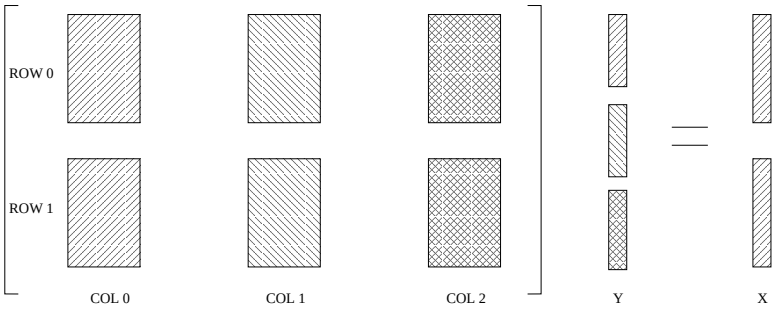


Figure 3.13: Parallel matrix-vector multiplication $(\mathbf{V_{SC}})^{-1}\mathbf{y} = \mathbf{x}$. The operator $\mathbf{V_{SC}}$ is distributed on 3×2 processors grid. The vector $\mathbf{y}$ is partitioned into three sub-vectors according to number of columns, and vector $\mathbf{x}$ is partitioned into two sub-vectors according to number of row of the processor grid

.

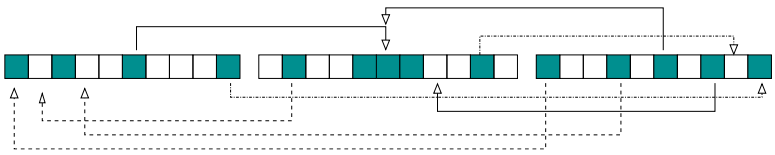


Figure 3.14: Gathering of values of the vector $\mathbf{y}$ within rows in processor mesh.

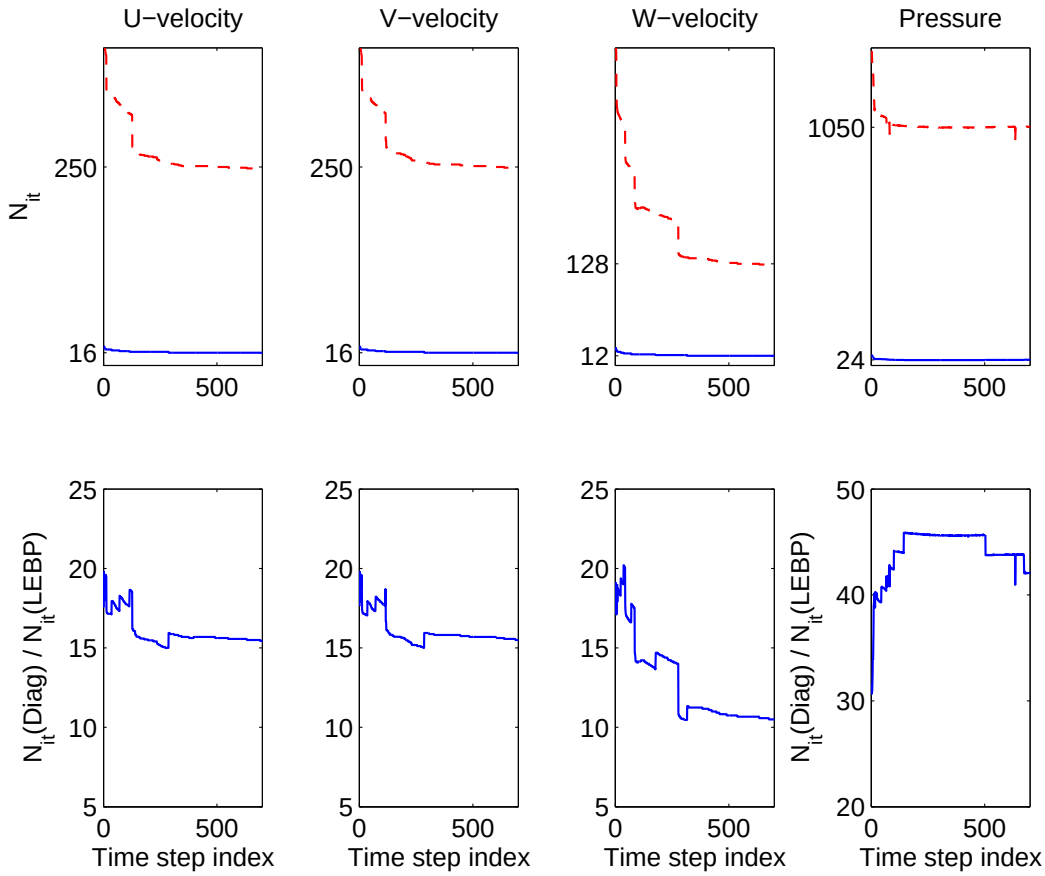


Figure 3.15: Number of PCG iterations for three velocity components and pressure. Upper plots: number of iterations: solid line - LEBP, dash line - Diagonal Preconditioner. Lower plots - reduction in iteration count. Problem size: 67456 Tetrahedral elements with eight-order polynomial approximation.

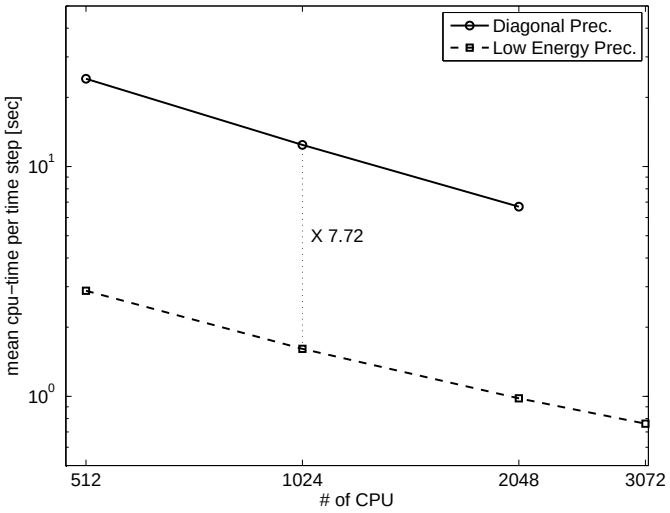


Figure 3.16: Blue Gene (SDSC): Mean CPU-time for simulation with Diagonal and Low Energy Basis Preconditioner. Problem size: 67456 Tetrahedral elements with eight-order polynomial approximation. Computation performed using 1 processors per node.

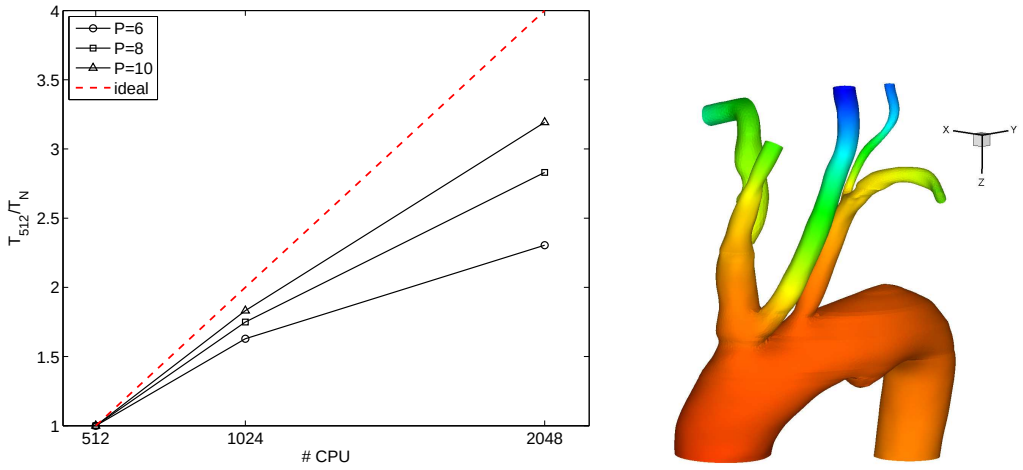


Figure 3.17: CRAY XT3 (ERDC): Left: Performance of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ with LEBP. Parallel speed-up. Problem size: 120,813 Tetrahedral elements with 6th, 8th and 10th order polynomial approximation. Right: geometry of the computational domain, color corresponds to pressure values.

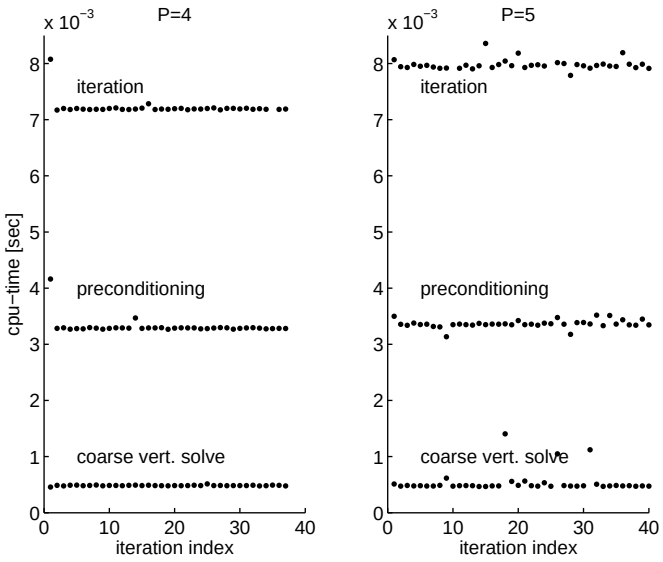


Figure 3.18: CRAY XT3 (PSC): Performance of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ with LEBP. CPU-time balance. Problem size: 19,270 Tetrahedral elements with 4th (left) and 5th (right) order polynomial approximation. Computational domain of carotid artery, illustrated in figure 3.7. Computation performed on 128 CPUs.

3.5 Acceleration of Iterative Solution of Dynamical Systems

Effective parallel preconditioner is only one possibility to accelerate iterative solver. In addition to efficient preconditioning, the number of conjugate iterations can be significantly reduced by providing a good initial state. In this section we explore two approaches to accelerate iterative solver by improving the initial state.

Discretization of first order (in time) PDEs

$$\frac{\partial \mathbf{u}}{\partial t} = f(t, \mathbf{u}, \mathbf{x}) \Rightarrow \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = f(t, \mathbf{u}, \mathbf{x})$$

often requires iterative solution of a system of equations $\mathbf{A}\mathbf{u}^{n+1} = \mathbf{b}$, and it is common practice to use the solution from a previous time step $\mathbf{u}^n$ as an initial state provided for iterative solver. However, *approximate* solution for $\mathbf{u}^{n+1}$ can be obtained using various techniques. Depending on the method and the smoothness of $\mathbf{u}(t)$, the distance between approximate solution (here and thereafter denoted by $||\mathbf{u}_{ap}^{n+1} - \mathbf{u}^{n+1}||$) can be significantly smaller than $||\mathbf{u}^n - \mathbf{u}^{n+1}||$ leading to reduction in number of iterations. Reduction in the iteration count obviously leads to a faster solution and enhances the overall parallel efficiency of a solver, hence it is crucial that computing $\mathbf{u}_{ap}$ will require a *minimal computational effort*.

We focus on two methods to compute $\mathbf{u}_{ap}^{n+1}$. The *first method* is based on a simple extrapolation (in time)

$$\mathbf{u}_{ap}^{n+1}(\mathbf{x}) = \sum_{k=0}^{N-1} \beta_k \mathbf{u}^{n-k}(\mathbf{x}). \tag{3.9}$$

The coefficients β_k are computed using Lagrange extrapolation technique. The method is general and can be applied in conjunction with the finite difference, finite element and spectral spatial discretization. In spectral methods the solution is approximated by expansion

$$\mathbf{u}(t, \mathbf{x}) = \sum_{m=1}^{M} \hat{\mathbf{u}}_m(t) \phi_m(\mathbf{x}). \tag{3.10}$$

In the family of *nodal* spectral methods the values of a function $\mathbf{u}(t, \mathbf{x}_j)$ at quadrature points $\mathbf{x}_j$ are the same as the values of coefficients $\hat{\mathbf{u}}_j(t)$ since $\phi_m(\mathbf{x}_j) = \delta_{m,j}$. In the *modal* spectral approximation the values $\hat{\mathbf{u}}_m(t)$ are grid-independent, hence two choices for time extrapolation are available: the solution can be extrapolated in physical space or in modal space. The two choices are not always equivalent, which will be discussed in section

3.5.1. We will denote the operator to calculate the approximate solution using the simple extrapolation by $\mathbf{u}_{ap} = EXT(\mathbf{u})$.

The *second method* is based on proper orthogonal decomposition (POD), where the field $\mathbf{u}(t, \mathbf{x})$ is represented by orthogonal temporal ($a_q(t)$) and spatial ($\Psi_q(\mathbf{x})$) modes:

$$\mathbf{u}(t, \mathbf{x}) = \sum_{q=1}^{Q} a_q(t)\Psi_q(\mathbf{x})$$

The approximate solution $\mathbf{u}_{ap}$ is obtained from

$$\mathbf{u}_{ap}(t + \Delta t, \mathbf{x}) = \sum_{q=1}^{Q} a_q(t + \Delta t)\Psi_q(\mathbf{x}),$$

where values $a_q(t + \Delta t)$ are extrapolated from $a_q(t - m\Delta t), \quad m = 0, ..., Q - 1$. We will denote the operator to calculate the approximate solution using POD by $\mathbf{u}_{ap} = POD(\mathbf{u})$.

The two acceleration techniques are applied in solving Navier-Stokes equations with $\boldsymbol{\mathcal{NEKTAR}}$. Specifically, we apply the extrapolation techniques to predict better the initial state for Helmholtz equation for the velocity. Velocity is a dynamic variable and it is a smooth function in time, hence a good prediction of $\mathbf{u}^{n+1}$ can be obtained via extrapolation methods. Pressure is not a dynamic variable but a constrain required to ensure the incompressibility of the velocity field, for that reason it is difficult to obtain a good initial state for the pressure solver.

3.5.1 Prediction of Numerical Solution Using Polynomial Extrapolation

Approximation of initial guess $\mathbf{u}_{ap}^{n+1}$ is computed using formula (3.9), and can be performed in physical space or in the modal space (3.10). The numerical algorithm implemented in $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ requires transforming velocity values from the modal to physical space in order to compute the non-linear term. Hence, choosing to perform extrapolation of solution in physical space will not require additional computationally expensive operation. However, it is still advantageous to perform the extrapolation using modal values $\hat{\mathbf{u}}_m(t)$ due to several reasons:

- Transformation from the physical to modal space is computationally expensive. Extrapolation performed in the physical space will require solution of a projection problem $\mathbf{M}\hat{\mathbf{u}}_{ap}^{n+1} = (\mathbf{u}_{ap}^{n+1}, \mathbf{\Phi})$, which demands considerable computational effort. To reduce the computational cost it is possible to solve a local projection problem (elementwise) directly using precomputed inverse of the mass matrix $(\mathbf{M^e})^{-1}$. The local projection destroys the C^0 continuity, hence additional computational effort might be needed if C^0 continuity of $\hat{\mathbf{u}}_{ap}^{n+1}$ is required. It is possible to maintain the C^0 requirement even if $\hat{\mathbf{u}}_{ap}^{n+1}$ is computed by a local projection, such procedure is equivalent to solution of a projection problem defined on Ω_e with unique Dirichlet boundary conditions computed on the interfaces of elements.

- The number of modal degrees of freedom is typically lower than the number of quadrature points required for projection operations, consequently extrapolation in the modal space requires less floating point operations and storage.

- Solution of a linear system arising in spectral element discretization, is typically performed using Schur decomposition technique which decouples the boundary and interior modes. The interior modes are typically computed using direct solver, whereas, the boundary modes are computed iteratively, hence extrapolation of the interior modes is not required, which leads to considerable computational savings.

In this study $\mathbf{u}_{ap} = EXT(\mathbf{u})$ is computed using modal coefficients and the C^0 continuity is enforced by overwriting the values of boundary modes of one element with those from the adjacent element. We note that such approach leads to slightly different values of $\hat{\mathbf{u}}_{ap}^{n+1}$ than would be obtained from solution of a global projection problem, however, it is the least

expensive technique from computational standpoint and it preserves the spectral accuracy. We also remind, that we employ the extrapolation not to compute the velocity field $\mathbf{u}^{n+1}$, but to obtain an initial guess for the iterative solver. Hence the accuracy and stability of a numerical scheme are not affected by methods for $\mathbf{u}_{ap}^{n+1}$ approximation.

Computational complexity: The aforementioned extrapolation technique requires storing solution from the previous N time steps. The required storage per field is then $N \times Nqp \times sizeof(double)$ bytes, where Nqp is the total number of quadrature points if the extrapolation is performed in physical space; alternatively Nqp is the number of *global* boundary modes. Additional storage for $\mathbf{u}_{ap}^{n+1}$ is not required since the data can be written instead of solution field from the time step t^{n-N+1}. At each time step one *dscal* and $(N-1)$ *daxpy*[1] calls to BLAS library are required. Transformation of a solution into modal space requires N_{el} direct solves of a linear system $\mathbf{M}^e\hat{\mathbf{u}}_{ap}^{n+1} = (\mathbf{u}_{ap}^{n+1}, \phi)$ which can be performed using precomputed LU factorization or inverse of a symmetric operator $\mathbf{M}$, note that only solution for boundary modes is required. If $\mathbf{u}_{ap}$ is a vector field then one system with multiple right hand sides should be solved, which is more efficient then solving a series of a single right hand side systems.

[1] *dscal* function computes $y[i] = ax[i]$; *daxpy* function computes $y[i] = ax[i] + y[i]$.

3.5.2 Prediction of numerical solution using POD

POD is a very effective method for identifying an energetically dominant set of eigenmodes in an evolving system. For a set of data $\mathbf{u}(t, \mathbf{x})$, represented as a function of physical space $\mathbf{x}$ and time t, POD determines a set of orthogonal basis functions of space $\Psi_i^q(\mathbf{x})$ and temporal modes $a_q(t)$; here $i = 1,\ 2,\ 3$ is the coordinate index and $q = 1, 2, ..., Q$ is the mode index.

We employ the method of snapshots to compute the POD modes in a time interval $T = (Q-1)\Delta t$. The inner product between every pair of velocity fields (snapshots) $C(t,t') = \int_\Omega \mathbf{u}(\mathbf{x},t)\mathbf{u}(\mathbf{x},t')d\mathbf{x}$ is the temporal auto-correlation covariance matrix $C(t,t')$ used as the kernel. The temporal modes $a_q(t)$ are the eigenvectors of the $C(t,t')$ matrix and are calculated by solving an eigenvalue problem of the form: $\int_T C(t,t')a_q(t')dt' = \lambda_q a_q(t)$. Using orthogonality, the POD spatial modes $\Psi_i^q(\mathbf{x})$ are calculated by $\Psi_i^q(\mathbf{x}) = (\lambda_q)^{-1}\int a_q(t)u_i(t,\mathbf{x})dt$. The eigenvalue of a single mode represents its contribution to the total kinetic energy of the flow field $\int_\Omega \langle u_i(\mathbf{x})u_i(\mathbf{x})\rangle d\mathbf{x}$ which is equal to the sum over all eigenvalues. Therefore, the eigenspectrum of the decomposition can be regarded as the primary information indicating the importance of each individual mode from the energetic point of view. The modes with the lowest numbers are the most energetic modes and correspond to coherent flow structures.

In order to approximate the solution at time step t^{n+1} the following procedure is employed:

1. The solution fields from time steps $t^{n-k}, \quad k = 0, 1, ..., Q-1$ are stored in physical space and at every time step t^n the correlation matrix $\mathbf{C}$ is constructed:

$$C_{i,j} = (\mathbf{u}(t^i,\mathbf{x}), \mathbf{u}(t^j,\mathbf{x})), \quad i,j = 1, ..., Q.$$

   There is no need to recompute $\mathbf{C}$ for *all* indices $i, j = 1, ..., Q$ at every time step, but only the inner product $(\mathbf{u}(t^n,\mathbf{x}), \mathbf{u}(t^j,\mathbf{x})), \quad j = 1, ..., Q$. The inner product of fields computed at times $t^{n-k}, k = 0, ..., Q-1$ does not change. The correlation matrix is computed in parallel and one global summation of a vector of a size Q is required. We also considered a variation of the POD of a vector field, where the velocity components are treated as independent (uncorrelated) scalar fields. The computational complexity of such method is only slightly higher: a) global summation of a vector of size $3Q$ is required, and b) the eigenvalues and eigenvectors of three correlation matrices (instead

of one) should be computed. To distinguish the two approaches we will denote the first one by $\mathbf{u}_{ap} = POD_1(\mathbf{u})$, while its variation will be denoted by $\mathbf{u}_{ap} = POD_2(\mathbf{u})$.

2. The eigenvalues and eigenvectors $(a_q(t))$ of $\mathbf{C}$ are calculated at each time step, then the spatial modes $\Psi_i^q(\mathbf{x}), q = 1, ..., Q_R$ are computed. The value of $1 \leq Q_R \leq Q$ is computed at each time step using the following criteria:

   a) The total energy is computed: $E_{1,Q} = \sum_{k=1}^{Q} \lambda_q$. The value of $1 \leq Q_R \leq Q$ is then chosen such that the first Q_R eigenvalues contribute $E_{1,Q} - TOL\_POD$ of the kinetic energy.

   b) Alternatively, the threshold criteria can be applied, such that the Q_R parameter satisfies $\lambda_q < TOL\_POD, \quad \forall q > Q_R$.

3. The spatial modes $\Psi_i^q(\mathbf{x}), q = 1, ..., Q_R$ are computed. Typically $Q_R < Q$, which leads to considerable computational savings.

4. The values of $a_q(t^{n+1})$ are computed using formula (3.9).

5. The approximate solution field is computed from

$$\mathbf{u}_{ap}(t^{n+1}, \mathbf{x}) = \sum_{q=1}^{Q_R} a_q(t^{n+1}) \Psi_i^q(\mathbf{x}). \tag{3.11}$$

Note, that $\mathbf{u}_{ap}(t^{n+1}, \mathbf{x})$ is computed in physical space and then it is transformed into the modal space which is typically the most computationally expensive step of the POD-based extrapolation.

Computational complexity: The POD-based extrapolation technique requires $2 \times Q \times Nqp \times sizeof(double)$ bytes storage per field - half for solution at previous time steps and half for spatial modes. Memory requirements for the correlation matrix $\mathbf{C}$ is $Q \times Q$ and the same memory is required to store the temporal modes. At each time step the following arithmetics is required:
i) Q values of $\mathbf{C}$ are computed via numerical integration
$C_{1,i} = \sum_{j=1}^{Nqp} [u(t^n, \mathbf{x}_j) u(t^{n-i}, \mathbf{x}_j) w_j], i = 0, ..., Q-1$, here w_j are the integration weights; This requires one *dvmul* operation[2] and Q vector-vector dot products.

[2]function *dvmul* computes $z[i] = x[i] \cdot y[i]$.

ii) eigenvalues and eigenvectors of $\mathbf{C}$ are computed by *dsyev* function[3].

iii) Q_R spatial modes are computed by performing one *dscal* and $Q_R - 1$ *daxpy* operations for each field.

iv) field $\mathbf{u}_{ap}$ is computed from the spatial and temporal modes, this operation holds the same computational complexity as (iii).

v) Transformation of a solution into modal space, which computational complexity has been estimated in section 3.5.1.

[3]function *dsyev* computes all eigenvalues and, optionally, eigenvectors of a real symmetric matrix.

3.5.3 Results

To investigate numerically the effectiveness of the extrapolation methods we considered several cases: a) turbulent flow in stenosed carotid artery, b) transient flow in the domain of brain blood vessels, consisting of very large number of spectral elements. Before we proceed further we want to familiarize the reader with the following notations:

Nel - number of spectral elements.

$\Delta t = t^n - t^{n-1}$ - time step size, n is the time step index.

P - order of polynomial expansion for spatial approximation.

In this section $\mathbf{x}^k$ denotes a solution vector at iteration k and vector $\mathbf{b}$ denotes a suitable forcing term.

$||a|| = \sqrt{\sum_{i=1}^{N} a_i^2}$.

$r^k = ||\mathbf{A}\mathbf{x}^k - \mathbf{b}||$ - residual, at iteration k.

$r^k_{Ns} = r^k/Ns$ - residual scaled by the length of vector $\mathbf{x}$ (number of unknowns).

$r^k_s = \frac{1}{||\mathbf{x}^0||}||\mathbf{A}\mathbf{x}^k - \mathbf{b}||$ - residual scaled by the initial guess.

The accuracy of solver has been verified by simulating unsteady flow in a pipe, forced by periodically varying pressure gradient. The numerical solution was compared to the analytical solution (Womersley velocity profile). In table 3.7 we show the error computed for u and w (streamwise) velocity components; comparable accuracy is achieved for all choices of the initial guess.

P	$\mathbf{x}^0 = \mathbf{u}^n$	$\mathbf{x}^0 = EXT(\mathbf{u})$	$\mathbf{x}^0 = POD_1(\mathbf{u})$
4	1.0E-3 (4.3E-3)	1.0E-3 (3.5E-3)	1.0E-3 (3.5E-3)
6	1.8E-5 (2.1E-4)	2.0E-5 (5.2E-5)	2.0E-5 (5.2E-5)
8	9.9E-7 (7.3E-6)	8.5E-7 (3.2E-6)	8.5E-7 (3.2E-6)

Table 3.7: Simulations of unsteady flow in a pipe: L_∞-error for u (w - streamwise) velocity components. Initial guess ($\mathbf{x}^0$) for conjugate gradient solver is provided by: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - simple extrapolation with $N = 4$, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ - POD-based extrapolation with $Q = 4$. $\Delta t = 2.0E-4$, stopping criteria for conjugate gradient solver: $r^k_s < TOL\_CG = 1.0E-12$

Performance of the preconditioned conjugate gradient solver in conjunction with different acceleration techniques has been examined by monitoring the residual and relative error in the solution. Unsteady flow simulations in the domain of carotid artery were performed

using the three approaches for initial guess approximation, namely, $\mathbf{x}^0 = \mathbf{u}^n$, $\mathbf{x}^0 = EXT(\mathbf{u})$ and $\mathbf{x}^0 = POD_1(\mathbf{u})$. The stopping criteria for the iterative solver was $r^k < 1E-8$. In figure 3.19 we show typical convergence rate of r^k_{Ns} and of the relative error in the solution $||\mathbf{x}^k - \mathbf{x}^{k-1}||/Ns$, where Ns is the size of vector $\mathbf{x}$, in the simulations considered in this section Ns is of order $O(1E5)$ to $O(1E6)$. In solutions of Helmholtz equations for velocity with LEBP the convergence rate of the residual was $\log_{10}(r^k_{Ns}) \propto -0.3k$. In simulations with diagonal preconditioner we obtained convergence rate $\log_{10}(r^k_{Ns}) \propto -0.021k + f(k)$ for all choices of initial guess, here $f(k)$ is a periodically varying function reflecting non-uniform convergence rate. In figure 3.19 we present the convergence rate monitored in the aforementioned simulations. The data in figure 3.19 was collected at time step $n = 150$, analysis of data from other time steps provided similar results. The significant difference in the initial residuals, observed in figure 3.19, and consequently in the number of iterations is due to the choice of initial guess.

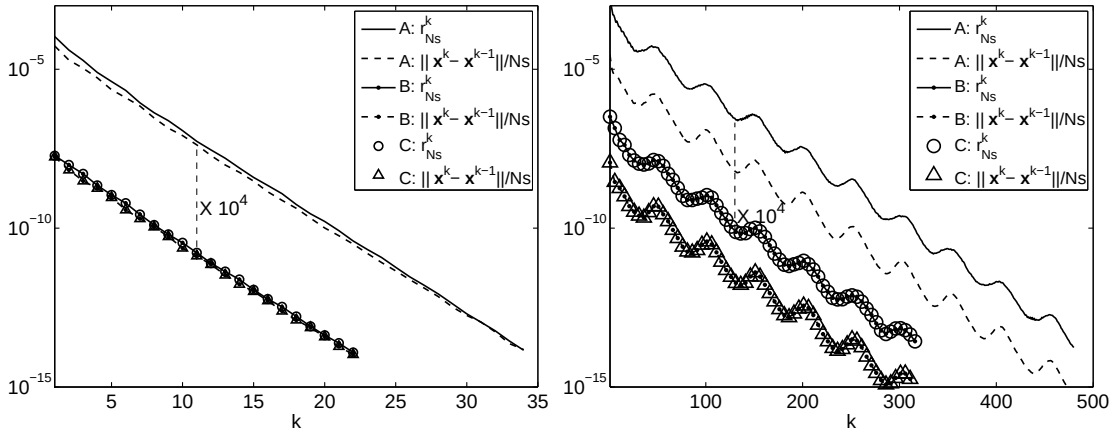


Figure 3.19: Preconditioned conjugate gradient solver: effect of improved initial guess. Convergence of residual (r^k_{Ns}) and solution ($||\mathbf{x}^k - \mathbf{x}^{k-1}||/Ns$). $A : \mathbf{x}^0 = \mathbf{u}^n$, $B : \mathbf{x}^0 = EXT(\mathbf{u})$, $N = 3$, $C : \mathbf{x}^0 = POD_1(\mathbf{u})$, $Q = 3$. Solution with low energy (left) and diagonal (right) preconditioners. $P = 6$, $\Delta t = 1.0E-4$, $Nel = 22,441$.

The data presented in figure 3.19 shows that the difference between two consecutive solutions ($\mathbf{x}^k - \mathbf{x}^{k-1}$) is bounded by the residual, moreover, in simulation with LEBP $||\mathbf{x}^k - \mathbf{x}^{k-1}|| \approx r^{k-1}$. Indeed,

$$\mathbf{PA}(\mathbf{x}^k - \mathbf{x}^{k-1}) = \mathbf{r}^k - \mathbf{r}^{k-1} \Rightarrow ||\mathbf{PA}(\mathbf{x}^k - \mathbf{x}^{k-1})|| < ||\mathbf{r}^k|| + ||\mathbf{r}^{k-1}|| < 2||\mathbf{r}^{k-1}||,$$

where $\mathbf{PA}$ is a preconditioned operator $\mathbf{A}$. The LEBP is spectrally close to the operator

$\mathbf{A}$, hence $\mathbf{PA} \approx \mathbf{I}$, which explains the short distance between $||\mathbf{x}^k - \mathbf{x}^{k-1}||$ and r^{k-1}.

The relatively constant rate of convergence of r^k_{Ns} and weak dependence of the condition number κ corresponding to the LEBP on the computational grid allow good estimate of the number of conjugate gradient iterations for the Helmholtz solver:

$$Nit \approx \frac{\log_{10}(r^0/TOL\_CG)}{0.3}.$$

In order to estimate the Nit the initial residual must be estimated in terms of known parameters such as Δt and TOL_CG. The value of r^0 is related to the accuracy in prediction of the initial guess x^0:

$$\mathbf{A}\mathbf{x}^K - \mathbf{b} = \mathbf{r}^K, \mathbf{A}\mathbf{x}^0 - \mathbf{b} = \mathbf{r}^0 \Rightarrow ||A(\mathbf{x}^0 - \mathbf{x}^K)|| \leq ||\mathbf{r}^0|| + ||\mathbf{r}^K|| < ||\mathbf{r}^0|| + TOL\_CG,$$

here K is the index of the last iteration. From the last equation we obtain $||A(\mathbf{x}^0 - \mathbf{u}^{n+1})|| < ||\mathbf{r}^0|| + TOL\_CG$. That is estimating accuracy of prediction of the initial guess $||\mathbf{x}^0 - \mathbf{u}^{n+1}||/Ns$ will provide an estimate on r^0_{Ns}.

In the case of $\mathbf{x}^0 = \mathbf{u}^n$ the initial residual is $r^0_{Ns} \approx \Delta t \dot{\mathbf{u}}$ (where $\dot{\mathbf{u}} = \partial \mathbf{u}/\partial t$), which is the dominant error.

In the case $\mathbf{x}^0 = EXT(\mathbf{u})$ the initial residual depends on several factors:

a) the extrapolation error $\epsilon_{ext} \propto (\Delta t)^N \mathbf{u}^{(N)}$. The upper limit corresponds to $N = 1$, that is the error is of order $O(\Delta t)\dot{\mathbf{u}}$.

b) If the extrapolation error is very small then the dominant error in prediction of the solution at time step t^{n+1} will be due to termination of the conjugate gradient iterations at previous time steps prior to reaching $r^k = 0$, and the error can be estimated as $\epsilon_{CG} \approx TOL\_CG/Ns$. Assuming that $||\mathbf{PA}|| = \gamma_{LEBP}||\mathbf{I}||$, with $\gamma_{LEBP} > 1$ we can relate ϵ_{CG} to the residual as $\epsilon_{CG} = ||\hat{\mathbf{x}} - \hat{\mathbf{x}}^K||/Ns \leq TOL\_CG/Ns$, where $\hat{\mathbf{x}}$ is the exact solution of $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$ and $\hat{\mathbf{x}}^K$ is the solution obtained at the last conjugate gradient iteration. The data presented in figure 3.19 shows that for LEBP the value of $\gamma_{LEBP} \approx 1+\delta$, while for the diagonal preconditioner $\gamma_{DP} >> \gamma_{LEBP}$. Then $\mathbf{x}^0 = EXT(\mathbf{u}) + EXT(TOL\_CG) = EXT(\mathbf{u}) + TOL\_CG$. The ϵ_{CG} also contaminates the right-hand-side of the Helmholtz equation through the nonlinear and pressure term, we can estimate that the last error is of order $C_1 TOL\_CG$. Numerical experiments indicate that the constant C_1 is very small and it is in the range of 1 to 10. If the extrapolation error is negligible, the convergence of the Helmholtz solver is

expected in just one to $\log_{10}(10)/0.3 \approx 3.3$ iterations.

In the case of $\mathbf{x}^0 = POD(\mathbf{u})$ the initial residual depends on:

a) the error associated with evaluation of the temporal and spatial POD modes. This error increases if the correlation matrix is illconditioned.

b) extrapolation error in computing $a_q(t + \Delta t)$ this error can be estimated as $\epsilon_{ext} = (\Delta t)^Q \sum_{q=1}^{Q_R} \left(\lambda_q a_q^{(Q)}\right)$. The first temporal mode $a_1(t)$ is typically smooth function and high-order derivatives of $a_1(t)$ are negligible. It is usually the high-order temporal modes that tend to oscillate, fortunately the contribution of this modes is usually negligible, i.e., $\lambda_q/E << 1, \quad q > 1$.

c) error due to termination of the conjugate gradient iterations ϵ_{CG}, which already has been estimated above.

Note that high-order polynomial extrapolation may lead to erroneous results related to high Lebesgue constants, which grows exponentially for equidistant grid. In the examples provided bellow we show that the number of conjugate gradient iterations (with LEBP) indeed fits the provided estimate.

The results presented in table 3.7 and figure 3.19 suggest that, in terms of accuracy, the performances of a simple and POD-based accelerator are equivalent. Both acceleration techniques lead to significant reduction in the iteration count. There are several choices to define stopping criteria for the conjugate gradient solver. In the performed experiments the iterations were terminated when $r^k < TOL\_CG = 1E{-}8$. The solution-independent criteria allowed consistent comparison between the three methods, however, the TOL_CG value used for the tests was very low. In the following experiments we will use TOL_CG typically employed in many engineering applications and the stopping criteria for the iterative solver will be $r^k < TOL\_CG = 1E - 5$.

Turbulent flow simulations in stenosed carotid artery

In this section we consider a simulation of unsteady flow in domain of stenosed carotid artery. The narrowing of the internal carotid artery creates a strong jet-flow, due to curvature and geometric asymmetry the jet adheres to arterial wall and becomes unstable when flow exceeds certain speed. In figure 3.20 we present typical velocity data, monitored downstream of the stenosis, high frequency oscillations signify turbulence.

In table 3.8 we compare the effectiveness of the acceleration techniques, both technique

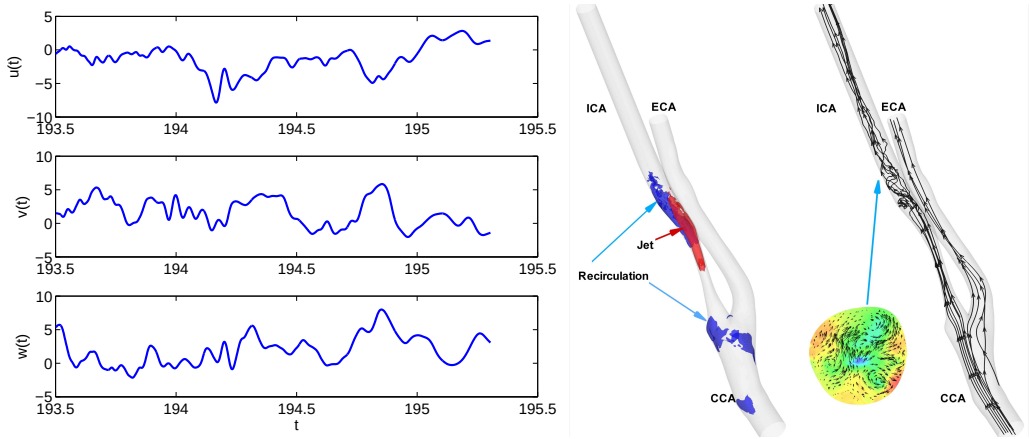


Figure 3.20: (in color) Unsteady flow simulations in stenosed carotid artery. Left: Non-dimensional velocity monitored downstream stenosis. The oscillations signify unsteadiness of a flow. Right: a high-speed region - red iso-surfaces, back-flow regions - blue iso-surfaces; instantaneous path-lines of swirling flow and cross-stream secondary flows.

	$\mathbf{x}^0 = \mathbf{u}^n$		$\mathbf{x}^0 = EXT(\mathbf{u})$, $\mathbf{N = 3}$			$\mathbf{x}^0 = POD(\mathbf{u})$, $\mathbf{Q = 3}$		
P	r^0_{Ns}	Nit	r^0_{Ns}	Nit	T_e	r^0_{Ns}	Nit	T_e
6	2.8E-4	24	1.6E-8	9.1	1.3E-3	1.6E-8	9.1	3.6E-2
10	1.7E-5	19.2	1.9E-9	7.1	1.7E-3	1.9E-9	7.1	1.1E-1

Table 3.8: Turbulent flow simulations in stenosed carotid artery: performance of iterative solver with different initial guesses. r^0_N, Nit, T_e - average (over time) initial residual (normalized by the number of unknowns), number of iterations (Nit) and CPU-time (in seconds) required for extrapolation at each time step. The initial guess is provided by: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n is provided as an initial guess, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - initial guess computed by a simple extrapolation, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ - initial guess computed with POD. Data is averaged over time steps 100 to 4,000, preconditioner: low energy; $\Delta t = 5.0E-5$, $P = 6, 10$, $Nel = 22,441$.

show comparable performance. Reduction of the iteration count in simulation with higher resolution ($P = 10$) is intriguing. First, we note that the LEBP exhibits excellent p- and h-scaling. Second, we may point out to two reasons for such behavior:

i) The accuracy of extrapolation depends on the smoothness of the extrapolated function. In a separate study we verified that relatively accurate solution for the given problem may be obtained with $P \geq 8$. Simulation with $P = 6$ leads to under resolved solution and erroneous oscillations, which disappear in simulations with $P = 10$.

ii) Stopping criteria for conjugate gradient iterations. In both simulations the iterations were terminated when $r^k < TOL\_CG = 1E-5$. Due to higher resolution, the number

of unknowns corresponding to $P = 10$ is higher than in simulation with $P = 6$, hence $r^k_{Ns_{P=6}} > r^k_{Ns_{P=10}}$. Lower tolerance stopping criteria minimizes the error of iterative solver. Assume that $\mathbf{x}_{P_1}$ was computed with higher spatial resolution than $\mathbf{x}_{P_2}$, but with the same stopping criteria for the conjugate gradient iterations. Let DOF_1 (DOF_2) be the number of unknown degrees of freedom corresponding to $\mathbf{x}_{P_1}$ ($\mathbf{x}_{P_2}$), then $r^K_{DOF_1} < r^K_{DOF_2}$ where K signifies that the residual corresponds to the last iteration, i.e. $r^K < TOL\_CG$, consequently

$$\mathbf{x}_{P_1}(t) = \mathbf{x}_{P_2}(t) + error(t), \quad error(t) \approx TOL\_CG/DOF_1. \tag{3.12}$$

Next, lets apply the extrapolation formula on both parts of (3.12) to obtain the initial guess for solution at time step $t + \Delta t$. The predicted solution based on the right hand side of (3.12) is polluted by an extrapolated error term, which limits its accuracy.

The lower number of iteration corresponding to high-order polynomial approximation is appealing particularely in the context of parallel computing. It is well known that better scalability is achieved with higher order polynomial expansion since the ratio of computation versus communication is increasing. We recall that the most communication intensive steps of the Navier-Stokes solver are the Helmholtz and Poisson solvers; for example, the calculation of the non-linear term requires no communication at all. The low iteration count reduces the communication cost in the conjugate gradient solver and the volume of computation linearly, hence the overall scalability of the Navier-Stokes solver increases, particularly for high P.

It is reasonable to assume that higher order extrapolation may lead to more accurate results, as long as the extrapolated field is sufficiently smooth. In table 3.9 we present results obtained with different order of extrapolation ($N, Q = 3$, $N, Q = 4$ and $N, Q = 5$). About two order of magnitude improvement in the accuracy of initial guess $\mathbf{x}^0 = EXT(\mathbf{u})$ is observed, and the reduced number of iterations reflect this improvement. However, fifth-order accurate extrapolation ($N = 5$) does not lead to lower initial residual, which can be explained by insufficient TOL_CG and possible Runge effect related to high-order interpolations on equidistant grid. In the case of $\mathbf{x}^0 = POD_1(\mathbf{u})$ and $Q = 4$ the improvement is not significant; and with higher POD expansion order ($Q = 5$) it even gets worse. The relatively poor performance of the POD-based extrapolation is due to illconditioned correlation matrix. For example, the condition number of the correlation matrix formed at time step

410 is of order 10^{16}. Large condition number results in error in computation of temporal and spatial POD modes, and, consequently, relatively high extrapolation error.

	$\mathbf{x}^0 = EXT(\mathbf{u})$			$\mathbf{x}^0 = POD_1(\mathbf{u})$		
	r^0_{Ns}	Nit	T_e	r^0_{Ns}	Nit	T_e
N,Q=3	1.6E-8	9.1	1.3E-3	1.6E-8	9.1	3.6E-2
N,Q=4	2.75E-10	3.7	1.3E-3	1.16E-8	6.75	3.9E-2
N,Q=5	1.94E-10	3.7	1.3E-3	3.33E-8	9.2	5.5E-2

Table 3.9: Turbulent flow simulations in stenosed carotid artery: high-order extrapolation. $r^0_N s$, Nit, T_e - average (over time) normalized initial residual number of iterations (Nit) and CPU-time (in seconds) required for extrapolation at each time step. The initial guess is provided by: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n is provided as an initial guess, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - initial guess computed by a simple extrapolation, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ - initial guess computed with POD. Data is averaged over time steps 100 to 4,000, preconditioner: low energy; $\Delta t = 5.0E-5$, $P = 6$, $Nel = 22,441$.

In tables 3.8 and 3.9 we showed data averaged over many time steps and over the three components of the velocity field. In figure 3.21 we compare the number of iterations required at every time step for each of the three velocity components (u, v, w), up to eight-fold reduction in iteration count is observed. The reduction is due to improved initial guess, which minimize the initial residual r^0, also shown in figure 3.21. In table 3.10 we compare the number of iterations and initial residual (averaged over 40,000 time steps) for three choices of the initial guess: a) solution from time step t^n, b) velocity field extrapolated from four previous time steps, and c) velocity field reconstructed by POD technique, also using data from four previous time steps. Slightly higher number of iterations is observed when the initial guess is computed using POD.

The low iteration count observed in the aforementioned simulation was due to two reasons: 1) good initial guess and 2) application of very effective parallel low energy preconditioner. The low energy preconditioner requires considerable computational effort, blocking collective and non-blocking point-to-point communications and also matrix-vector multiplications. The diagonal preconditioner requires no communication and insignificant computational effort, hence it might be feasible to use the less effective but embarrassingly parallel preconditioner in conjunction with the methods providing a good initial guess. In table 3.11 we compare the performance of the conjugate gradient solver with the low energy and diagonal preconditioner and two choices of initial guess. It is clearly seen that the choice of

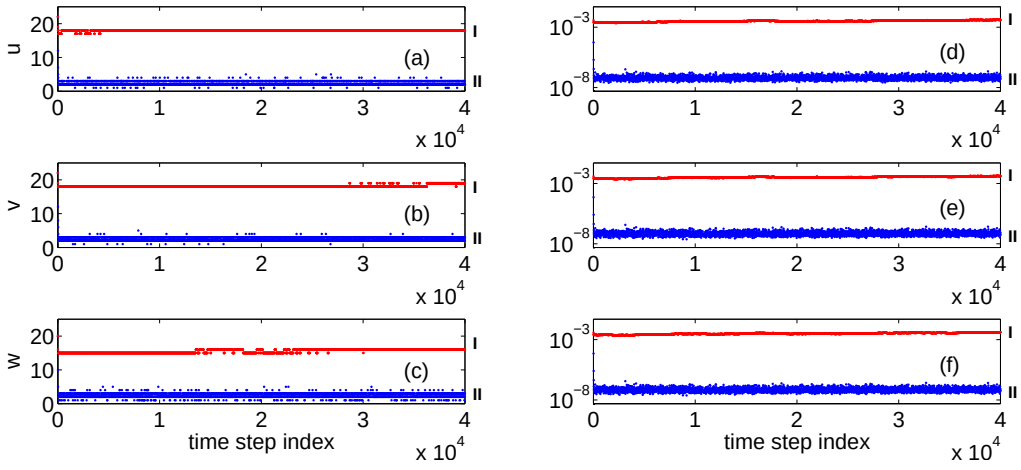


Figure 3.21: Turbulent flow simulations in stenosed carotid artery: (a-c) - number of iterations required by iterative Helmholtz solver for three velocity components. (d-f) - initial residual $r_N^0 = f(u_{ap})$, curve marked by **I** (**II**) correspond to $\mathbf{u}_{ap}^{n+1} = \mathbf{u}^n$ ($\mathbf{u}_{ap}^{n+1} = EXT(\mathbf{u}),\ N = 4$). $\Delta t = 5.0E-5$, $P = 6$, $Nel = 22,441$.

low energy preconditioner is advantageous.

	$\mathbf{x}^0 = \mathbf{u}^n$		$\mathbf{x}^0 = EXT(\mathbf{u})$		$\mathbf{x}^0 = POD(\mathbf{u})$	
	Nit	$\bar{r}_{Ns}^0$	Nit	$\bar{r}_{Ns}^0$	Nit	$\bar{r}_{Ns}^0$
u	17.97	2.60E-3	2.31	6.80E-8	3.33	1.99E-7
v	18.12	3.20E-3	2.42	7.56E-8	3.55	2.54E-7
w	15.53	5.46e-3	2.36	7.65E-8	2.48	6.62E-8

Table 3.10: Turbulent flow simulations in stenosed carotid artery: average number of iterations (Nit) and average initial residual $\bar{r}_N^0$ for three choices of initial guess: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n is provided as an initial guess, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - initial guess computed by a simple extrapolation with $N = 4$, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ - initial guess computed by via POD with $N = 4$. Data is averaged over time steps 100 to 40,000, preconditioner: low energy; $\Delta t = 5.0E-5$, $P = 6$, $Nel = 22,441$.

Simulations of a transient flow in Circle of Willis

In this section we consider a simulation of transient flow in domain of Circle of Willis (CoW) reconstructed from MRI images of a human brain. The simulation is started from the initial solution $\mathbf{u}(t = 0) = 0$, steady velocity profile is imposed at three inlets of the domain. The simulation time considered here is significantly lower than the time required to establish a steady state solution. The size of the computational domain of CoW comparing to the domain of carotid artery is very large. CoW consists of 22 arteries and is discretized into

	Nit		CPU-time (sec)	
preconditioner	$\mathbf{x}^0 = \mathbf{u}^n$	$\mathbf{x}^0 = EXT(\mathbf{u})$	$\mathbf{x}^0 = \mathbf{u}^n$	$\mathbf{x}^0 = EXT(\mathbf{u})$
low energy	17.21	2.36	0.188	0.037
diagonal	170.8	37.29	1.09	0.25

Table 3.11: Turbulent flow simulations in stenosed carotid artery: average number of iterations (Nit) and average CPU-time required by three Helmholtz solves (for the three velocity components together) for two choices of initial guess: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n is provided as an initial guess, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - initial guess computed by a simple extrapolation with $N = 4$; and two preconditioners: low energy and diagonal; $\Delta t = 5.0E-5$, $P = 6$, $Nel = 22,441$.

$Nel = 162,909$ tetrahedral spectral elements. The flow is laminar and is characterized by regions of recirculations and swirl. The main point of this section is to show that the aforementioned techniques aiming to provide a good initial guess for iterative solver can be successfully applied in simulations of a flow in very large computational domains. The stopping criteria for conjugate gradient iterations is set to $r^k < TOL\_CG = 1E-5$ ($r^k_{Ns} \approx 7.9E-12$). In figure 3.22 the convergence rate of residual is presented. The data corresponds to time step 500; the presented residuals correspond to the $u-$ component of the velocity field, the results for $v-$ and $w-$ components are similar. We observe that that an accurate prediction of the initial guess for PCG results in significantly faster convergence. Another observation is that the rate of convergence of the residual is the same as in the previous study, i.e., $\log_{10}(r^k_{Ns}) \propto -0.3k$ and is independent of the initial guess.

In table 3.12 we compare the number of iterations required by the Helmholtz solver with respect to three choices of initial guess. Similarly to the turbulent flow simulations, p-refinement results in lower iteration count. The effect of improved initial guess is noticeable. To understand the poorer performance of the POD-based extrapolation we monitor the eigenspectra of the correlation matrix C at every time step (see table 3.13). The very fast decay in the eigenspectra signifies that the correlation length between velocity fields is significantly greater then the size of a time step, which results in ill-conditioned correlation matrix. The correlation length in this simulation asymptotically approaches to infinity since $\partial\mathbf{u}/\partial t \to 0$. The negative sign of the smallest eigenvalue λ_4 is due to ill-conditioned $\mathbf{C}$. Since the number of modes used for reconstruction of the velocity field (Q_R, see the forth column in table 3.13) is computed dynamically, the negative λ_4 will restrict the $Q_R < 4$. We can observe that the number of iteration is correlated with the Q_R parameter, and for

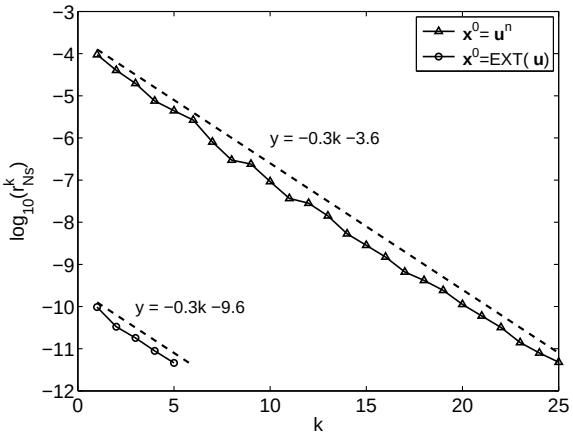


Figure 3.22: Flow simulations in Circle of Willis: effect of improved initial guess and convergence of residual (r^k_{Ns}) in solution with preconditioned conjugate gradient solver. The dash lines depict the linear least square approximation for the convergence rate. $N = 4$, $P = 4$, $Nel = 162,909$, $\Delta t = 5.0E-4$, preconditioner: low energy.

$Q_R = 4$, the number of iteration is comparable to what is achieved with $\mathbf{x}^0 = EXT(\mathbf{u})$.

	$\mathbf{x}^0 = \mathbf{u}^n$		$\mathbf{x}^0 = EXT(\mathbf{u})$		$\mathbf{x}^0 = POD(\mathbf{u})$	
	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$
P=3	24.60	2.99E-4	3.70	3.34E-10	6.81	9.70E-9
P=4	22.15	6.01E-5	3.15	1.06E-10	5.81	2.05E-9
P=5	20.85	1.77E-5	3.27	7.22E-11	4.94	5.90E-10
P=6	18.80	8.20E-6	2.67	4.50E-11	4.06	2.87E-10

Table 3.12: Flow simulations in Circle of Willis: performance of iterative solver with different initial guesses. Average number of iterations (Nit) and average normalized initial residual $\bar{r}^0_N$ for three choices of initial guess: a) $\mathbf{x}^0 = \mathbf{u}^n$ - solution at time step t^n is provided as an initial guess, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ - initial guess computed by a simple extrapolation with $N = 4$, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ - initial guess computed by via POD with $Q = 4$. Data is averaged over time steps 300 to 3,000. $Nel = 162,909$, $\Delta t = 5.0E-4$, preconditioner: low energy.

Acceleration of Poisson solver for the pressure

In order to accelerate convergence of the Poisson solver for the pressure, the extrapolation of the pressure field from the solutions at the previous time step has been applied. In figure 3.23 we compare the number of iterations required by the Poisson solver and plot the convergence rate of the residual. Computing initial guess from for the pressure solver by the extrapolation indeed reduces the number of conjugate gradient iterations. In many numer-

U_{it}	V_{it}	W_{it}	Q_R	λ_4	λ_3	λ_2	λ_1
7	6	7	3	-8.03e-14	3.37e-10	9.75e-05	1.38e+04
7	7	7	3	-8.96e-13	3.35e-10	9.74e-05	1.38e+04
4	4	4	**4**	**1.01e-12**	3.37e-10	9.73e-05	1.38e+04
7	6	7	3	-1.26e-12	3.38e-10	9.72e-05	1.38e+04
8	8	8	3	-6.90e-13	3.36e-10	9.72e-05	1.38e+04
7	7	7	3	-7.13e-13	3.39e-10	9.71e-05	1.38e+04
4	3	4	**4**	**1.57e-12**	3.39e-10	9.70e-05	1.38e+04
7	6	7	3	-1.86e-12	3.40e-10	9.70e-05	1.38e+04
4	3	4	**4**	**4.05e-13**	3.39e-10	9.69e-05	1.38e+04
3	3	3	**4**	**1.41e-13**	3.39e-10	9.68e-05	1.38e+04

Table 3.13: Flow simulations in Circle of Willis: performance of POD-based accelerator. U_{it}, V_{it} and W_{it} - number of iterations required for solution of Helmholtz equations for velocity at times teps 2991 to 3000. Q_R - number of POD modes used for velocity field reconstraction. $\lambda_i,\ i = 1, 2, 3, 4$ - eigenvalues of correlation matrix $\mathbf{C}$. $Q = 4$, $P = 4$, $Nel = 162,909$, $\Delta t = 5.0E-4$, preconditioner: low energy.

ical experiments we observed that improving the initial guess for the Poisson solver by the polynomial extrapolation with $N = 2, 3$ reduced the number of iteration by approximately 50%, however, for higher order extrapolation ($N \geq 4$) the iteration count started to grow.

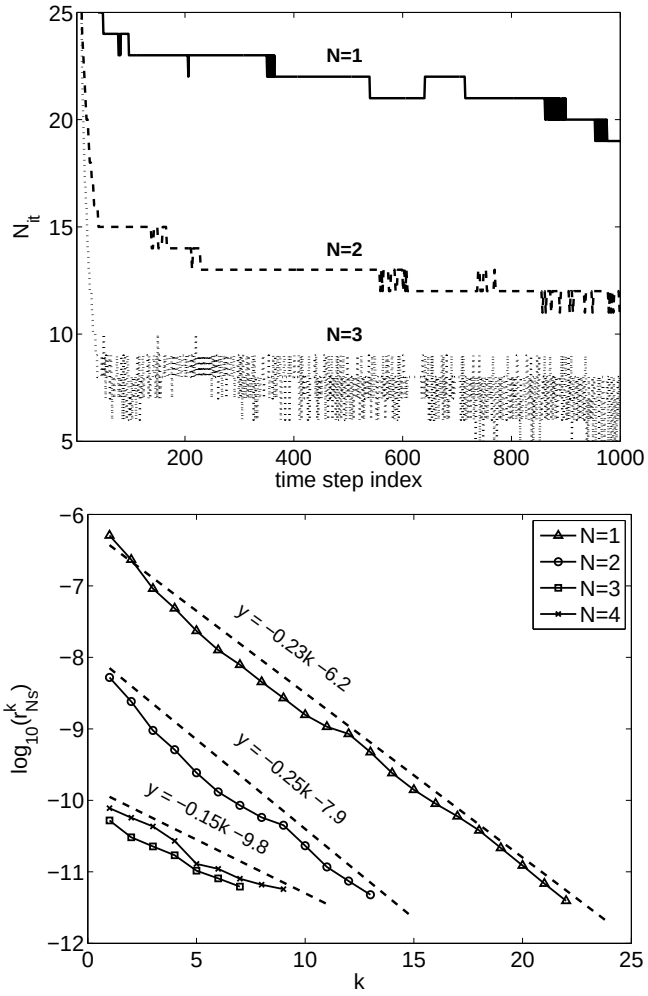


Figure 3.23: Flow simulations in Circle of Willis: effect of improved initial guess for the pressure and convergence of residual (r^k_{Ns}) in solution with preconditioned conjugate gradient solver. Top: number of iterations required by Poisson solver for the pressure during the first 1000 time steps. Bottom: convergence of residual at time step 500. The dash lines depict the linear least square approximation for the convergence rate. $P = 4$, $Nel = 162,909$, $\Delta t = 5.0E - 4$, preconditioner: low energy.

3.6 Summary

Chapter 3 focuses on robustness of iterative solution of linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric positive definite matrix. The linear system is solved using preconditioned conjugate gradient method.

In the first part of Chapter 3 we extend the work of Sherwin&Casarin [76] on the low energy preconditioner and associated with it coarse space linear vertex solve preconditioner. Specifically, we improve the robustness of the low energy preconditioner for prismatic elements, and developing a parallel coarse space linear vertex solver. Platform-specific optimizations were considered. Very good scalability in the solution of large scale problems on up to 4096 processors was achieved. In particular, the parallel efficiency increases with the polynomial order of the spectral/*hp* element. The preconditioner was successfully applied to simulations of unsteady bioflows in domains with complex geometry, where the dominant cost is due to the elliptic solves. We show that the LEBP and the coarse space linear vertex solver scale on up to 4096 processes of CRAY XT3 and 3072 processes of BlueGene/L (maximum available at the time of the study).

In the second part of Chapter 3 methods to further increase effectiveness of iterative solution of a system $\mathbf{Ax} = \mathbf{b}$ are discussed. Specifically, two methods for approximation of the initial guess are studied. As a benchmark problem unsteady 3D flow simulations in very large and geometrically complex domain of arterial tree have been considered. The acceleration techniques where applied for iterative solution of the Helmholtz equation for the velocity and Poisson equation for the pressure. We show that the convergence of PCG of Helmoltz solver can be achieved without sacrificing accuracy in just 2-4 iterations, in addition the computational time and the parallel efficiency of a solver is improved. We show that the convergence of PCG of Poisson solver for the pressure can also be improved but since the pressure is not a dynamic variable, the accelerations techniques considered in this study are less effective with respect to the Helmholtz solver for velocity.

The accurate estimate of solution at time step t^{n+1} has utmost importance in implicit solution of non-linear partial differential equations. In general implicit solvers use large size of time-step than their explicit counterpart due to superior stability. Extrapolation of numerical solution on coarse mesh with POD might be have have a better accuracy than the simple polynomial extrapolation, this assumption must be verified in the future research.

Here results corresponding to POD_1 based extrapolation have been presented. Our numerical experiments have shown that the use of $\mathbf{x}^0 = POD_1(\mathbf{u})$ and $\mathbf{x}^0 = POD_2(\mathbf{u})$ techniques in most cases provided with similar results. The advantages of $\mathbf{x}^0 = POD_2(\mathbf{u})$ appeared only in one simulation - unsteady flow in a pipe, where according to analytical solution only one (streamwise) w-velocity component is not zero while the other two components $u = v = 0$. In numerical solution values of $u,\ v$ differ from the analytical solution due to discretization error. Extrapolation with with $\mathbf{x}^0 = POD_1(\mathbf{u})$ considers a single correlation matrix constructed from the inner product of the all three velocity fields, and consequently only one set of eigenvectors is computed. The number of eigenmodes (Q_R) employed to reconstruct the velocity field for all three components is then the same, which may adversely affect the accuracy in prediction of the initial state for the $u,\ v$ velocity components. Specifically, an amplification of the numerical error may occur which consequently will result in providing initial state that will only increase the number of iteration. Use of $\mathbf{x}^0 = POD_2(\mathbf{u})$ allows to set up the Q_R parameter for each velocity component independently, which in particular cases may help to reduce the number of iterations.

Chapter 4

Large-scale arterial flow simulation

4.1 Introduction

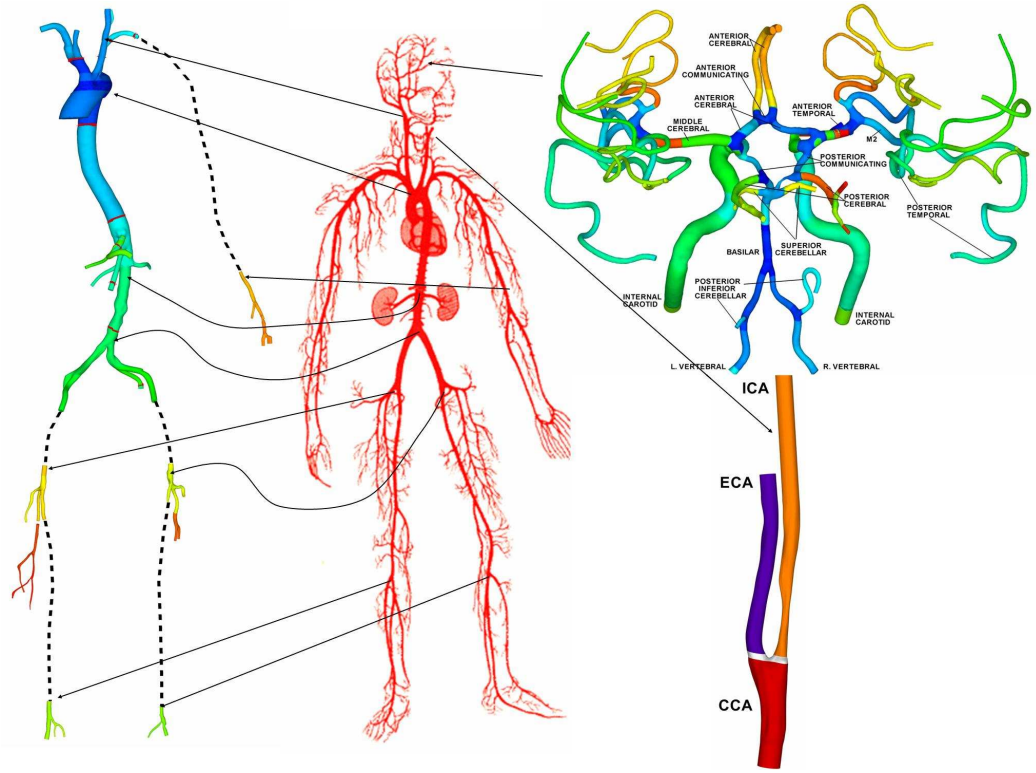


Figure 4.1: 3D model of major vessels and bifurcations of the human arterial tree reconstructed with gOREK from a set of CT, DSA CT and MRA images. Colors represent different parts of the model. Left: Aorta and adjacent arteries. Right top: Cranial arterial network. Right bottom: Carotid artery.

Blood circulation in the human arterial tree is the envy of every engineer: for an average

adult, the blood travels in just one minute more than 60,000 miles, that is 1/4 of the distance from the Earth to the moon. Simulating the human arterial tree is a grand challenge and requires state-of-the-art algorithms and computers. A simple estimate of the enormous resolution required can be obtained as follows: assuming that we use tetrahedral elements to descritize the typical blood volume for a human (5 liter), with a $0.5mm$ edge for each tetrahedron (and a corresponding volume of $0.0147mm^3$), we will require more than 339 million finite elements. For high-order discretizations, such as spectral/hp elements, the number of grid points required is then $339M \times (P+3)(P+2)^2$, which for (polynomial order) $P = 4$ results in approximately 85.5 billions of grid points, an enormous number indeed! Admittedly, this estimate is conservative because it is based on a uniform discretization of arteries, arterioles and capillaries. However, such an estimate points to the fact that, at present time, even with petaflop computing resources (performing up to 10^{15} operations per second), the brute force approach of a full 3D simulation at all scales is not feasible and, hence, a new hierarchical modelling approach for the human arterial tree is required; such modelling reduces the problem size and corresponding computational complexity and can be characterized by three distinct spatial length scales as follows:

1. *The macrovascular network (MaN) consisting of large arteries*, down to diameter of 0.5 mm, which are patient-specific and can be reconstructed from clinical imaging.

2. *The mesovascular network (MeN) consisting of small arteries and arterioles*, from 500 μm down to 10 μm, which follow a tree-like structure governed by specific fractal laws [26, 105]. For example, the human brain contains about 10 million small arteries and arterioles (this number is computed based on Murray's law [63, 75] with a modified index ($q = 2.5$) and asymmetric structure).

3. *The microvascular network (MiN) consisting of the capillary bed*, which follows a net-like structure. Its topological statistics have recently been quantified for the human brain in [26]. The typical number of capillary segments in the brain is more than 1 billion.

In order to make progress in simulating the human arterial tree, this hierarchical approach should involve simulations of two regimes. The first includes all arteries that can be accurately imaged clinically at the present time (e.g. MaN; see figure 4.1). The second

regime includes the subpixel dynamics ($MeN$and MiN, as described above) acting as closure to the large-scale arterial dynamics (MaN). Today, accurate simulations of the entire $MaN$are possible on the emerging petaflop computers, whereas significant simplifications are required for simulating $MiN$and MeN.

Over the past two decades, most studies of bioflows have considered a single bifurcation as a model geometry. Recently, the widely available parallel computers, mostly in the form of PC clusters, and their increasing use in biomechanics has led to simulations with multiple bifurcations [36, 97, 27, 28] and even full-scale arterial trees on the supercluster of distributed computers, such as the TeraGrid (http://www.teragrid.org), involving more than 20 bifurcations computed on thousands of processors [35].

Briefly we can present the evolution of the large-scale blood flow dynamics modeling as following. Till 2005 the state-of-the-art models considered 1D flow simulations in about a hundred of arteries. At the same time the 3D simulations were performed in usually simplified models of a few arterial bifurcations, while high resolution simulations where performed in very small computational domains.

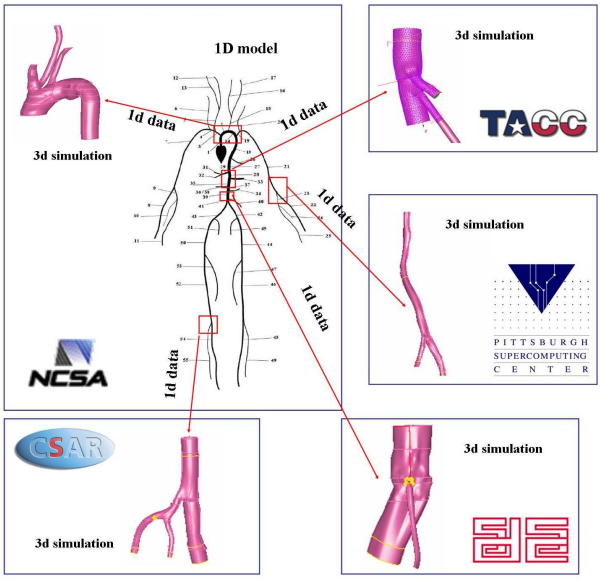


Figure 4.2: Blood flow simulation on distributed supercomputers. The disjoint 3D domains are coupled by 1D domains

The first 3D simulation of blood flow in *multiple* arterial segments coupled with real-time visualization was performed during the Supercomputing exhibition in Seattle at 2005 by Dong *et al.* [33]; this work considered a loose coupling of the 1D and 3D models. The

flowrates predicted by 1D simulation were imposed as inlet boundary conditions for *discontinuous* three-dimensional domains (see figure 4.2). The solution in the 3D domains was computed using hundreds of processors on four supercomputers, geographically distributed across USA and also on one supercomputer in the UK. The communication was handled using grid technology by MPICH-g2 library (recently replaced by the more advanced MPIg library). The requirement for distributed computing was due to an insufficient number of computer processors offered by each of the supercomputing centers. The new generation of massively parallel supercomputers allows performing such simulations within a single machine. From the perspective of parallel computing, this simulation was a significant achievement [24]. Due to discontinuity in the 3D representation of the arterial network the flow dynamics, particularly secondary flows, developed in one arterial segment did not propagate to adjacent segments, and important information on *interaction* of the flow at different arterial regions was lost. The contribution of the study presented in this Chapter is in stepping up to the next level of modeling and algorithmical complexity in *high-resolution* simulation of arterial flow in very large but *continuous* 3D computational domain. The new development is in *coupling* of 3D domains directly, i.e., establishing interface boundary conditions to exchange 3D data computed on both sides of interfaces in order to maintain C^0 continuity in velocity and pressure fields (see figure 4.3).

One of the first challenges in patient-specific arterial flow simulation is reconstructing vascular geometry of arterial tree from medical images. The next challenge is setting-up proper inflow-outflow boundary conditions. Flow simulations in geometrically complex arterial networks involve many inlets and outlets and, unless the closure problem is solved by simulating both $MeN$and MiN(a rather formidable task at present), proper boundary conditions should be used to account for the neglected dynamics downstream. Next challenge is performing numerical simulation. Current and projected advances in computer architectures involving hundreds of thousands of processors cannot be exploited for large-scale simulations of the human arterial tree (or of many other physical and biological problems) based on existing domain decomposition algorithms and corresponding parallel paradigms. Not only we have to address the tremendous complexity associated with data transfer amongst thousands of of processors, but more fundamentally the solution of linear systems with billions degrees of freedom (DOFs) and corresponding condition number exceeding one million is a rather formidable task.

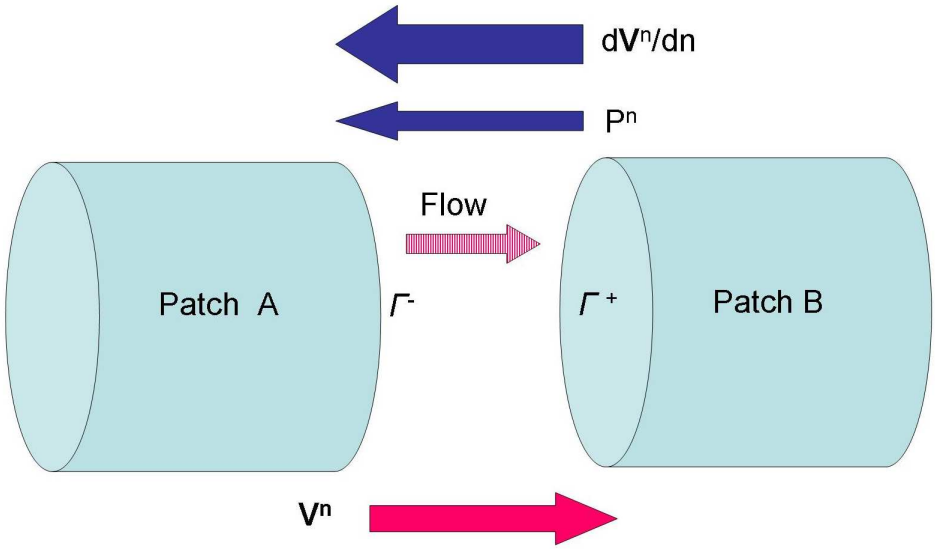


Figure 4.3: Two patches A and B are connected by the interface boundary conditions. Velocity computed at the outlet of A is imposed as Dirichlet boundary conditions at the inlet of B; pressure and velocity flux computed at inlet of B are imposed as boundary conditions at outlet of A.

Chapter 4 focuses on the three aforementioned challenges. In section 4.2 we briefly review existing techniques for processing medical images in order to reconstruct the vascular geometry. Then we present two new numerical methods developed for large scale 3D flow simulations in arterial networks counting hundreds of arteries. The *first method* is designed to seamlessly integrate an available clinical data (flow rates) in numerical simulations in order to impose *outlet* boundary conditions. The theoretical and numerical study on the new type of boundary conditions is presented in the first part of Chapter 4. The *second method* addresses the numerical and computational challenges in solving billions degrees of freedom problems on tens of thousands computer processors. Ultra-parallel flow simulations on hundreds of thousands of processors require new multi-level domain decomposition methods. In the second part of Chapter 4 we present such a new two-level method that has features of discontinuous and continuous Galerkin formulations. Specifically, at the coarse level the domain is subdivided into several big overlapping or non-overlapping patches and within each patch a spectral element discretization (fine level) is employed. New interface conditions for the Navier-Stokes equations are developed to connect the patches, relaxing the C^0 continuity and minimizing data transfer at the patch interface. Communication between thousands of processors is handled by Multilevel Communicating Interface (MCI).

We perform several 3D flow simulations of a benchmark problem and of arterial flows to evaluate the performance of the new method and investigate its accuracy. In the last section of this Chapter we focus on unsteady flow simulations of a flow in intracranial arterial tree. The 1D and 3D simulations are performed.

List of Symbols/Notations

Ω - computational domain.

$\mathbf{x}$ - cartesian coordinate system or solution space, depends on context.

ξ - coordinates of cartesian system, defined on Ω_e.

P - order of polynomial expansion.

$\mathbf{u} = [u\ v\ w]$ - velocity field.

p - pressure field.

R - resistance.

C - capacitance.

4.2 Reconstruction of vascular networks from medical images

Several techniques for obtaining information of arterial geometries exist:

- Computed tomography (CT): This method is based on the idea of the Italian radiologist Alessandro Vallebona who first suggested in the 1930s that it is possible to map a slice of a human body on a radiographic film. CT produces two-dimensional (2D) images (slices) of a human body, which, upon assembly in a 3D field, demonstrate various structures (organs) based on their ability to block the X-ray beam. CT angiography is an enhancement of the method, where a contrast material is injected into the blood to allow clear capturing of the blood volume and thus detection of the shape of the vessels' internal wall.

- Digital subtraction angiography CT, (DSA CT): This is a recent enhancement of the CT technique allowing a seamless 3D-based digital separation of vessels from the bones [56, 55]. This method is particularly valuable for visualization and reconstruction of the cranial arterial network, in figure 4.4(left) an example of DSA CT is provided.

- Magnetic resonance angiography (MRA): MRA is based on magnetic resonance imaging and provides images with superior contrast resolution compared to CT. The drawback of the MRA method for arterial shape reconstruction is its inability to capture a volume where blood is stagnant, since the method is based on detecting a moving fluid.

The aforementioned methods provide a set of 2D images which upon integration into a 3D data set is processed using different numerical procedures. Example of 3D image data is shown in figure 4.4(right). The reconstruction of arterial wall geometries is based on analysis of the data encoded in the images. Two common approaches for arterial wall identification are: (a) level set method - a numerical technique for identifying interfaces and shapes [101, 74], and (b) detection of arterial wall with a threshold method. The decision on what technique to use is typically based on the quality of the medical data and availability of proper software, however, at the present time there is no preferred method [62]. All methods have the same weakness in automatically distinguishing between vessel bifurcation and vessel fusion generated as an artifact in the image, as well as in the ability

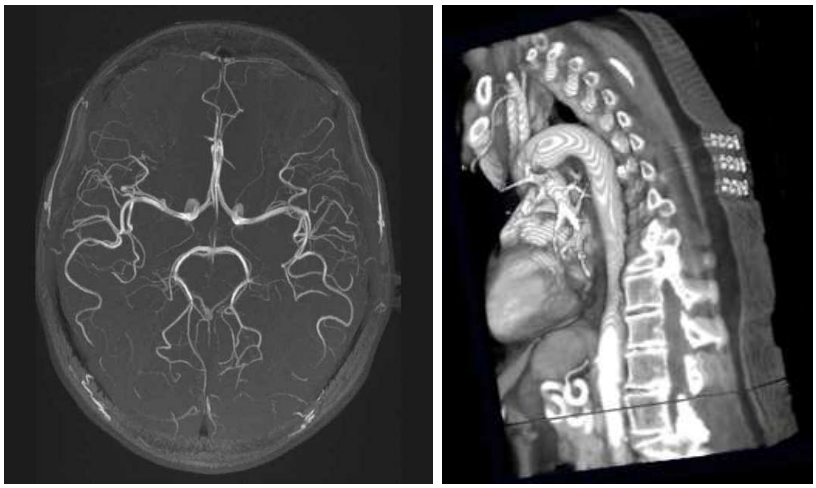


Figure 4.4: Computed tomography. Left: DSA CT of intracranial vasculature - view from above, the upper part of the image correspond to the front. Right: 2D images assembled into 3D domain. The large curved vessel is aorta.

to separate between adjacent bones and vessels. Another weakness of current automated arterial network reconstruction is in removing some features, such as presence of small vessels that are not supposed to be included in the simulation or, otherwise, in dealing with incomplete data when a part of the vessel is not seen.

To alleviate the aforementioned difficulties we developed a software package named gOREK[1] with a user-friendly Graphic User Interface, which allows the researcher to interact with medical data and to choose an appropriate numerical routine for reconstruction. The GUI is schematically presented in figure 4.5. Full automation of the geometry reconstruction process is not feasible, and in fact there is no commercial or free software which can guarantee automatic high quality geometry reconstruction and meshing. It is crucial to control the segmentation process to avoid "digitally" merged vessels, insertion of veins and bones as arterial walls. gOREK was optimized for interactive processing of the images to manage the geometry reconstruction. The end product is a geometric model of the arterial wall saved in PLOT3D or STL format and is compatible with different mesh generation softwares. We use Gridgen - a commercial mesh generator developed by Pointwise [6], which also allows editing and construction of geometric databases. Typical time for geometry reconstruction is 2 to 12 hours, depending on the image quality and the users experience. From the computational standpoint, the task can be performed on a standard

[1] the word OREK in Hebrew translates artery.

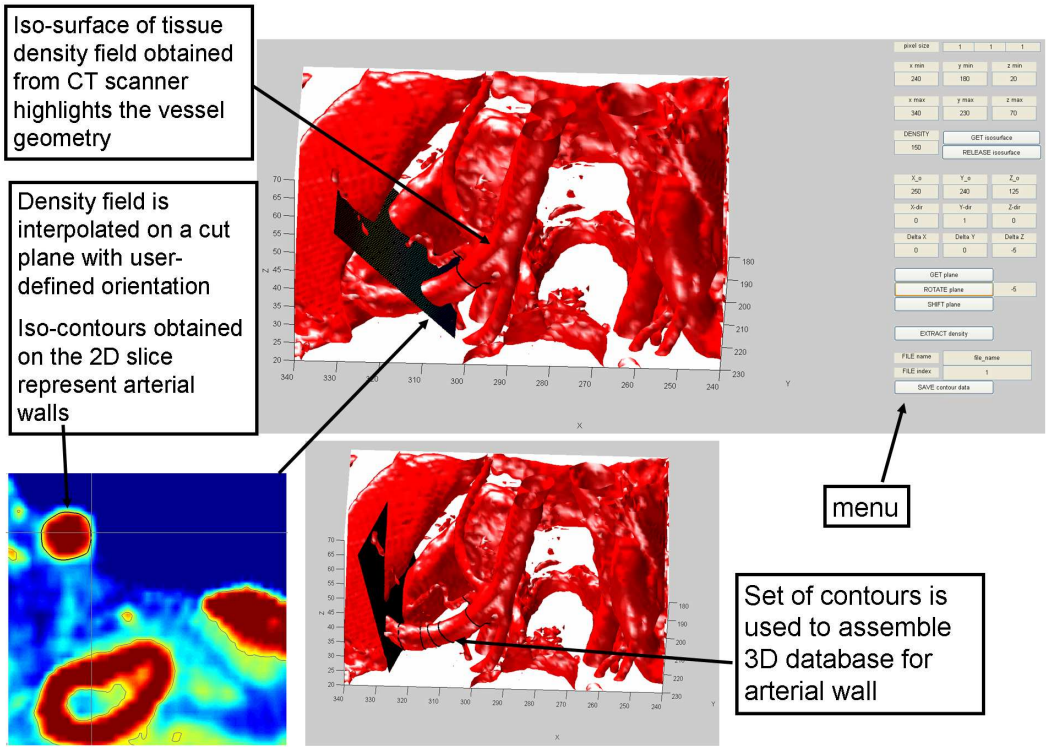


Figure 4.5: Reconstruction of arterial geometry with gOREK.

desktop computer with 2-4GB of RAM.

In figure 4.1(top right) we present an example of a cranial arterial model reconstructed from MRA and DSA CT images. The model has 65 arteries, 4 inlets and 31 outlets. Due to high complexity of the arterial network, which also includes vessels of different sizes, several numerical approaches to reconstruct the vessel wall geometries were integrated. The inner part of the arterial network (Circle of Willis, CoW) was reconstructed by extracting iso-surfaces from MRA data with predefined threshold. The left and right anterior cerebral arteries are located very close to each other and their iso-surfaces were merged, hence these arteries had to be digitally separated. The small vessels, such as posterior temporal, were reconstructed using another technique: initially, the arterial medial axis is computed and the vessel diameter along the medial axis is approximated. Then, ring-like contours are seeded along the medial axis to form a pipe-like structure. Once all parts of the arterial tree are extracted the data are uploaded into Gridgen, where additional editing is performed to integrate all parts into one smooth surface, eliminate remaining small features, and form the inlet and outlet regions of the arteries. As a final step, a finite element mesh was

generated. Example of arterial surface reconstructed with gOREK and edited in Gridgen is shown in figure 4.6.

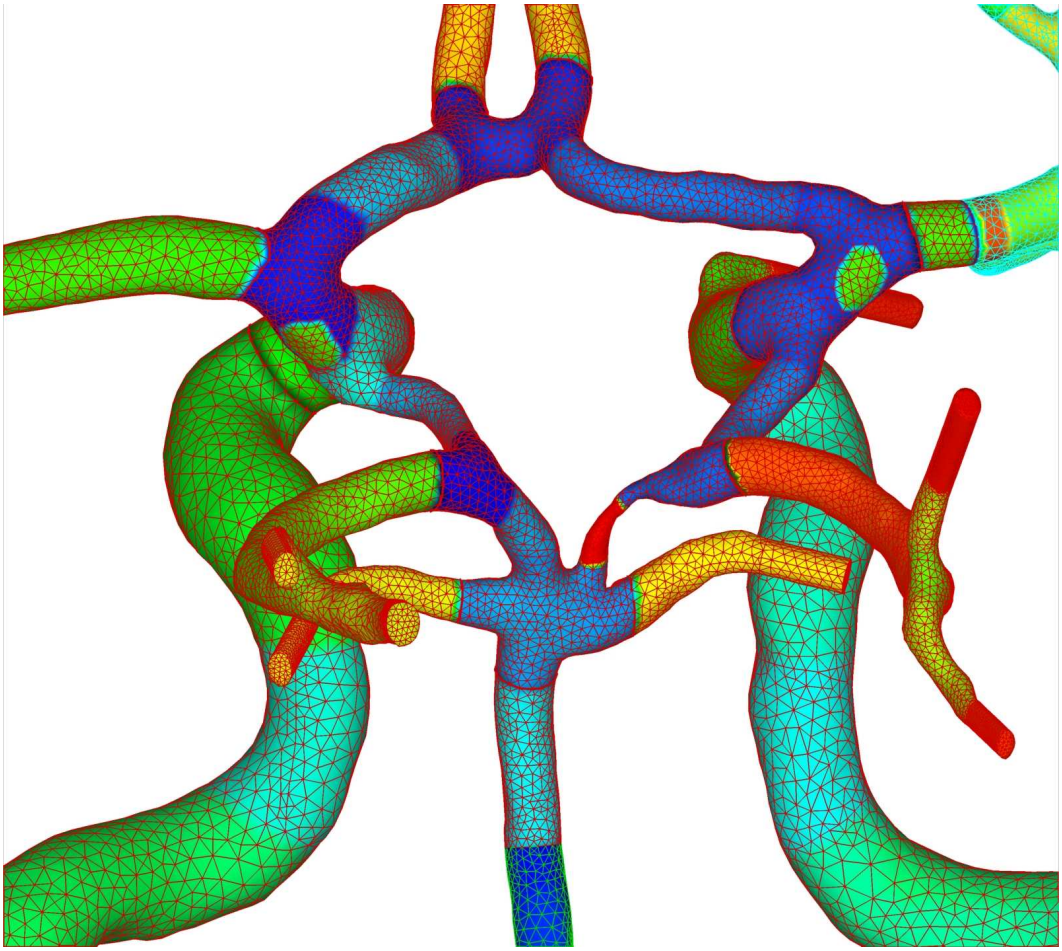


Figure 4.6: (in color) Reconstruction of Circle of Willis from MRI images. colors represent different patches of arterial surface reconstructed with gOREK and also created in Gridgen. Red and light blue triangles depict finite element surface mesh.

In figure 4.7(left) we plot the surface of a carotid artery bifurcation, reconstructed from a set of MRA images. The arterial wall is represented by several patches of a parametric surface. The surfaces corresponding to the arterial branches were generated by gOREK while the smaller patches were created in Gridgen.

Arterial geometry in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$: Spectral/hp element method employed in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ provides a dual path to convergence (i.e. decay of numerical error): (a) h-convergence, with the accuracy of the solution depending on the size of elements; and (b) p-convergence with the accuracy depending on the order of polynomial approximation and on the smoothness of the approximated solution. In the case of a smooth solution, exponential

rate of convergence is obtained. The unstructured surface mesh generated using any meshing technique is an ensemble of many flat triangular or quadrilateral elements. The sharp angles, created by the merging of the surface elements may affect adversely the smoothness of the approximated solution leading to poor p-convergence. To recover exponential convergence, the flat surfaces of boundary elements can be mapped on curved (smooth) arterial wall surfaces. This points to the importance of a consistent mesh generation procedure. The surface mesh must be generated on a smooth surface provided by a geometric database, extracted during image processing. Then, the same database must be used to map a flat elemental surfaces on a curved boundary [70, 80]. In reconstructed carotid artery, illustrated in figure 4.7(left), the parametric surfaces were used for mesh generation and subsequently for projection of faces of the boundary elements on the smooth surface. However, it was observed that if the original geometric database suffers from some degree of roughness then it cannot be used for projection of flat surfaces. In such cases, alternative methods for surface smoothing are available [98]. In our numerical algorithms we use a combination of techniques. The plot in figure 4.7(right) demonstrates a part of the geometric model of the cranial arterial network. The spectral element mesh was created using the smoothing technique described in [98].

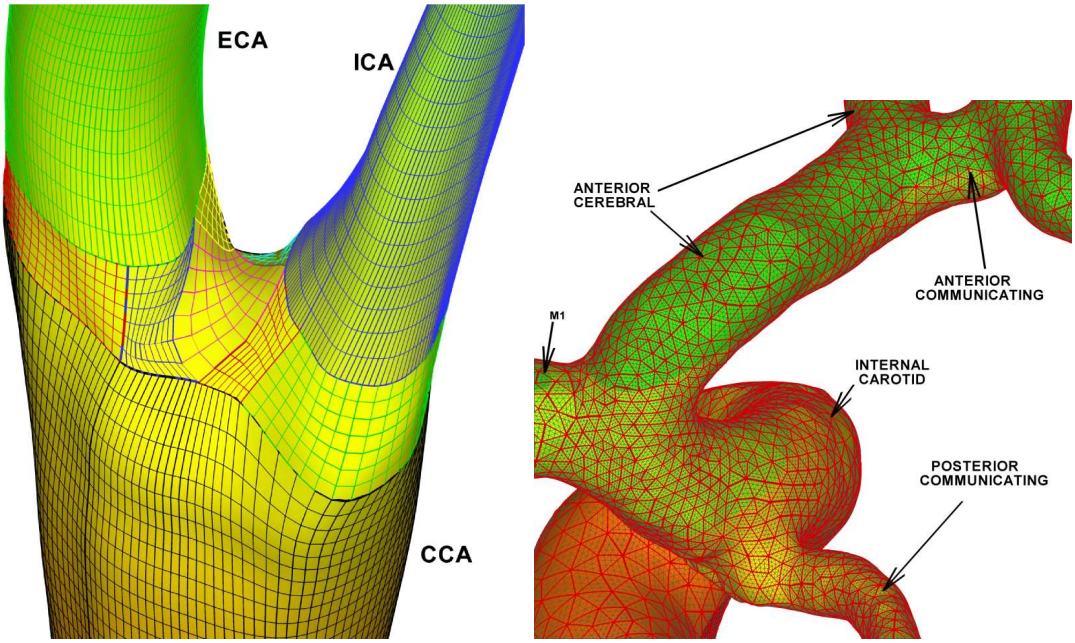


Figure 4.7: (in color) Left: Common, external and internal carotid arteries (CCA, ECA and ICA, respectively) reconstructed from magnetic resonance angiography (MRA) data. The arterial wall is represented by parametric surfaces. Right: Cranial arterial network. The geometrical model was reconstructed from MRA and digital subtraction angiography computed tomography images. Smoothing of the surface boundary was done with the technique described by Volino and Magnenat-Thalmann [98].

4.3 Outflow boundary conditions for arterial networks with multiple outlets

Geometrically-complex arterial networks involve many inlets and outlets, and consequently simple boundary conditions employed in a single arterial junction are not suitable for multiple junctions as they may lead to explosive instabilities, e.g., due to erroneous backflow induced at one or more of the outlets, the loss of mass conservation, etc. However, even if the outflow boundary conditions employed lead to stable simulations, the results are often very sensitive to the type of the boundary conditions employed. The arterial geometry affects the results but the effects are typically *local*, e.g., pockets of secondary flow, recirculation, etc. In contrast, inflow and outflow boundary conditions affect the *large scale features* of the flow, such as the flow rate ratio in junctions and the pressure distribution. Moreover, different inflow/outflow boundary conditions in domains with multiple outlets may lead to a variation of local features such as wall shear stress (WSS) even at regions located far away from the terminal vessels. Hence, whenever it is possible, in simulations of arterial flow in patient-specific geometries, corresponding *patient-specific* inlet and outlet boundary conditions should be applied as well.

An example of this sensitivity, in terms of the pressure distribution, is shown in the simulation results presented in figure 4.8; the two plots corresponding to *constant pressure* outflow boundary condition and the *RC-type* boundary conditions presented in this paper. The geometry consists of cranial arteries located in the left hemisphere of a human brain. The arterial network has twenty vessels, one inlet (Middle Cerebral artery) and ten outlets. The significant difference in pressure distribution even in the upstream vessels is clearly seen, and this leads to 300% difference in flow rates through the outlets. In table 4.1 we summarize the results of these two simulations.

outlet ID	1	2	3	4	5	6	7	8	9	10
$p_{out} = 0$: Δp	22	22	22	22	22	22	22	22	22	22
RC: Δp	44.4	54.4	42.5	7.1	42.6	44.7	27.9	33.2	29.9	18.8
$p_{out} = 0$: Q	4.8	0.9	0.8	19.3	2.8	2.9	1.7	1.1	3.4	6.6
RC: Q	9.5	2.9	1.6	6.1	5.3	5.3	2.2	1.8	5.4	5.1

Table 4.1: Simulation with RC and fixed pressure boundary conditions ($p_{out} = 0$): pressure drop between inlet and outlets and flow rates computed at each outlet. Data is presented in non-dimensional units.

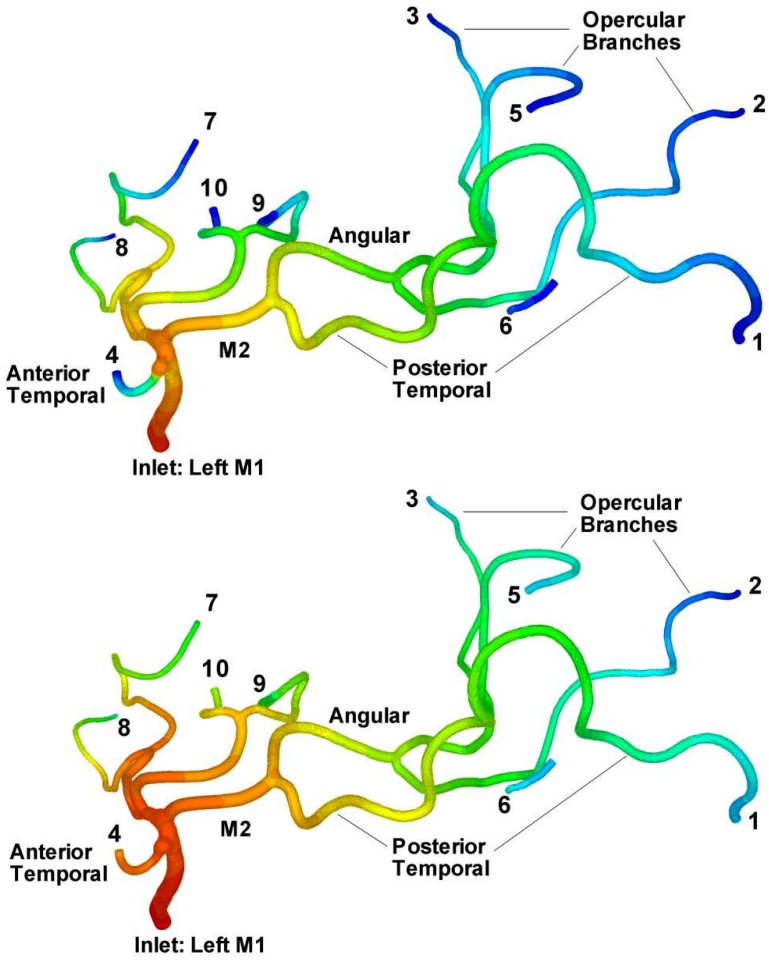


Figure 4.8: (in color) Steady flow simulation in a three-dimensional domain with 20 cranial arteries. Effect of different outflow boundary conditions. Upper - resistance boundary condition; Lower - constant pressure boundary condition ($p_{out} = 0$). Colors represent pressure distribution (red - high, blue - low). Numbers correspond to outlet IDs, consistent with table 4.1.

A similar result was obtained in simulating brain aneurysms for the inflow boundary condition [15], where it was found that for the fusiform aneurysm there was a 100% WSS variation by changing the effective *inflow profile* whereas for the saccular aneurysms the effect was not as pronounced.

The importance of outflow boundary conditions has been long recognized by the community and the large literature reflects that [97, 66, 81, 39, 84]. Some works have attempted to model the peripheral arterial network in order to derive more realistic pressure boundary conditions at the outflow. In the following we provide a brief overview of the four main approaches:

Constant pressure boundary conditions: In many 1D, 2D and 3D simulations the outflow boundary conditions at each outlet are provided by fixing the pressure. This type of boundary condition is reasonable for simulations of steady and unsteady flows in a domain with a single outlet but also for steady flow simulations in domains with multiple outlets, provided that the pressure at each outlet is known from measurements. Care must be taken to account for phase shift in pressure waves at the different outlets. In unsteady flow simulations in domains with multiple branches, fixing the pressure at outlets provides physiologically incorrect results. From the computational standpoint, prescribing constant pressure at each outlet is the least complicated.

Resistance boundary conditions: The Resistance boundary condition (R-BC) is based on the assumption of a linear dependence between the pressure and flow rate at each outlet. R-BC has been applied by many researchers for flow simulation in 1D domains with multiple outlets, see [97, 81, 39]; it can also be applied to 2D and 3D flow simulations. However, application of R-BC for a flow in rigid domains may lead to numerical instabilities since flow rate fluctuations at *all* frequencies are transferred to pressure oscillations. The resistance boundary condition is equivalent to constant pressure boundary condition for steady flow simulations. Pressure computed in steady flow simulation with the R-BC can be imposed in simulation with the fixed pressure boundary condition, and the results of the two simulations will be identical. In terms of computational complexity, R-BC is more expensive than the constant pressure BC, since the integral of velocity at each outlet must be computed at each time step. Nevertheless, it involves a perfectly parallelizable implementation, which requires only a subset of processors for computing the integral and performing blocking communication (blocking communication requires synchronization of processors, i.e., if one completes its task early it will still wait for other processors to finish their tasks, which degrades the performance of the numerical solver). [83]. Moreover, these processors can be grouped in disjoint groups - one per each outlet in order to overlap the computation of the integral and perform blocking communication within each group concurrently.

Windkessel model boundary conditions: There are several variations of the Windkessel model for boundary conditions. The most common is the *three element* Windkessel model, which is often denoted as RCR [39, 64, 42]. This type of BC can be applied to both steady and unsteady flow simulations. There are two major drawbacks for this method: (a) similarly to the R-BC, flow rate fluctuations at *all* frequencies are transferred to the pressure, and (b)

there are several parameters at each outlet that must be adjusted (typically two resistances and a capacitance, e.g. 30 parameters for the example of figure 4.8). The fitting of resistances and capacitances is usually done iteratively [84], thus simulations over several cycles are required, which increases the computational cost considerably. The selection of parameters is based on the values published in [87], but such values are clearly patient-specific. Detailed discussion on the RCR model is provided in section 4.3.1, where we compare it with the proposed RC method. From the computational standpoint, the integral of flow rate at each outlet must be computed at each time step.

Impedance boundary conditions: This type of boundary condition was investigated in [66, 65], and it involves an analytical approach for modeling the outflow boundary conditions. More recently, new results with the impedance boundary condition were presented in [97, 84, 85]. This method is based on approximating the arterial network as 1D tree-like structure, where the *linearized* flow equations can be solved analytically, time-periodicity of the flow is assumed; it can be applied to 1D, 2D or 3D flow simulations in domains with multiple outlets. Although accurate, convergence of the solution is achieved after several cardiac cycles, which makes the simulation expensive, particularly in the 3D case. For example, in simulation of unsteady flow in the cranial arterial tree (see figure 4.8) the solution converged to a periodic state with reasonable accuracy only after eight cycles. The computational cost of simulation of one cycle on 512 processors of CRAY XT3 was three hours. Using the impedance boundary condition, pressure at each outlet is computed from a convolution integral of the impedance and flow rate, which is taken over one cardiac cycle. At the beginning of simulation, values of flow rates are unknown and the convolution integral uses some "predicted" values. Accurate prediction of flow rates at outlets may improve the convergence rate. The convolution integral of the impedance and flow rate must be computed at each time step. The last operation is parallelizable, however it involves a blocking communication [83]. The flow rate for each outlet and at each time step must be also computed.

Alternative techniques to specify outflow boundary conditions in hemodynamics simulations can be found in the literature. One such method is based on a suitable variatonal formulation of the flow rate problem [46], the drawback of this problem is the introduction of non standard finite element subspaces. Another method is to impose prescribed flow rates at multiple outlets by means of Lagrangian multipliers [37, 96]. Additional technique is to

couple 3D simulation with reduced model (1D) simulation by means of interface boundary conditions [38, 40, 94]. In the 3D-1D approach the outflow pressure boundary condition for 3D solver is provided by 1D simulation. However, the outflow boundary conditions for the 1D simulations should still be imposed, possibly by one of the aforementioned methods. From computational point of view, these two techniques involve considerable computational effort compared to the aforementioned four approaches.

Our goal is to develop a new *scalable* and *efficient* type of pressure boundary condition applicable to vascular flow simulations in domains with multiple outlets. Our method is *not* a new model that attempts to mimic the peripheral resistance but rather a numerical procedure that allows to impose accurately and in a straight-forward manner measured *in-vivo* flow rates at terminal outlets. The method is valid for steady and unsteady flow simulation; moreover, periodicity of flow is not required. In summary, we base our choice of the boundary conditions on the following physiological and numerical considerations:

- *Accuracy*: One outcome is to obtain a model that has a high level of accuracy (in terms of patient-specific flow simulation) both with respect to geometry and inflow/outflow.

- *Simplicity*: Flow simulation in complex arterial networks may require imposing outflow boundary condition on tens or hundreds of terminal vessels.

- *Robustness*: Solution of Navier-Stokes equations in large domains, especially with high-order methods, is very expensive. Currently, a typical numerical simulation of flow in a modest size of vascular tree during *one* cardiac cycle may take from one to three days and requires hundreds or thousands of processors. Thus, the dependence of convergence of the numerical solution on the boundary conditions should be minimized.

- *Scalability*: It is essential that the outflow boundary conditions can be scaled, such that multilevel parallelism can be used. Three-dimensional simulation of a flow in arterial tree with tens or hundreds of vessels, junctions and multiple inlets and outlets requires use of hundreds or thousands processors. Several types of outflow boundary conditions involve additional computation and blocking communication, e.g., to compute flow rates. Multi-level parallelism allows blocking of subset of processors (instead of synchronizing thousands) [34].

- *Stability*: The boundary condition should not lead to spurious oscillations that render the simulation unstable.

With regards to spatial discretization, we employ the spectral/*hp* element code $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ validated in many biological flow studies [67]. The computational domain, used by $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$, consists of structured or unstructured grids or a combination of both. In each element the solution is approximated in terms of hierarchical mixed-order Jacobi polynomial expansions. In the current study, the computational domains are subdivided into tetrahedral elements and the solution is approximated with polynomials of order $P = 4$ and $P = 5$. A second-order splitting scheme was employed for temporal discretization [48].

The section is organized as follows: In section 4.3.1 we review suitable filtering for bioflow simulations. In section 4.3.2 we analyze Stokes flow in 1D arterial network and derive the relationship between the flow rates at the domain outlets and resistance. In section 4.3.3 we present results.

4.3.1 Filtering the high-frequency Oscillations

The relation of pressure and flow waves in arteries has been studied for centuries; a comprehensive review of the topic can be found in [64]. As illustrated in figure 4.9 the velocity and pressure waves in different arteries have different forms, however, there are some similarities in the pressure-flow relationship. From the plots in figure 4.9 we observe that the peak of the flow wave precedes the peak of the pressure wave. Also, the high frequency components of the flow wave are attenuated and are not reflected in the pressure wave. This last observation is key in the success of the three-element Windkessel (RCR) model to relate the pressure and the flow waves.

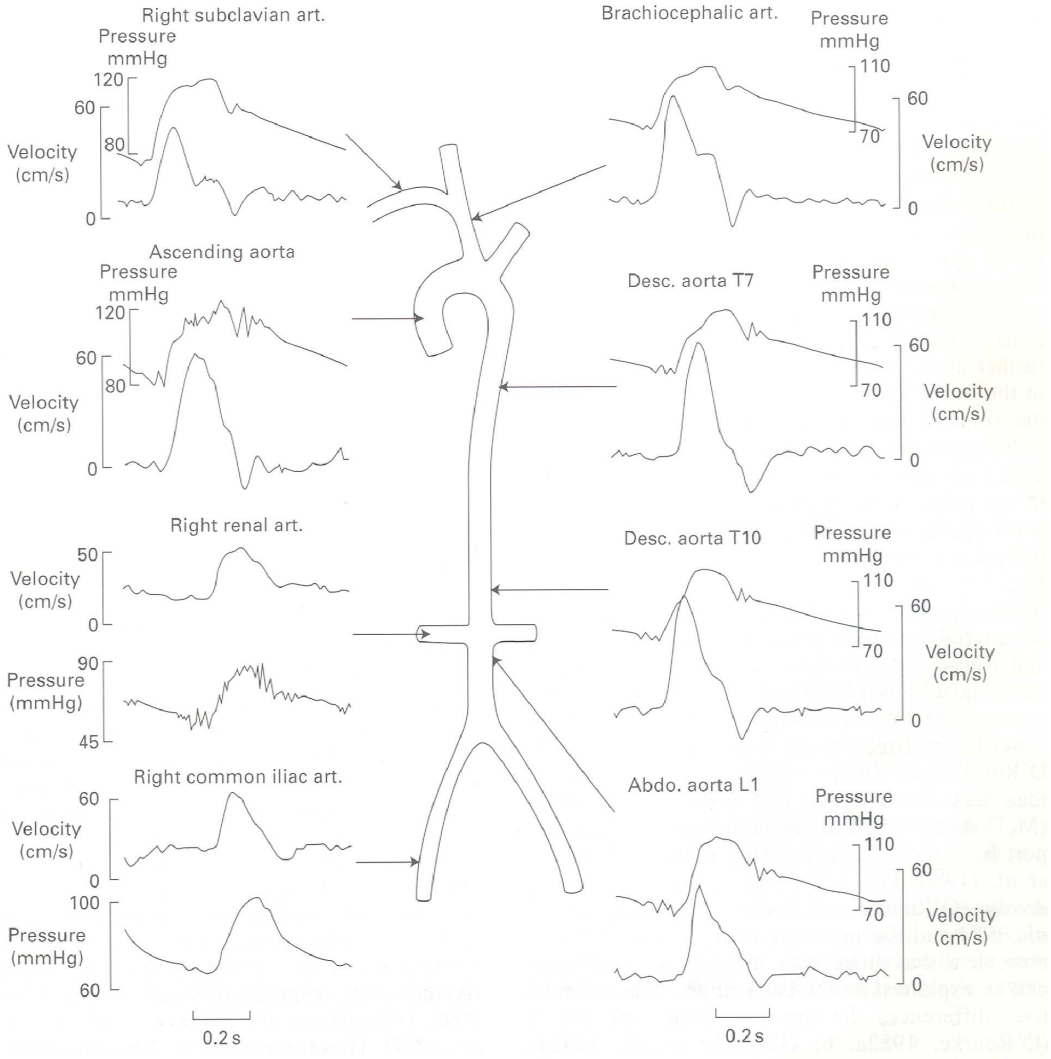


Figure 4.9: Pressure and flow waveforms in different regions of a human arterial system. Adopted from Mills *et al.* [61] and published in [64].

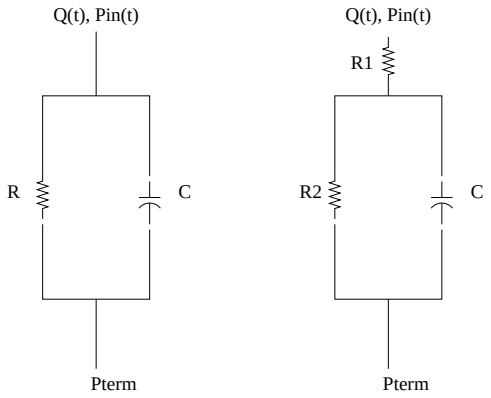


Figure 4.10: Left: Low-pass filter - RC circuit. Right: Three-element Windkessel model - RCR circuit.

In this section we compare the three element Windkessel (RCR) model and a simple low-pass filter, which we will refer to as the RC model. The two basic components of the low-pass filter are a resistor and a capacitor while in the RCR an additional resistor is added. The the RC and the RCR filters are shown in figure 4.10. In an electric circuit the value $(p_{in} - p_{term})$ is the difference of electrical potential between two points (the voltage potential) while Q is the electric current. In fluid dynamics the pressure p and the flow rate Q correspond to the voltage potential and current. In a low-pass filter, the relationship between electric potential and current (or correspondingly the pressure and flow rate) is given by

$$p_{in} - p_{term} + RC\frac{dp_{in}}{dt} = RQ, \tag{4.1}$$

where R and C are coefficients for the resistance and capacitance and $p_{term} = const$. Later, we will use formula (4.1) to define the RC-type of boundary condition for pressure. Next, we represent p_{in} and Q by a Fourier series:

$$p_{in}(t) = \sum_{k=0}^{\infty} \widehat{p}_k e^{i\omega_k t}, \tag{4.2a}$$

$$Q(t) = \sum_{k=0}^{\infty} \widehat{Q}_k e^{i\omega_k t} \tag{4.2b}$$

and we assume that $P_{term} = const$ to obtain:

$$\widehat{p}_0 = R\widehat{Q}_0 + p_{term} \tag{4.2c}$$

$$\widehat{p}_k = \widehat{Q}_k \frac{R}{1 + i\omega_k RC}, \quad k > 0. \tag{4.2d}$$

Equation (4.2c) suggests that for steady input we obtain a linear dependence between Q and P and defines the *Resistance* boundary condition. Equation (4.2d) implies that high frequency oscillations in flow rate, $Q(t)$, are attenuated and not reflected on the pressure $P_{in}(t)$, consistent with the aforementioned requirements.

In the Windkessel (RCR) model, the capacitance C reflects the elastic properties of a vessel, while R is the resistance of the downstream arterial network. As we will show below, the parameters of the RC models can be tuned such that one can obtain a good approximation to the RCR circuit for low frequency waves. That is, the resistance and capacitance in the RC model are related to those of the RCR model and have the same physiological meaning. However, in this study, we use the RC circuit as a *numerical model* and not as a physiological model, thus the values of R and C are based on numerical considerations only.

For a fixed resistance parameter R, the effectiveness of the filter depends on the capacitance C. Very low values of C will result in poor filtering of waves with low and intermediate wave number, whereas large values of C will result in very effective filtering. In electric circuits, the value $\omega_B = 1/(RC)$ is denoted as a "breakpoint" or the "half-power" point. For $\omega > \omega_B$, very fast decay in the amplitude of reflected wave is observed. The solution to equation (4.1) has two parts, a transient solution and a periodic state. Although, larger values of C are desirable for effective filtering, they prolong the transient state. The rate of decay of the transient solution $p_{in} \propto p(t=0)e^{-t/(RC)}$ depends on the value of RC; small RC leads to very fast convergence to periodic state. In section 4.3.3, we will show that during the initialization of the simulation high frequency oscillations appear, whose decay rate depends on C.

In the RCR circuit the relationship between the electric potential and the current is given by:

$$p_{in} + R_2 C \frac{dp_{in}}{dt} = (R_1 + R_2)Q + p_{trem} + R_1 R_2 C \frac{dQ}{dt}, \tag{4.3}$$

Fourier analysis of (4.3) gives:

$$\widehat{p}_0 = (R_1 + R_2)\widehat{Q}_0 + p_{term} \tag{4.4}$$

$$\widehat{p}_k = \widehat{Q}_k \frac{R_1 + R_2 + i\omega_k R_1 R_2 C}{1 + i\omega_k R_2 C} = \widehat{Q}_k \left[\frac{R_1 + R_2}{1 + i\omega_k R_2 C} + \frac{i\omega_k R_1 R_2 C}{1 + i\omega_k R_2 C} \right], \quad k > 0. \tag{4.5}$$

An important difference between the RC and RCR circuits is that the high frequency waves (e.g., any numerical noise) in flow rate are *not* filtered sufficiently by the RCR circuit for any values of R_1 and R_2. Clearly, when $R_1 \ll R_2$ the RCR and RC circuits perform similarly, except for the response at very high frequencies in Q. However, for relatively larger values of R_1 any random noise added to $Q(t)$ is transferred to $P_{in}(t)$ without significant attenuation.

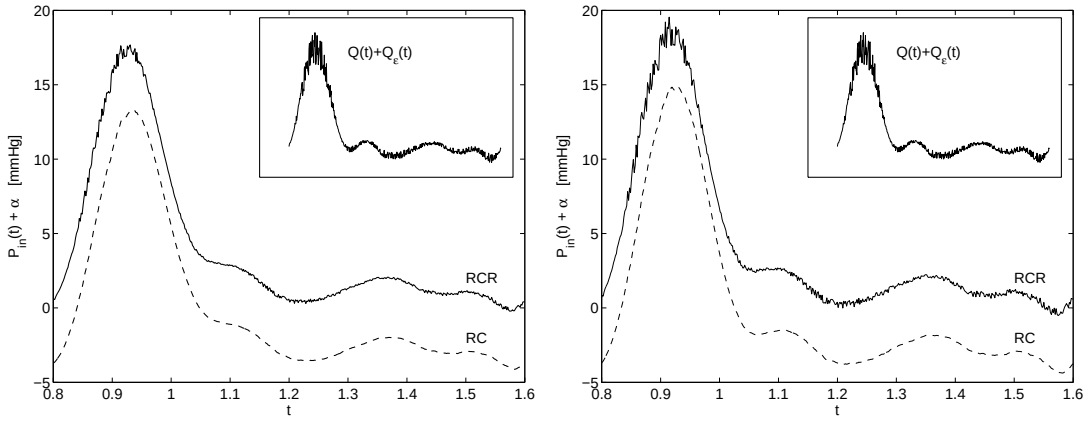


Figure 4.11: Pressure response to incoming flow wave $Q(t) + Q_\epsilon(t)$. Left: RCR model: $R_1 = 50$, $R_2 = 250$, $C = 0.05/(R_1 + R_2)$, $\alpha = 5$ and RC model: $R = R_1 + R_2 = 300$, $C = 0.05/R(R_2/R)$, $\alpha = 0$. Right: RCR model: $R_1 = 100$, $R_2 = 200$, $C = 0.05/(R_1 + R_2)$, $\alpha = 5$ and RC model: $R = R_1 + R_2 = 300$, $C = 0.05/R(0.8R_2/R)$, $\alpha = 0$.

In figure 4.11 we compare the pressure wave generated by the RC and RCR models in response to the same flow wave. The curve Q(t) represents typical flow rate in a carotid artery [89]. The mean value of $Q(t)$ was set to zero and the random perturbation $Q_\epsilon(t) = 0.2Q(t)\eta(t), \quad -1 \leq \eta \leq 1$, was added to the flow rate to highlight the filtering properties of the RC and RCR models. We note that the mean value of $p_{in}(t)$ was set to zero, which explains the negative pressure values in figure 4.11. The curve corresponding to the RCR model was shifted by $\alpha = 5$ for illustration purposes only to avoid overlapping between the RC and RCR curves. As expected, the random noise added to the imposed flow rate is filtered out when the RC model is applied but it is not filtered as effectively for the RCR model, particularly for higher R_1 values.

The relationship between pressure wave and flow wave, known as *impedance* $Z(\omega)$, is

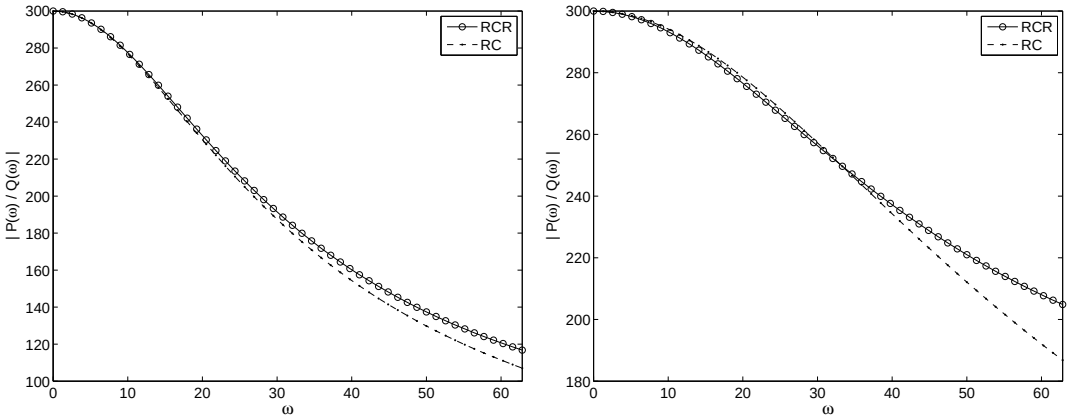


Figure 4.12: Magnitude of impedance versus mode number in the RC and RCR circuits. Left: RCR model: $R1 = 50$, $R2 = 250$, $C = 0.05/(R_1 + R_2)$ and RC model: $R = 300$, $C = 0.05/R(R2/R)$. Right: RCR model: $R1 = 100$, $R2 = 200$, $C = 0.05/(R_1 + R_2)$ and RC model: $R = 300$, $C = 0.05/R(R2/R)$.

defined in Fourier space by

$$Z(\omega) = \frac{p(\omega)}{Q(\omega)}.$$

Low values of impedance suggest that $P(\omega)$ is not sensitive to corresponding values of $Q(\omega)$. In the RC model, according to (4.2d), the impedance for high frequency waves vanishes, while in the RCR model it converges to a constant. In the numerical solution of flow equations, especially in large rigid domains with multiple outlets, high frequency oscillations in flow rate – that typically occur at different frequencies for different outlets – may lead to instabilities. Hence, from the numerical standpoint, filtering high frequency waves using the RC model enhances stability. In figure 4.12 we show the impedance corresponding to the RC and RCR models. Here, we use the same resistance and capacitance parameters as in figure 4.11. Note, that both models predict practically the same impedance for the low frequency modes typically considered in numerical studies of arterial flow. For high frequency the value of impedance predicted by the RC model will vanish to zero, while the value of impedance predicted by the RCR model according to formula (4.5) will converge to R_1.

4.3.2 Analysis of Stokes flow in simple networks

In this section we derive the relationship between the resistances, defined by the RC boundary condition, and the flow rates at outlets. We use two simplified models of arterial-like networks: the first model consists of one inlet and two outlets (Y-shaped bifurcation) while the second network of one inlet and three outlets. We show that the ratio of flow rates through terminal vessels can be approximated by the inverse of the ratio of the terminal resistances, namely

$$\frac{Q_i(t)}{Q_j(t)} \approx \frac{R_j}{R_i},$$

where Q_j and R_j are the flow rate measured at outlet j and the resistance parameter at the same outlet, respectively. This result is valid for any number of outlets as we will see in the simulations presented in section 4.3.3.

We can also extend the RC boundary condition to time-dependent cases by employing the $R(t)C$ boundary condition. Based on this extension, we can then impose in numerical simulation an experimentally measured $Q_i(t)/Q_j(t)$ ratio using the simple approximation

$$\frac{Q_i(t)}{Q_j(t)} \approx \frac{R_j(t)}{R_i(t)},$$

which is the same relation as before for the steady case.

4.3.2.1 Steady RC boundary condition

First, we consider Poiseuille flow in a single pipe of radius r and length L. The pressure - flow rate relationship (P-Q) is given by

$$\frac{dp}{dz} = \frac{p_{outlet} - p_{inlet}}{L} = KQ, \quad K = -\frac{8}{\pi R^4 Re}, \tag{4.6}$$

where Re is the Reynolds number.

Next, we consider a simplified Y-shaped arterial bifurcation with one inlet and two outlets, as illustrated in figure 4.13. As in a case of a single pipe, here we also assume that the flow is fully developed, z-independent and the pressure drops linearly in each segment. We denote the values of the pressure at the inlet, the bifurcation point and the two outlets, respectively, as p_{in}, p_b, p_1, and p_2, while the values of flow rates are denoted similarly. By

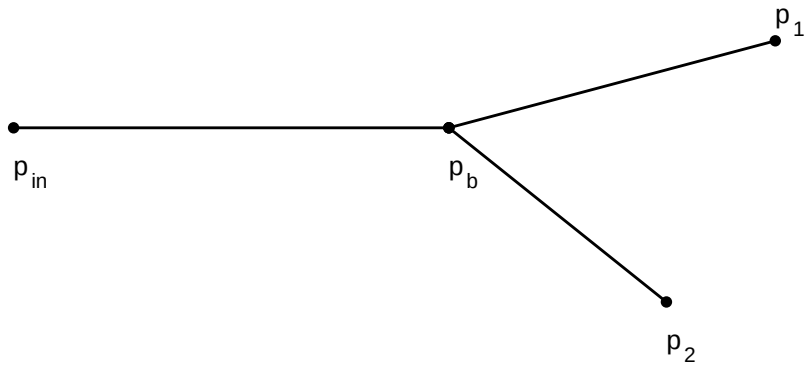


Figure 4.13: 1D model arterial bifurcation: one inlet and two outlets.

applying (4.6) to each of the three segments we obtain

$$p_b - p_{in} = L_{in}K_{in}Q_{in}$$

$$p_1 - p_b = L_1K_1Q_1$$

$$p_2 - p_b = L_2K_2Q_2$$

and from the last two equations we get:

$$p_2 - p_1 = L_2K_2Q_2 - L_1K_1Q_1.$$

Our goal is to choose appropriate pressure boundary conditions p_j in order to achieve the specified Q_j. Substituting the resistance boundary condition $p_j = R_jQ_j, \quad j = 1, 2$, (which is a particular case of the RC boundary condition in the steady regime) into the equation above, yields:

$$Q_2R_2 - Q_1R_1 = L_2K_2Q_2 - L_1K_1Q_1,$$

and thus

$$\frac{Q_1}{Q_2} = \frac{L_2K_2 - R_2}{L_1K_1 - R_1}, \tag{4.7}$$

here and thereafter we assume that $p_{term} = 0$. So, in order to achieve the desired flow rates in the numerical simulation we can set the parameters R_j according to (4.7).

Next we consider *pulsatile flow* in a 1D Y-shaped bifurcation. The bifurcation consists of pipe-like segments which are rigid and have constant diameter. The equation for the flow

rate is obtained by integrating the equation for the velocity $\mathbf{V} = (\mathbf{0}, \mathbf{0}, \mathbf{W}(\mathbf{t}, \mathbf{r}))$, which is assumed to be unidirectional:

$$\frac{\partial}{\partial t}\int_0^{D/2} Wrdr = -\frac{\partial}{\partial z}\int_0^{D/2} prdr + \frac{1}{Re}\int_0^{D/2} \frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial W}{\partial r}rdr.$$

The integral on the left hand side represents flow rate. The first integral on the right hand side is a force equivalent to $\bar{p}A$, where $\bar{p}$ is the average pressure and A is the cross-sectional area of the pipe-like segment. For simplicity of notation and without loss of generality, in the following the quantity $\bar{p}A$ is denoted by p. The second integral on the right hand side can be written in terms of the flow rate scaled by an appropriate parameter K, namely KQ. Then, in each segment the flow is approximated by:

$$\frac{\partial Q}{\partial t} = -\frac{\partial p}{\partial z} + KQ.$$

We note that the parameter K in this case is the same as in (4.6) under the assumption of parabolic flow profile only. In the case of a Womersley velocity profile, the parameter $K(t)$ will oscillate around some value K_m, which corresponds to the steady component of the Womersley flow. The amplitude of oscillation is bounded and depends on the coefficients of Womersley modes.

The 1D flow equations in the terminal segments are:

$$L_j\frac{\partial Q_j}{\partial t} = p_b - p_j + L_jK_jQ_j, \quad j = 1, 2,$$

here again we assume that the pressure depends linearly on z as in the case of Poiseuille or Womersly flow. Applying the last equation for two terminal segments ($j = 1$ and $j = 2$) we obtain:

$$L_1\frac{\partial Q_1}{\partial t} - L_2\frac{\partial Q_2}{\partial t} = p_2 - p_1 + L_1K_1Q_1 - L_2K_1Q_2.$$

The RC boundary condition reads

$$p_j = -R_jC_j\frac{\partial p_j}{\partial t} + R_jQ_j = -\alpha_j\frac{\partial p_j}{\partial t} + \beta_jQ_j.$$

Using the Fourier transform of Q and P and the formula for the RC boundary condition,

starting from:

$$L_1\frac{\partial Q_1}{\partial t} - L_2\frac{\partial Q_2}{\partial t} = -\alpha_2\frac{\partial p_2}{\partial t} + \alpha_1\frac{\partial p_1}{\partial t} + Q_1(L_1K_1 - \beta_1) - Q_2(L_2K_2 - \beta_2),$$

$$L_1\hat{Q}_{1,k} - L_2\hat{Q}_{2,k} = -\alpha_2\hat{p}_{2,k} + \alpha_1\hat{p}_{1,k} - \frac{i}{\omega_k}\hat{Q}_{1,k}\gamma_1 + \frac{i}{\omega_k}\hat{Q}_{2,k}\gamma_2, \quad \gamma_j = L_jK_j - R_j,$$

we obtain:

$$\frac{\hat{Q}_{1,k}}{\hat{Q}_{2,k}} = \frac{L_2 + \frac{i}{\omega_k}\gamma_2 - \frac{R_2R_2C_2}{1+i\omega_kR_2C_2}}{L_1 + \frac{i}{\omega_k}\gamma_1 - \frac{R_1R_1C_1}{1+i\omega_kR_1C_1}}. \tag{4.8}$$

Note that for $\omega = 0$ in equation (4.8) we recover equation (4.7). Setting $RC = \mathcal{O}(1)$ (our numerical experiments suggest that $C \leq 0.2/R$), $R \gg L$ and $R \gg LK$, we can approximate equation (4.8) as

$$\frac{\hat{Q}_{1,k}}{\hat{Q}_{2,k}} = \frac{R_2}{R_1} + \mathcal{O}(\epsilon) \quad if \quad R_j \gg L_j\omega_j, \tag{4.9a}$$

$$\frac{\hat{Q}_{1,k}}{\hat{Q}_{2,k}} = \frac{L_2}{L_1} + \mathcal{O}(\epsilon) \quad if \quad R_j \ll L_j\omega_j, \tag{4.9b}$$

which means that for sufficiently large values of R_j we can control effectively $\hat{Q}_{1,k}/\hat{Q}_{2,k}$ ratio for certain frequencies ω_k and, consequently, obtain required $Q_1(t)/Q_2(t)$. In practice, non-linear effects may lead to small deviations in $Q_1(t)/Q_2(t)$; we will investigate this point further in numerical simulations in section 4.3.3.3.

Let us now investigate the error in (4.9a) using a 3D model of the Y-shaped bifurcation shown in figure 4.14. The dimensions of the model are comparable to the size of common carotid artery, which bifurcates to internal and external carotid arteries. For simplicity, for now we consider steady flow with low Reynolds number $Re = 1$. The error in (4.9a) depends on the values of L_j, K_j and the effects of 3D geometry of the model. In order to minimize the error in (4.9a) we require $R_j \gg |L_jK_j|$. We recall that $K_j \propto \frac{1}{Re}$, thus the selection of $Re = 1$ is more restrictive in terms of choosing a value for R_j. In flows in arteries with radius of 2mm to 20mm the Reynolds number varies between 50 to 2000, the increased Reynolds number may be a result of physical exercise [90]. We should also note here that high Reynolds flows in arteries may result in transition to turbulence, e.g. in a case of stenosed carotid artery. In that case nonlinear effects, ignored in the derivation of the $Q - R$ relation, will influence the accuracy of approximation. In a study of a turbulent flow in a stenosed carotid artery, where the RC model was implemented, we observed a

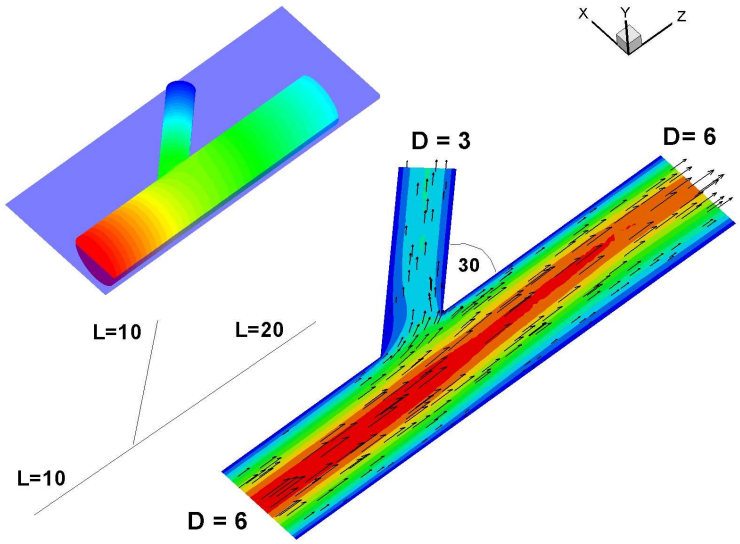


Figure 4.14: (in color) 3D model of Y-shaped bifurcation: Top left: 3D computational Domain intersected by a plane at y=0; colors represent pressure. Bottom right: Flow pattern at midplane; colors represent the velocity component in the flow direction. Bottom left: Corresponding 1D model; L and D denote the length and the diameter of each segment. The 3D domain is subdivided into 4026 spectral elements with polynomial order $P = 5$. The simulation was performed on 32 processors of CRAY XT3.

deviation of up to 5% in the ratio of flow rates through the two outlets with respect to imposed resistance ratio. Results of this study are presented in section 4.3.3.3.

In our simulation, according to 1D model, $L_1K_1 = -0.63$ and $L_2K_2 = -5.03$. The two limiting cases for the Q_1/Q_2 ratios are

1. $Q_1/Q_2 \to 8$, when $R_j \ll |L_jK_j|$, and

2. $Q_1/Q_2 \to 1.5$, when $R_j \gg |L_jK_j|$ and $R_2 = 1.5R_1$.

In table 4.2 we present results of 3D numerical simulations with different values of R_j. We observe that for sufficiently large R_j the approximation error in (4.8) vanishes and the pressure drop between the inlet and the two outlets converges to fixed values.

In the case of pulsatile flow, the approximation error ϵ in (4.8) should be analyzed for each wave number ω_k. In figure 4.15 we plot the $Q_1(\omega)/Q_2(\omega)$ ratio computed from equation (4.8) for $Re = [1,\ 10,\ 100]$. Here, the L_j parameters are the same as in the previous example and the parameters K_j are computed from equation (4.6). The $Q_1(\omega)/Q_2(\omega)$ ratio was computed for terminal resistances $R_1 = [10,\ 100,\ 1000]$ and $R_2 = [15,\ 150,\ 1500]$. As we can see in the three plots, the $Q_1(\omega)/Q_2(\omega)$ ratio remains constant for sufficiently large

R_1	R_2	Q_1/Q_2	error	$p_{in}-p_1$	$p_{in}-p_2$
0.0	0.0	10.40	593%	0.0151	0.0151
0.1	0.15	3.69	146%	0.0139	0.0271
1	1.5	1.80	20%	0.0125	0.0412
10	15	1.53	0.20%	0.0122	0.0446
100	150	1.50	0.00 %	0.0120	0.0450

Table 4.2: Steady flow in 3D domain with two outlets: $Re = 1$. Dependence of the Q_1/Q_2 ratio and ΔP on the terminal resistances R_1, R_2 in 3D simulation of steady flow in a bifurcation. The error is computed as $(Q_1/Q_2 - R_2/R_1)/(R_2/R_1)$ %.

values of R_1 and R_2.

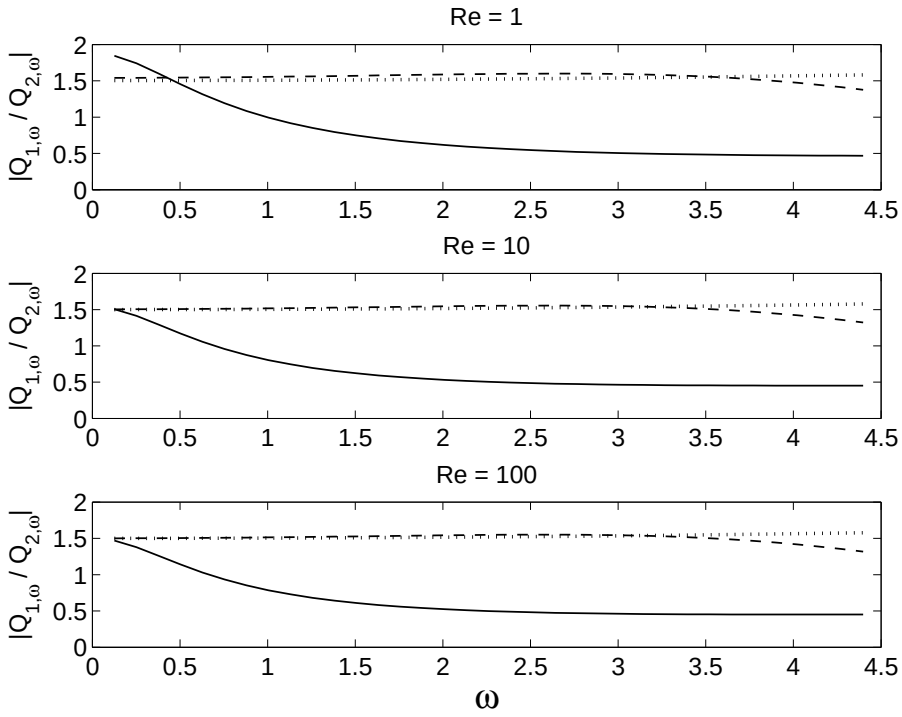


Figure 4.15: $Q_1(\omega)/Q_2(\omega)$ ratio computed using formula (4.8). Solid-line $R_1 = 10,\ R_2 = 15$; dash-line $R_1 = 100,\ R_2 = 150$; dot-line $R_1 = 1000,\ R_2 = 1500$, $C_j = 0.2R_j$.

In appendix C we derive the resistance - flow rate relationship in a **network of five vessels**. We show that by neglecting friction we can extend the $R - Q$ relation to the network with an arbitrary number of segments:

$$R_1Q_1 \approx R_2Q_2 \approx R_3Q_3... \approx R_jQ_j...$$

Stability of the RC boundary condition

In order to show that the RC boundary condition does not affect stability adversly we analyze a unidirectional ($\mathbf{V} = \mathbf{V}$(0,0,W(t,r))) and time-periodic flow in a pipe of constant radius. The pressure boundary condition at the outlet is provided by the RC model, and Dirichlet velocity boundary condition at the inlet and Neumann boundary condition at the outlet are imposed. We consider a spatial domain $\Omega = \{r, z : \ 0 \leq r \leq R, \ 0 \leq z \leq L\}$ with a fully developed flow described by

$$\frac{\partial W}{\partial t} = -\frac{\partial p}{\partial z} + \nu \frac{1}{r}\frac{\partial}{\partial r} r \frac{\partial W}{\partial r}. \tag{4.10}$$

Let us multiply (4.10) by W and integrate over Ω and over a time period T to obtain the kinetic energy $E(t)$:

$$E(T) = \frac{L}{2}\int_0^R (W^2|_t^{t+T}) r dr = \int_t^{t+T} Q(p|_{z=0} - p|_{z=L}) d\tau - L\int_t^{t+T} \beta^2 d\tau, \tag{4.11}$$

where

$$\beta^2 = \nu \int_0^R r \int_0^L \left(\frac{\partial W}{\partial r}\right)^2 dz dr.$$

We will analyze separately the two terms of (4.11):

- *First term*: $I_1 = \int_t^{t+T} Q(p|_{z=0} - p|_{z=L}) d\tau$.
- *Second term*: $I_2 = L\int_t^{t+T} \beta^2 d\tau$.

Our goal is to show that the energy $E(T)$ will *not* grow without bounds as time t increases. The term I_2 is not negative and bounded due to the Dirichlet velocity boundary condition and incompressibility $\nabla \cdot \mathbf{V} = 0$. Thus, in order to guarantee that the energy is bounded, we need to show that the term I_1, which depends on the boundary conditions, will not grow without bounds. First, we note that the flow rate Q is prescribed at the inlet of the domain by a *finite* periodic function and is bounded. Second, the values of the pressure amplitudes $\hat{p_k}$ are computed from the prescribed resistance R, capacitance C and the corresponding *finite* flow rate amplitudes $\hat{Q}_k$, and, according to (4.2c) and (4.2d), $|\hat{p_k}| \leq |R\hat{Q}_k|$ for any wave number ω_k. Also, since Q is a periodic in time, $p(z = L)$ is periodic, and consequently

$$\int_t^{t+T} Q p|_{z=L} d\tau \leq R Q_0^2 T,$$

and is not a function of t. The pressure at the domain inlet depends linearly on the pressure at the outlet, thus it is also bounded in terms of the mean flow rate and the length of computational domain. Hence, the RHS of (4.11) is not increasing with t, consequently $E(T) \leq E_0,\ E_0 > 0$.

4.3.2.2 Time-dependent R(t)C-type boundary condition

The *RC* boundary condition prescribes a constant ratio of flow rates at terminal vessels, however this is not the case in physiological flows, where $Q_i(t)/Q_j(t) = f(t)$. In this section we provide a procedure on imposing time-dependent $R(t)C$ boundary conditions in order to incorporate measured (at outlets) *in-vivo* or *in-vitro* flow rates.

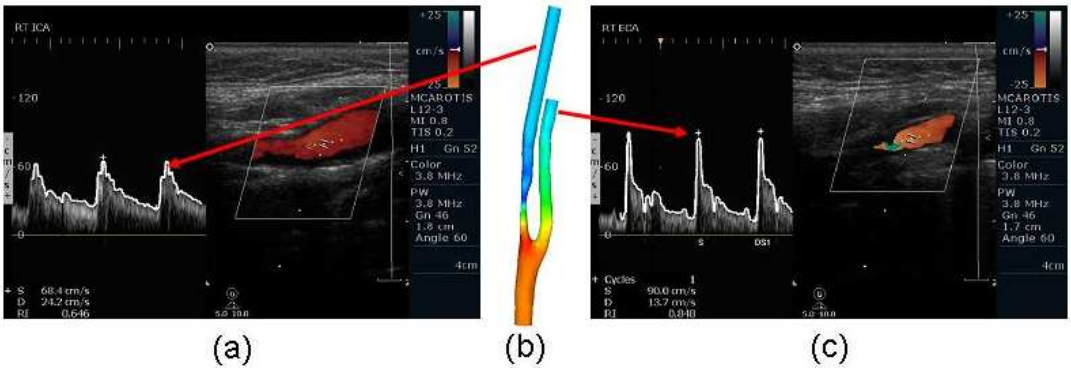


Figure 4.16: Human Carotid Artery - reconstruction from MRA images: Common Carotid Artery leads to Internal (left branch) and External (right branch) Carotid Arteries. Velocity profiles in ICA (a) and ECA (c), measured with Doppler Ultrasound technique and extracted from image (with Matlab), are marked by wide white curves.

Consider a simple Y-shaped bifurcation, such as the carotid artery, where the common carotid artery (CCA) is divided into external and internal carotid arteries (ECA and ICA), as shown in figure 4.16(b). It is possible to measure the flow rate at each of the arteries using non-invasive techniques such as Doppler Ultrasound or MRA. In figure 4.16(a,c) we present typical Ultrasound based measurements of velocity in the right ECA and ICA. The velocity is measured close to the center of the artery. From the velocity profile (marked by wide white curves) and the vessel diameter (D) measured during the same procedure, we can compute the flow rate through the two arteries. In general, measuring velocity at one point inside the vessel is not sufficient to estimate the flow rate. In the absence of additional measurements, the flow rate can be approximated as a linear function of velocity, scaled by the vessel's area. Alternative way for estimating the flow rate by introducing a relation between the flow

rate, the velocity and the Womersley number was proposed in [71]. MRA measurements provide a spatial distribution of the velocity, which leads to an accurate estimate of the flow rate. In figure 4.17 we plot the velocity waves and also the exact and approximate (with 25 Fourier modes) $V_{ECA}(t)/V_{ICA}(t)$ ratio extracted from the medical data. Assuming a linear flow rate-velocity relation $Q(t) = cV(t)0.25\pi D^2$, where c is a scaling coefficient, the $Q_{ECA}(t)/Q_{ICA}(t)$ ratio can be readily computed. Using physiological flow rates and fixing one of the resistance parameters to be constant, for example $R_{ECA} = const$, we can compute the required resistance at the second outlet, i.e.,

$$R_{ICA}(t) = R_{ECA}\frac{Q_{ECA}(t)}{Q_{ICA}(t)} = R_{ECA}f(t)$$

and assuming that the function $f(t)$ is periodic, we approximate it by a Fourier expansion

$$R_{ICA}(t) \approx R_{ECA}\left[A_0 + \sum_{k=1}^{M}\left(A_k cos(\omega_k t) + B_k sin(\omega_k t)\right)\right],$$

where A_0, A_k and B_k are coefficients of Fourier transform of a function $f(t)$. Alternatively, one can fix R_{ICA} and compute $R_{ECA}(t) = R_{ICA}\frac{Q_{ICA}(t)}{Q_{ECA}(t)}$.

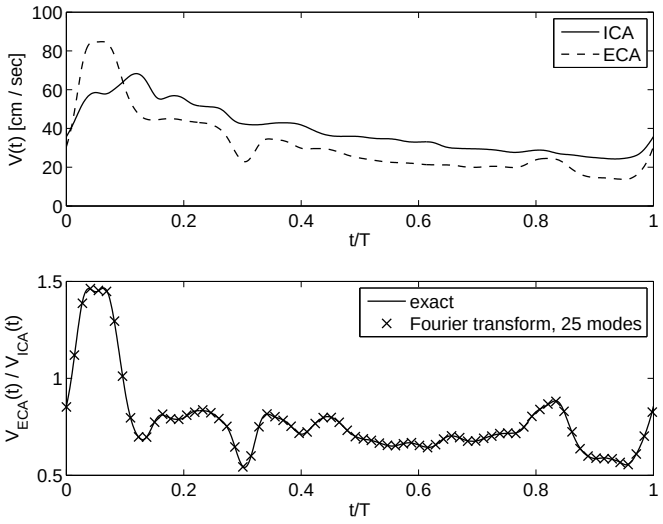


Figure 4.17: Velocity waves in carotid arteries measured with Doppler Ultrasound. Top: $V(t)$ in ECA and ICA. Bottom: Exact and approximated with 25 Fourier modes $V_{ECA}(t)/V_{ICA}(t)$.

4.3.2.3 Numerical implementation of RC and R(t)C boundary conditions

From the algorithmic standpoint, implementation of the RC boundary condition is straightforward and comparable to the Resistance boundary condition. Equation (4.1) can be discretized using, for example, the semi-implicit (first-order) scheme, i.e.,

$$p^{n+1} + RC\frac{p^{n+1} - p^n}{\Delta t} = RQ^n$$

but higher order semi-implicit time-stepping schemes can be also employed. The most computationally demanding part of imposing the RC-type boundary condition is the computation of the integral of velocity over each of the outlets, i.e.,

$$Q = \int \mathbf{V} \cdot \hat{\mathbf{n}} dA.$$

The element-wise domain decomposition, usually implemented in spectral/hp element (and also finite element) parallel solvers, results in assigning different processors to elements with faces on different outlets. Thus, the integral of velocity at each outlet is naturally computed in parallel. We can also map processors which contain mesh elements facing different outlets to a set of disjoint communicators, using the $MPI\_Comm\_Split()$ command. Note, that elements from different outlets can belong to the same partition and in this situation additional care must be taken in splitting the communicator. If individual partitions do not contain elements at multiple outlets then splitting is accomplished in just one call to $MPI\_Comm\_Split()$:

```
if (partition_contains_outlet == TRUE)
  color = outlet_ID;
else
  color = MPI_UNDEFINED;
MPI_Comm_split ( MPI_COMM_WORLD,     \\ communicator we want to split
                 color,              \\ outlet ID or MPI_UNDEFINED
                 rank_in_comm_world, \\
                 &comm_out);         \\ new communicator
if (color == MPI_UNDEFINED)
   comm_out = MPI_COMM_NULL;
```

If a particular partition (or several partitions) contains elements from more then one outlets, then several new communicators are formed:

```
MPI_COMM *comm_out;
comm_out = new MPI_COMM[Number_of_outlets];
for (outlet = 0; outlet < Number_of_outlets; ++outlet){

  if (partition_contains_outlet[outlet] == TRUE)
    color = 1;
  else
    color = MPI_UNDEFINED;
  MPI_Comm_split ( MPI_COMM_WORLD,     \\ communicator we want to split
                   color,              \\ outlet ID or MPI_UNDEFINED
                   rank_in_comm_world, \\
                   &comm_out[outlet]); \\ new communicator for outlet
  if (color == MPI_UNDEFINED)
     comm_out = MPI_COMM_NULL;
}
```

As a result of communicator splitting, the blocking $MPI\_Allreduce$ function, required to sum contributions from different processors, is executed over different sub-groups of processors. Processors which do not have elements facing the outlets will not participate in communication at all. A case when one or more processors hold elements from two or more outlets and consequently belong to two or more communicators usually happens when the number of processors is small (for example if number of processors is less then the number of outlets). However, in most situations it is very unlikely that one partition contains elements from multiple outlets, particularly when the number of processors is large, and partitioner software (like METIS, [3]) provides additional optimization generating contiguous sub-domains.

The memory requirements for RC boundary condition are negligible but an additional relatively small work is required to apply time-dependent $R(t)C$ boundary conditions. In order to avoid keeping in memory values of $R_j(t)$ at each outlet for every time step we can approximate the measured $R_j(t)$ with a smooth function $f_j(t)$ and perform interpolation

at every time step t^n. There is no requirement for time periodicity. In the case of periodic $Q_i(t)/Q_j(t)$, we can use a Fourier transform, so we need to store in memory only M modes and the mean value A_0 for each outlet. In the case of non periodic $Q_i(t)/Q_j(t)$, we can use spline or high-order polynomial approximation.

4.3.3 Simulation results

In the results presented in this section we used 3D models to test the RC and $R(t)C$ boundary conditions. Specifically, for the first two tests the geometrical models were constructed using cylindrical pipes attached together as illustrated in figure 4.18. The axis of all pipes belong to the same plane $y = 0$. The diameters of the three pipes are $D_1 = 6$, $D_2 = 3$ and $D_3 = 2$, here the subscripts denote the outlet index. Our goal is to verify numerically our analytical result, i.e.,

$$\frac{Q_i(t)}{Q_j(t)} \approx \frac{R_j}{R_i}.$$

4.3.3.1 Prototype case

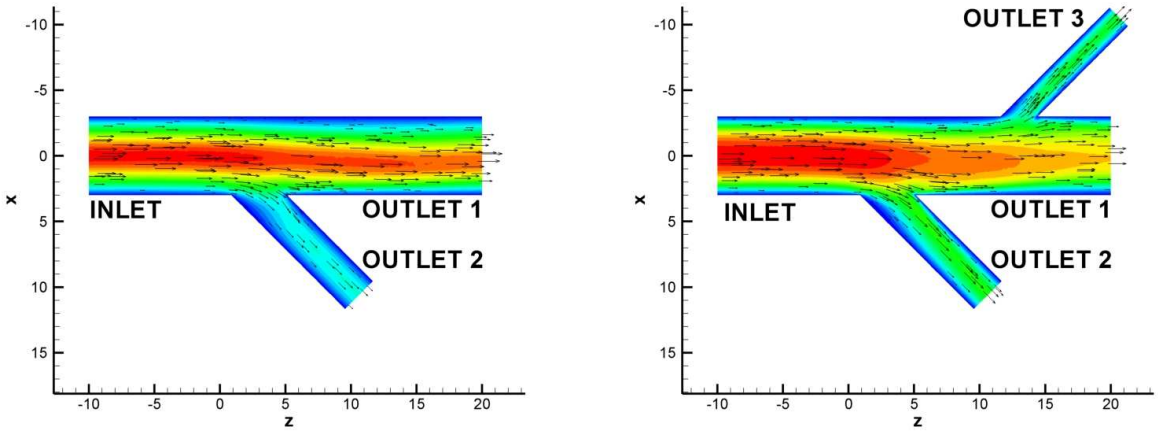


Figure 4.18: (in color) 3D computational domains with two and three outlets. They consist of pipe-like branches with diameters $D_1 = 6$, $D_2 = 3$ and $D_3 = 2$. Colors represent instantaneous w-velocity. The number of spectral elements in the domain with two outlets is 4026 while in the domain with three outlets is 5238; the solution is approximated with polynomial order $p = 5$. Simulations were performed on 32 processors of CRAY XT3.

In the first test we use a computational domain with two outlets (figure 4.18 left). First, we perform unsteady flow simulation with inlet boundary condition prescribed by two Womersley modes: the first mode corresponds to the mean flow while the second mode corresponds to $\omega = 2\pi/T$, where T is the nondimensional time period; here the Reynolds and the Womersley numbers are $Re_m = \frac{U_m D_{inlet}}{\nu} = 200$ and $Ws = 0.5 D_{inlet} \left(\frac{\omega}{\nu}\right)^{1/2} = 4$, where U_m is a mean velocity at the inlet and ν is a kinematic viscosity. The results are

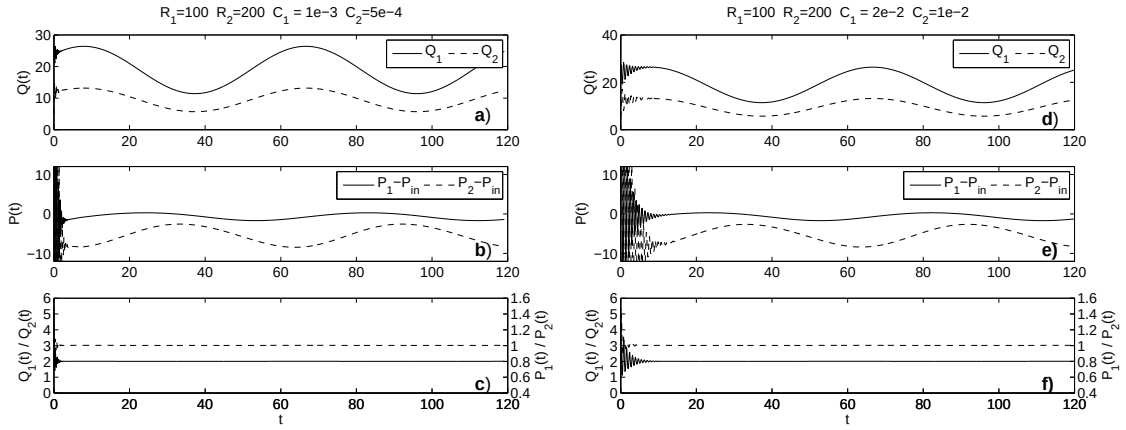


Figure 4.19: Numerical solution of 3D unsteady flow in a domain with two outlets using the RC boundary condition. Inlet boundary condition are specified with two Womersley modes. (a) and (d) show flow rates at outlet 1 (Q_1) and outlet 2 (Q_2); (b) and (e) show pressure differences; (c) and (f) flow rate (solid line, left Y-axis) and pressure (dash line, right Y-axis) ratios, Results in (a), (b) and (c) were computed with parameters $C_1 = 1e-3,\ C_2 = 5e-4$ while results in (d) (e) and (f) were computed with parameters $C_1 = 2e-2,\ C_2 = 1e-2$.

shown in figure 4.19 and confirm that the flow rate ratio satisfies $Q_1(t)/Q_2(t) = R_2/R_1 = 2$, the specified resistance ratio. The difference between the first two simulations (see figure 4.19) is in the C_j parameters. In particular, we observe prolonged oscillations in the flow rate and the corresponding pressure waveform during the transient period in the second case, where the C_j parameter is increased by a factor of 20.

In figure 4.20 we present results from a simulation where the inlet boundary condition for velocity is prescribed by five Womersley modes; the first mode corresponds to the mean flow and the rest are for the unsteady components. Note that in this case the $Q_1(t)/Q_2(t) = R_2/R_1 = 2$ is also satisfied.

In the next examples we consider simulations with time-dependent $R(t)C$ boundary conditions. The 3D computation domain for this test has three outlets (see figure 4.18 right). We use a time-dependent resistance $R = R(t)$ in order to impose the *desired* flow rate at all outlets. First, the solution with the impedance boundary conditions was computed. Second, a Fourier transform was applied to the monitored $Q_1(t)/Q_2(t)$ from the last cycle. Having the Fourier coefficients for the flow ratio, we then compute $R_2(t) = R_1 Q_1(t)/Q_2(t)$ and $R_3(t) = R_1 Q_1(t)/Q_3(t)$, where the reference resistance $R_1 > 0$ is a constant; in this simulation we used $R_1 = 100$. Alternatively, one may fix R_2 or R_3 and then compute the

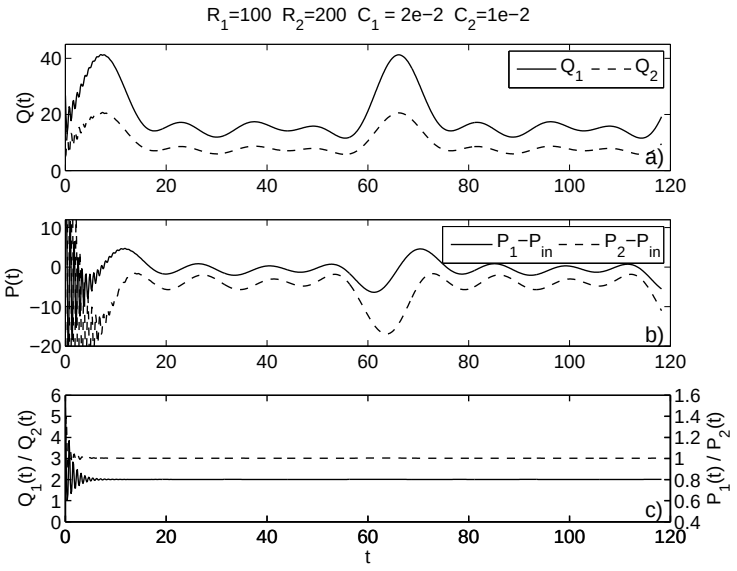


Figure 4.20: Simulation of 3D unsteady flow in a domain with two outlets using the RC boundary condition. The inlet boundary condition is specified with five Womersley modes. (a) flow rates at outlet 1 (Q_1) and outlet 2 (Q_2); (b) pressure differences; (c) flow rate (solid line, left Y-axis) and pressure (dash line).

values of $R_1(t)$ and $R_3(t)$ or $R_1(t)$ and $R_2(t)$. The choice of R is important, and according to our model we require that $R_j \gg L_j K_j$. We note that one can also use a time-dependent value for the reference resistance. In our simulations this was not necessary, however, if $Q_i(t)/Q_j(t)$ is extremely large, then adjusting the value of R_i such that $R_i(t)Q_i(t)/Q_j(t)$ will be in certain range, may be considered. In figures 4.21a and 4.21b we show the reference and the approximate (using 20 Fourier modes) $Q_1(t)/Q_2(t)$ and $Q_1(t)/Q_3(t)$ ratios. By "reference" data we denote the Q_j computed with the impedance boundary condition. In figure 4.21c we compare the results of the simulation. We observe very good agreement between the reference solution and the numerical solution computed with the $R(t)C$ boundary condition. We also note that a very short transient period develops for the convergence of $Q_j(t)$. The oscillations in pressure and flow rate values captured in figures 4.19 and 4.20 are due to two factors: a) The initial velocity field does not satisfy the continuity equation $\nabla \cdot \mathbf{V} = 0$, and b) the transient solution of equation (4.1) decays slowly when the value of RC is large. The incompressibility state is achieved after first few time steps. During that period large pressure fluctuations are observed at the inlet of the computational domain and flow rate fluctuations are observed at the outlets. This situation is typical for simu-

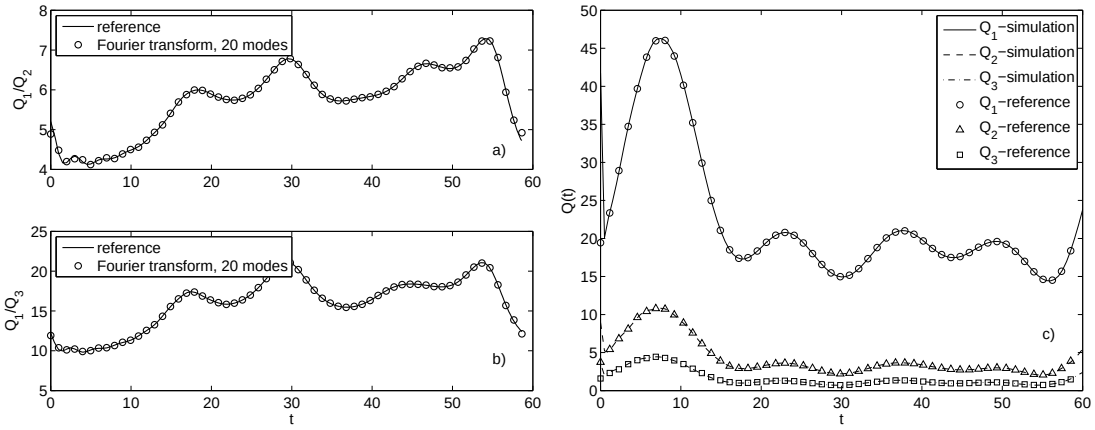


Figure 4.21: Numerical solution of 3D unsteady flow in a domain with three outlets using $R(t)C$ boundary condition. The inlet boundary condition is specified with five Womersley modes. Left top and bottom: Reference solution and solution approximated with 20 Fourier modes for $Q_1(t)/Q_j(t),\ j = 2, 3$. Right: Similar plot but for $Q_j(t)$.

lation with any type of boundary condition (including fixed pressure boundary condition) when the incompressibility constraint of the initial velocity field is violated. When RC or $R(t)C$-type boundary conditions are employed, flow rate oscillations are transfered into the pressure at outlets and consequently to the entire domain. These oscillations can be considered as numerical noise. The advantage of the RC and $R(t)C$ boundary condition is in attenuation of high frequency oscillations.

4.3.3.2 Cranial arterial tree

In the next example, we consider a computational domain containing 20 cranial arteries reconstructed from high resolution CTA images (see top plot of figure 4.8). The arterial network has 10 outlets and one inlet. In table 4.3 we provide details of this large-scale simulation on 512 Cray XT3 processors.

Number of elements	111,214
Polynomial order p	5
Δt	0.0025
Number of processors	512
Average cpu-time per time step	0.35 sec
Wall-clock time per one cycle	3 hours

Table 4.3: Flow simulation in domain of 20 cranial arteries. Computational details.

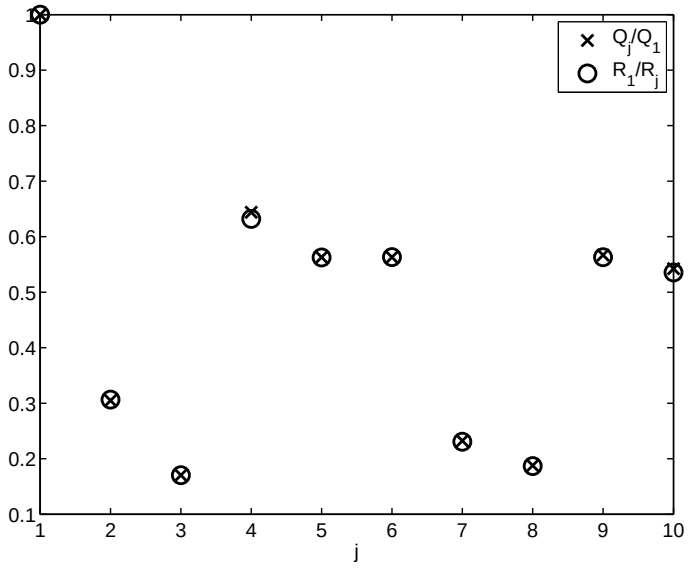


Figure 4.22: *Steady* flow simulation in a domain of 20 cranial arteries with RC boundary condition. Comparison between Q_j/Q_1 and R_1/R_j ratios. $Re = 200, Ws = 4.95$.

First, we simulate steady flow. In figure 4.22 we compare the ratio Q_j/Q_1 to R_1/R_j ratios for $j = 1, 2, ...10$. We observe very small deviation between the data. Second, we consider *time-dependent flow*. Initially, we performed a simulation with the impedance boundary condition [66], and reasonably good convergence of flow rates at all ten outlets was achieved after 6 to 8 cardiac cycles. In figure 4.23(left) we show the difference between flow rates over two successive time periods. In figure 4.23(right) we plot two convergence rates of Q_j at outlet with the fastest convergence ($j = 3$) and with the slowest convergence ($j = 4$). The curves for the convergence rates of Q_j at other outlets are located between the curves plotted for Q_3 and Q_4. The convergence of flow rate was estimated from a difference in flow rate at two subsequent cycles:

$$L_2(i) = \sum_{k=0}^{M-1} \frac{\sqrt{(Q(t_k + (i+1)T) - Q(t_k + iT))^2}}{M},$$

where M is a number of time steps in each cycle, i is the cycle index and T is the nondimensional time period; here time $0 \leq t_k < T,\ t_k = k\Delta t$ is measured from the beginning of a cycle.

Finally, we considered the recorded flow rate history at each outlet as a reference so-

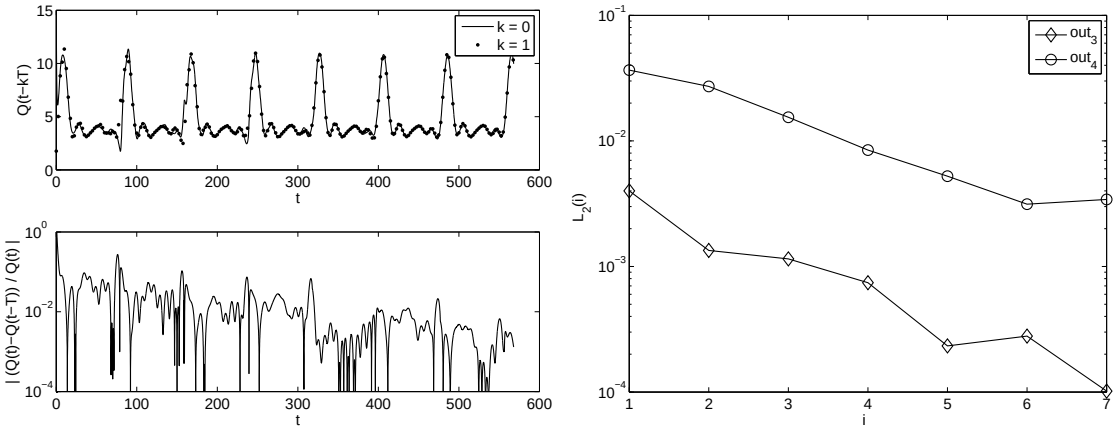


Figure 4.23: Unsteady flow simulation in a domain of 20 cranial arteries, convergence of flow rates. Left: pointwise difference in a flow rate at outlet No. 3. Convergence in the difference $|Q(t)-Q(t-T)|$ indicates convergence to periodic state. Right: convergence rate at outlets No. 3 (fastest convergence) and No. 4 (slowest convergence). $Re = 200, Ws = 4.95$.

lution and performed the simulation with the $R(t)C$ boundary condition. In figure 4.24 we compare the reference flow rates and the flow rates computed with the time-dependent $R(t)C$ boundary conditions during the first cycle. The initial condition for velocity in the last simulation was $\mathbf{V} = \mathbf{0}$, and as we can see from the data presented in figure 4.24 a very fast convergence of flow rates through all 10 outlets is achieved.

In order to verify that velocity fields computed with the $R(t)C$ and with the impedance boundary conditions are the same, instantaneous velocity fields from two simulations were compared. In figure 4.25 we show a section of an artery with a *saccular aneurysm* where the velocity fields were extracted. In figure 4.26 we plot the three components of the velocity extracted at $y = 185$. We observe that the velocity fields computed in simulations with different boundary conditions are practically the same.

The computational cost of simulation with impedance boundary condition was 27 hours on 512 processors of CRAY XT3. The computational *saving* in the simulation with the $R(t)C$ boundary conditions compared to the simulation with impedance boundary condition is more then 20 hours on 512 processors of CRAY XT3.

There is a fundamental difference between the $R(t)C$ and the impedance boundary condition. In *patient-specific* simulations with the $R(t)C$ method, measurement of flow rates only are required, while the impedance method requires measurements of both the

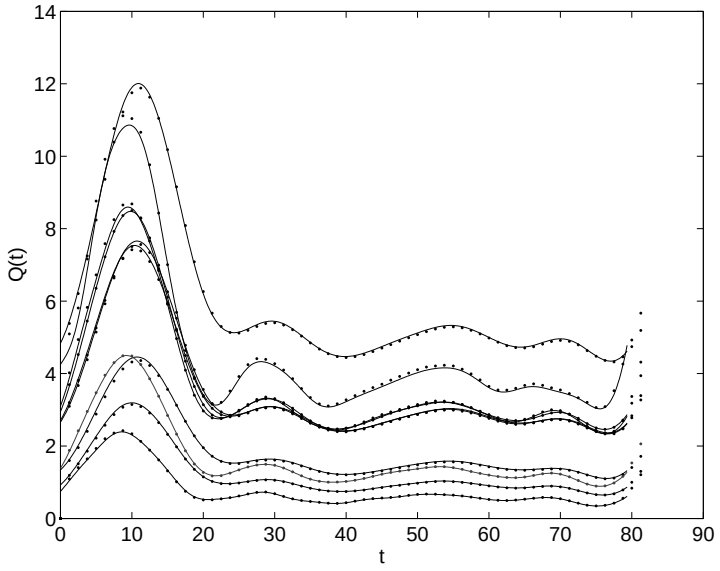


Figure 4.24: *Unsteady* flow simulation in a domain of 20 cranial arteries with two types of boundary conditions. Comparison of the reference and computed flow rate at 10 outlets. Solid line - reference solution, obtained with the Impedance boundary conditions; dot line - results of numerical simulation with corresponding $R(t)C$-type boundary conditions. $Re = 200, Ws = 4.95$.

flow rates and the pressure waves at each outlet in order to compute the patient-specific impedance. While it is reasonable to obtain the flow rates using existing medical equipment in *non-invasive* measurements, it is practically impossible to measure pressure in most of the arteries, since it requires complicated *invasive* procedures. Thus, in order to achieve *patient-specific* flow rates in simulations of a flow with multiple outlets using the impedance boundary condition, a costly iterative optimization over several parameters for each outlet is required. Moreover, at each iteration a simulation of several cardiac cycles must be completed before a reasonable convergence is achieved.

4.3.3.3 Stenosed carotid artery

In the next example we simulate unsteady flow in a stenosed carotid artery. The computational domain used for the study was reconstructed from MRA images and discretized into 19270 tetrahedral spectral elements, with the solution approximated with eighth-order polynomial expansion. In figure 4.27 we show the geometry of the model, flow patterns and time history of velocity at selected points. Due to the relatively high Reynolds number and

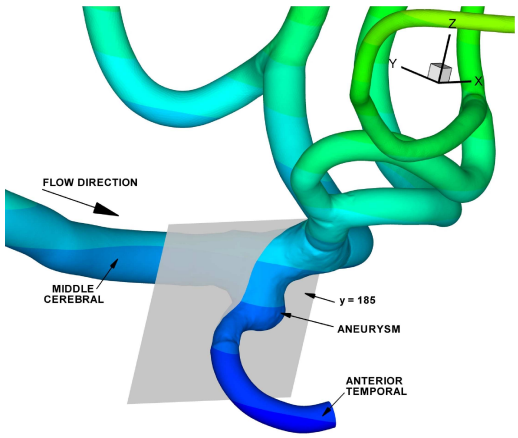


Figure 4.25: Cranial arterial tree geometry. Plane $y = 185$ intersects a small saccular aneurysm developed at the Middle Cerebral and Anterior Temporal (U-shaped) arteries. The velocity field in this section is characterized by strong secondary flows. Colors correspond to the z-coordinate.

complexity of the geometry, the flow develops an intermittent laminar-turbulent-laminar regime. A detailed study of the flow in stenosed carotid artery will be published elsewhere, so here we focus only on the boundary conditions imposed at the outlets. At the inlet we imposed Womersley velocity profile with 20 modes. The main characteristics of the flow at the inlet are: Reynolds number based on mean velocity and inlet diameter is $MEAN(Re_{inlet}) = 350$, peak Reynolds number based on inlet average (in space) velocity at the systolic peak (at $t = 0.15$, see figure 4.28) $MAX(Re_{inlet}) = 1345$, $Ws = 4.375$. The time period for one cycle is $T = 0.9sec$. Due to stenosis, the internal carotid artery has a 77% oclusion (based on cross-sectional area). The peak Reynolds number at the narrow part of the internal carotid artery is $MAX(Re_{stenosis}) = 395$. At the outlets of the internal (outlet 1) and external (outlet 2) carotid arteries we imposed the RC boundary condition, with $R_1 = 100$, $C_1 = 0.002$ and $R_2 = 60$, $C_2 = 0.005$.

In figure 4.28 we plot the computed flow rates and the Q_1/Q_2 ratio. The error in imposing $Q_1/Q_2 = R_2/R_1 = 0.6$ is less then 5%. The error reaches its maximum when the instantaneous Reynolds number is very low ($MIN(Re_{inlet}) = 46$) and at the onset of the turbulent regime. In the next simulation we imposed the *time dependent* resistance boundary condition, $R(t)C$. The value of the resistance at outlet 1 (ICA) was fixed $R_1 = 100$, while the value of the resistance at outlet 2 (ECA) was computed using prescribed $Q_1(t)/Q_2(t)$ flow

rate ratio. Results for two cases of $Q_1(t)/Q_2(t)$ ratio are presented in figure 4.29, very good agreement between the computed and reference flow rates is observed.

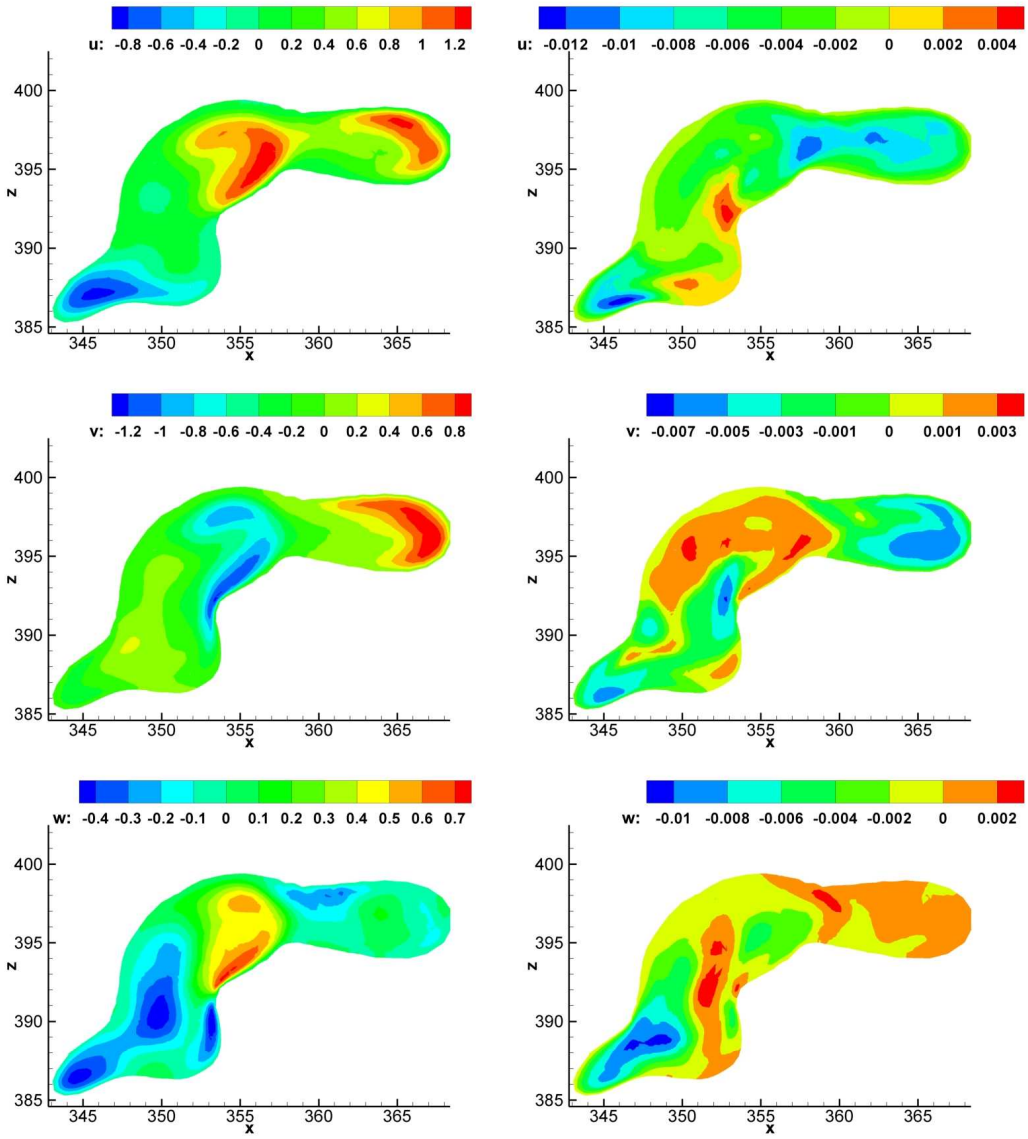


Figure 4.26: (in color) Unsteady flow simulation in a domain of 20 cranial arteries. Comparison of velocity fields at slice $y = 185$ depicted in figure 4.25. Left plots: solution with the Impedance boundary condition. Right plots: difference between velocity fields computed with the Impedance boundary condition and the corresponding $R(t)C$ boundary condition.

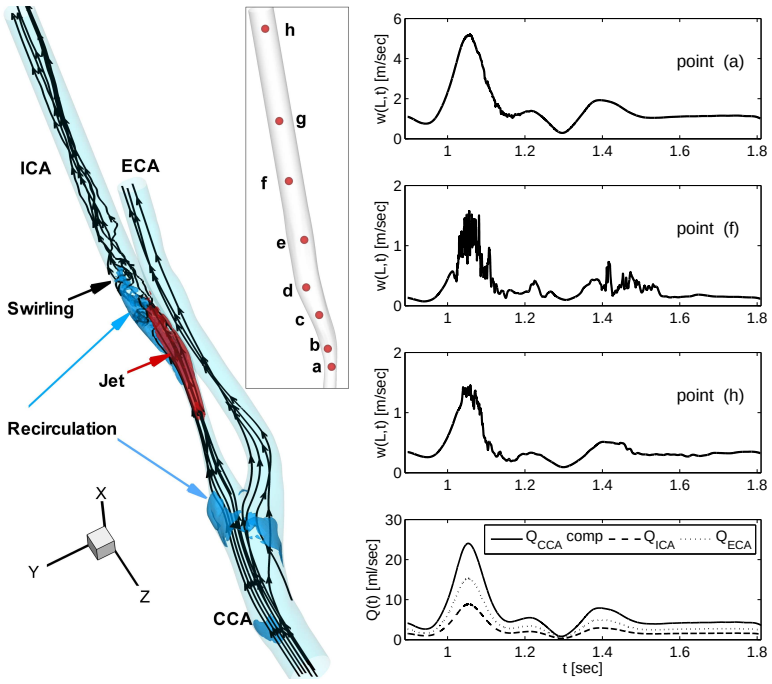


Figure 4.27: Simulation of intermittent laminar-turbulent-laminar flow in a stenosed carotid artery with RC-type boundary condition. Left: flow patterns: red iso-surface depicts high speed flow region (jet), blue iso-surfaces show regions of backflow; instantaneous stream lines demonstrate swirling flow. Right: flow rates and negative z-component of the velocity field monitored at the selected history points (in ICA); the locations of the points are marked by the red dots. $MEAN(Re_{inlet}) = 350$, $MIN(Re_{inlet}) = 46$, $MAX(Re_{inlet}) = 1345$, $Ws = 4.375$.

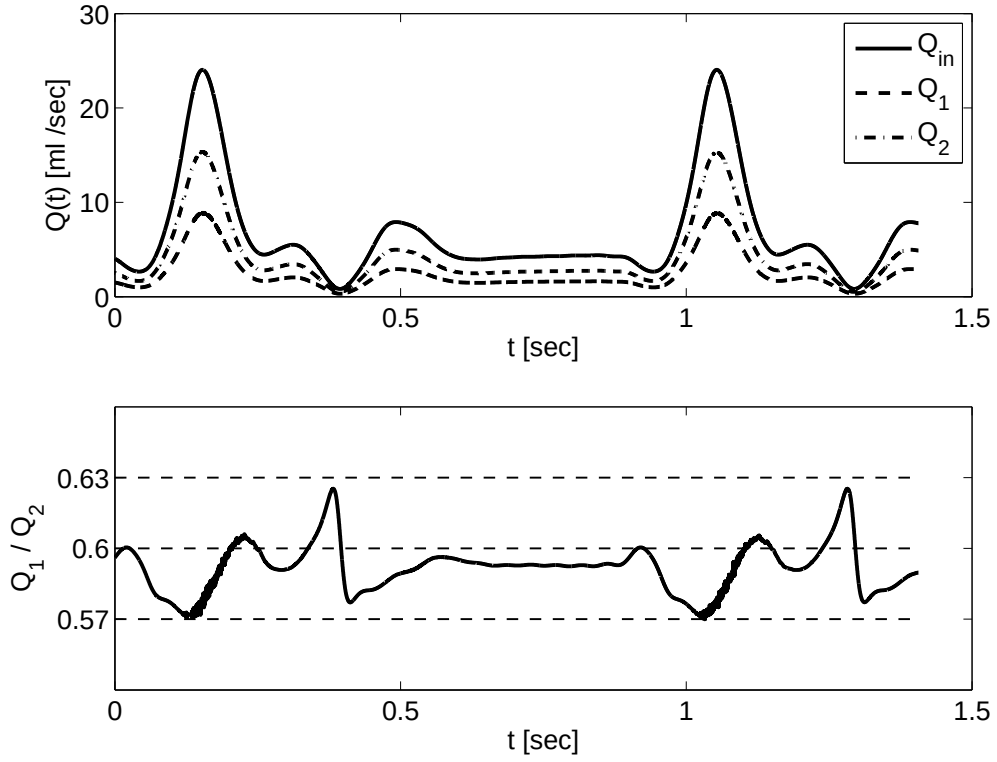


Figure 4.28: Simulation of intermittent laminar-turbulent-laminar flow in stenosed carotid artery with RC-type boundary condition. The prescribed by RC-type boundary condition flow rate ration $Q_1/Q_2 = R_2/R_1 = 0.6$ is obtained within 5% error. $R_1 = 100$, $R_2 = 60$, $C_1 = 0.002$, $C_2 = 0.005$. $MEAN(Re_{inlet}) = 350$, $MIN(Re_{inlet}) = 46$, $MAX(Re_{inlet}) = 1345$, $Ws = 4.375$.

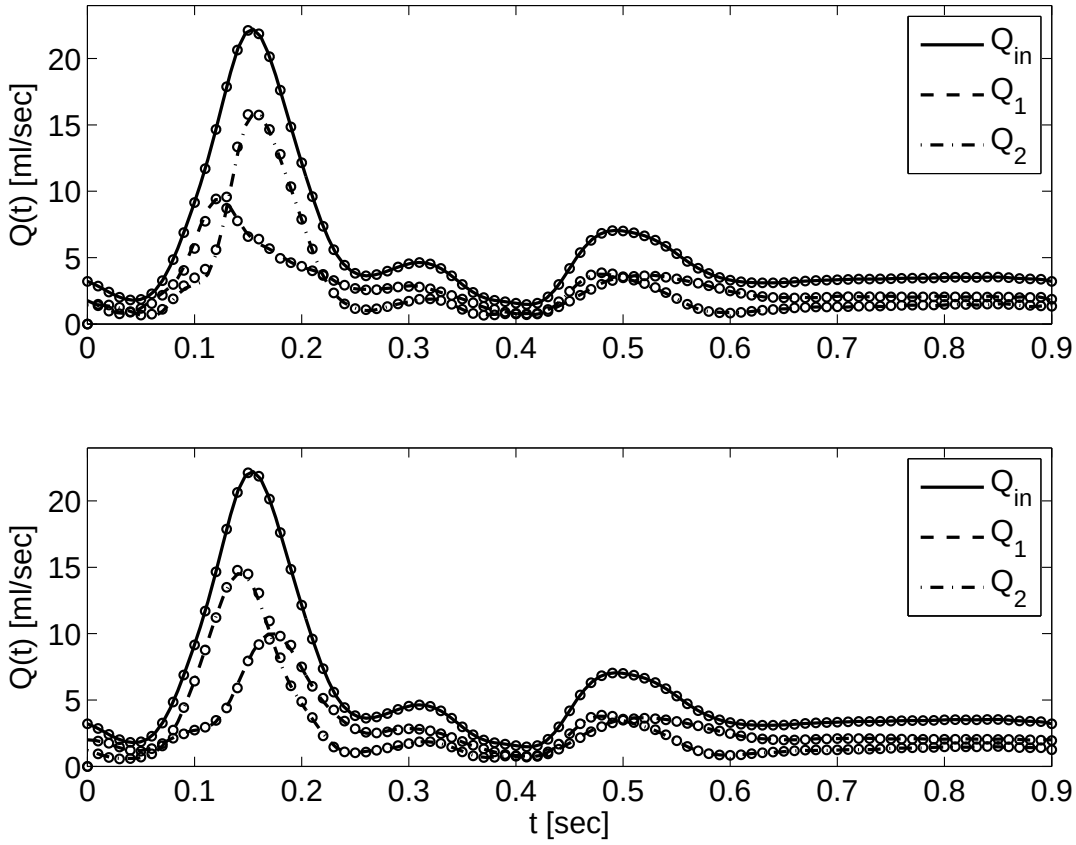


Figure 4.29: Simulation of intermittent laminar-turbulent-laminar flow in stenosed carotid artery with $R(t)C$-type boundary condition. Prescribed values of the flow rates are marked with lines; computed values of the flow rates are marked with dots. $R_1 = 100$, $R_2(t) = R_1(Q_1(t)/Q_2(t))$, $C_1 = 0.002$, $C_2 = 0.005$. $MEAN(Re_{inlet}) = 350$, $MIN(Re_{inlet}) = 46$, $MAX(Re_{inlet}) = 1345$, $Ws = 4.375$.

4.4 Two level domain partitioning

Advances in computer architecture present new challenges in developing parallel numerical algorithms. For example, many of the existing today parallel solvers show lower scalability on the new multi-core based supercomputers, when compared with the scalability achieved on the previous generation of a single-core computers. The number of communication paths between cores of different compute-nodes is significantly lower then the core count. Hence, the performance is adversly affected by the network contention when several messages are competing over a single path. To emphasize the need for new parallel paradigms we compare the performance of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ on CRAY XT3 (with one dual-core processor per node) and CRAY XT5 (with two quad-core processors per node) computers. On CRAY XT3 we used less optimized version of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$, i.e., without implementing the acceleration techniques presented section 3.5, while on CRAY XT5 the most optimized solver with lower iteration count and consequently lower volume of communication[2] has been employed. The results are presented in figure 4.30. We observe almost two-fold reduction in the computational time on XT5 due to implementation of the fast numerical algorithms; and at the same time the reduction in the *scalability* is clearly seen.

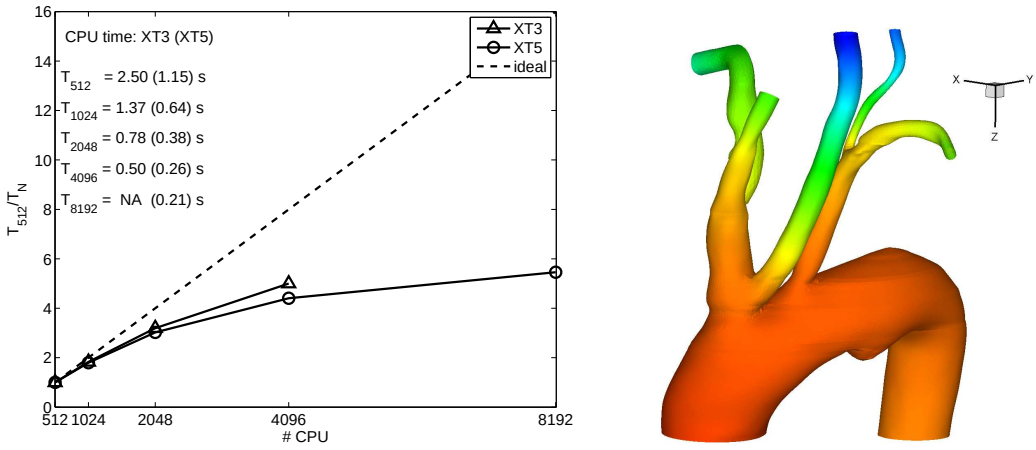


Figure 4.30: Performance of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ on CRAY XT3 and CRAY XT5: Left: Parallel speed-up. Problem size: 120,813 tetrahedral elements, $P = 10$, (226,161,936 quadrature points); preconditioner: LEBP. Right: geometry of the computational domain; color corresponds to pressure values. Computation performed on CRAY XT3 (ERDC) and CRAY XT3 (PSC, for 4096 processors), and CRAY XT5 (NICS). CRAY XT3 has one dual-core processor per node, CRAY XT5 has two quad-core processors per node.

[2]It was verified that scalability of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ increases due to acceleration techniques reducing the iteration count.

Hierarchical patterns of communication between processors and different speed of communications should be taken into account in design of new parallel algorithms. For example the architecture of a half petaflop Sun Constellation Linux Cluster (Ranger) located in Texas, Austin has several levels of hierarchy: four quad-core processors are composing a blade (a node), 12 nodes connected by local network create a "chassis", four "chassis" are contained in each of 82 "racks". The communication between cores located within the same blade is about ten times faster than the communication between cores from different nodes. Similarly, communication between processors within the same "shassi" is faster than between processors belonging to different racks. One of the challenges in designing scalable parallel algorithms is minimizing the slow communications by *topology aware domain decomposition* and "lamping" the communication intensive part of the code to processors connected with the fastest network. Another challenge is minimizing blocking collective operations.

In 3D large-scale simulations, the number of spectral elements can be well over a million, and due to the high-order polynomial expansion the number of DOFs may be over several billions. This large number of unknowns leads to construction of a global linear operator matrix with very high rank and consequently with very large condition number. Decoupling of the interior degrees of freedom by applying Schur decomposition leads to reduction in the size of the linear operator that must be inverted, however, the rank of the Schur complement is still very large.

In figures 3.17 and 4.30 we plot the performance of $\mathcal{Nek}\mathcal{T}\boldsymbol{ar}$ on the CRAY XT3 and XT5 for a simulation involving 120,813 elements. The scaling is favorable for high-order polynomial approximation, however there is a clear saturation in performance of the code on large number of CPUs. One of the bottle neck in scaling iterative solvers is scalability of efficient preconditioners. In section 3.4 we have discussed an implementation of parallel LEBP and the coarse space linear vertex solver. The coarse space linear vertex solver is implemented in two steps: In the first step, the global operator $\mathbf{V_{SC}}$ is constructed from the linear (vertex) modes, which are shared by different partitions, is constructed and inverted in parallel. In the second step, the local operator constructed from linear modes within each partition is inverted. The size of $\mathbf{V_{SC}}$ is increasing with the number of partitions, and this imposes two problems: (1) Inversion of a full matrix with large rank. For example, in a domain of 120,813 tetrahedral elements the rank of $\mathbf{V_{SC}}$ for the pressure solver is 12,496

when 256 processes are used and almost 25,066 when the domain is sub-divided (using METIS [3]) onto 8192 partitions. The rank of $\mathbf{V_{SC}}$ grows almost linearly with an increase in the number of processes. (2) The volume of computation performed by each process decreases relatively to the volume of communication between processes required by parallel matrix-vector multiply; hence the parallel efficiency of the preconditioner degrades.

To overcome the aforementioned problems, we propose to decompose the arterial network into a series of *weakly coupled* sub-domains or patches of *manageable* size for which high parallel efficiency can be achieved on a sub-cluster of processors. The continuity of numerical solution across different patches is achieved by providing appropriate interface conditions, based on a *hybrid* $C^0 - DG$-like approach. The method preserves the advantages of C^0 discretization, namely low number of DOFs compared to a full DG discretization and implicit treatment of the viscous terms of Navier-Stokes equation within a patch. The entire simulation is performed in two levels: (i) in the inner level, we solve conqurently a series of tightly coupled problems in each sub-domain using semi-implicit time stepping scheme and C^0 polynomial approximation, and (ii) in the outer level, we explicitly solve the weakly coupled problems by imposing interface boundary conditions on the sub-domains interfaces with a DG-like approach. In this Chapter we use $\mathcal{NEKTARG}$ - a numerical software employing the two-level domain decomposition method and Multi Level Communicating interface.

4.4.1 Formulation

The numerical approach is based on a two-level domain decomposition (2DD), where the large computational domain is initially decomposed into several overlapping or non-overlapping patches (coarse level) and SEM domain decomposition (1DD) is applied within each patch. In the illustration of figure 4.31 the computational domain Ω is subdivided into two *non-overlapping* patches Ω_A and Ω_B, with the thick dash black line depicting the location of the patch interface denoted by Γ. At every time step the tightly coupled problem defined within each patch is solved using the semi-implicit numerical scheme(2.34a). The boundary conditions at the interface are provided by exchanging values of the numerical solution across Γ.

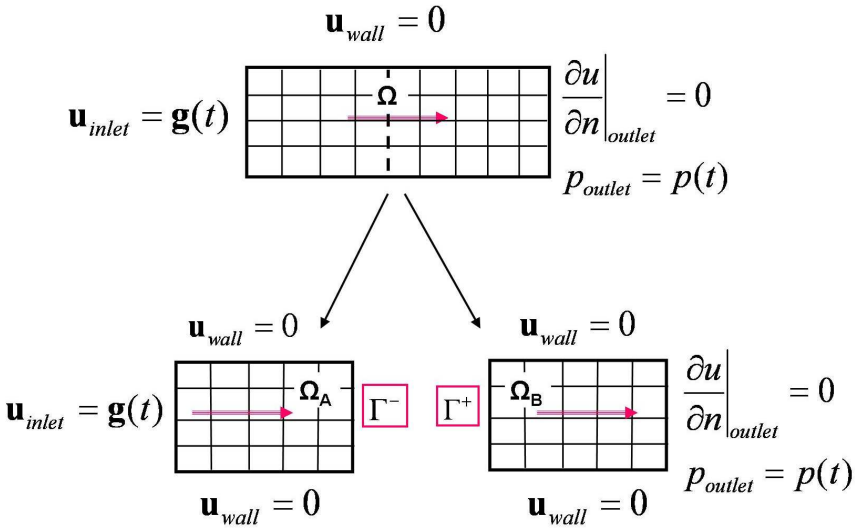


Figure 4.31: Large computational domain is subdivided into two non-overlapping patches. The red arrows indicate direction of a primary flow.

Solution of flow equations with *overlapping* patches adds additional layer of complexity. In figure 4.32 we provide an illustration of two overlapping domains. In simulations with overlapping domains the data for the pressure and velocity boundary conditions is taken from the inner elements of the patch, thus *decoupling* the pressure and velocity interfaces. It is known that the error introduced by the time-splitting scheme is larger at the boundaries of the domain and decays exponentially in the inward direction. In the case of zero overlap the error contaminating, for example, the pressure at the inlet of domain Ω_B is transfered across the interface and becomes a part of Direchlet boundary condition imposed at the outlet of Ω_A. The velocity computed at the outlet of Ω_A (where erroneous pressure boundary

condition is imposed) is transfered across the interface and imposed as Dirichlet boundary condition at the inlet of Ω_B. If the solution is not sufficiently smooth, the described above mechanism may lead to amplification of numerical error and instability. Use of overlapping domains overcomes the aforementioned difficulty and enhances stability.

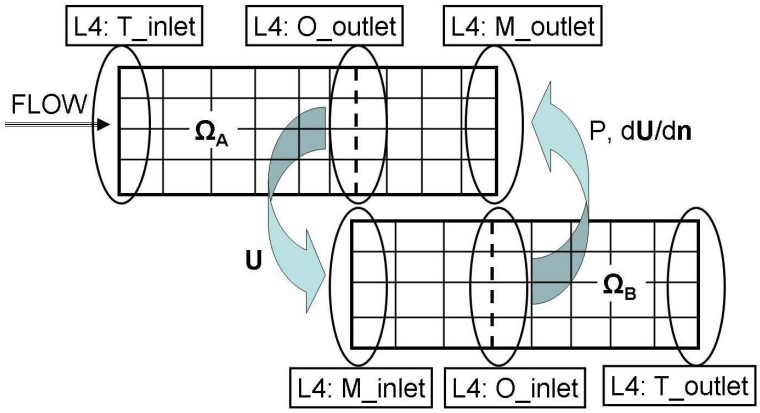


Figure 4.32: Schematic representation of two overlapping patches and interfaces.

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 the two-level domain decomposition can be efficiently performed within a single supercomputer as well as across geographically distributed computers. The latter is characterized by high latency (in comparison to the local communication). Therefore, the MCI implemented in $\mathcal{NEKTARG}$ is designed to optimize the inter-patch message passing, when executed on distributed supercomputers or on clusters using multi-core processors.

A schematic representation of MCI semi-hierarchical decomposition of the global communicator is illustrated in figures 4.33 to 4.34. The first three layers (L1, L2 and L3) of process groups (depicted in figure 4.33) are created by hierarchical decomposition of the global communicator. The fourth layer (L4), shown in figure 4.34(left), allows some degree of overlap, e.g., a process can belong to two or more L4 communicators. On the fifth layer

(L5), communication between ROOTs [3] of different L4 or/and L5 process sub-groups takes place. In the following we provide a more detailed description of the sub-communicators.

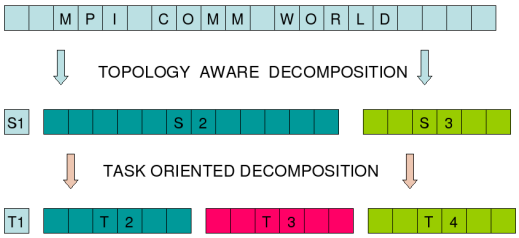


Figure 4.33: Multilayer Communicating Interface: High level communicator splitting. *MPI_COMM_WORLD* is subdivided according to the computer topology to form three non-overlapping process sub-group (S_j). The S_2 group is sub-divided using task-oriented splitting and four non-overlapping T_j groups are created. Cells represent processes.

The first layer (L1) of the MCI is composed of all available processes composing a default communicator setup by $MPI\_Init()$ function (i.e., *MPI_COMM_WORLD*). The L2 is derived from L1 using topology-aware splitting of *MPI_COMM_WORLD*, where processes executing on the same computer are grouped in non-overlapping manner (groups S_1, S_2 and S_3 in figure 4.33). The splitting of the global communicator is done dynamically, the MPICH-g2 and MPIg middlewares provide a functionality to easily split processes according to topology [7]. The 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 figure 4.33). The task oriented decomposition is controlled by information provided as an input to $\mathcal{NEKTARG}$ (see Appendix G). When the program is executed on a single supercomputer the L3 communicators are derived directly from the *MPI_COMM_WORLD*.

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 $\mathcal{NEKTARG}$ different types of inlets and outlets are represented by three objects: 1) class TerminalBoundary (see Appendix D); 2) class OvrLapBoundary (see Appendix E); and 3) class MergingBoundary (see Appendix F). Each of the three objects handles communication and provide required functionality for inlets and outlets, i.e., computing flowrates, mean pressure, mapping between patches, exchanging data for inter-patch conditions, etc. To this end, let us present the following L4 sub-communicators implemented in $\mathcal{NEKTARG}$ (see

[3] process with rank zero.

also an illustration presented in figure 4.32):

- *T_inlet*: sub-communicator consists of processes assigned to partitions with elements facing inlet of the global domain Ω.
- *T_outlet*: sub-communicator consists of processes assigned to partitions with elements facing outlet of the global domain Ω.
- *M_inlet*: sub-communicator consists of processes assigned to partitions with elements facing inlet of the patch that interfaces with the adjacent patch.
- *M_outlet*: sub-communicator consists of processes assigned to partitions with elements facing outlet of the patch that interfaces with the adjacent patch.
- *O_inlet*: sub-communicator consists of processes assigned to partitions with elements which faces are conforming with the faces of elements of the outlet of the adjacent patch.
- *O_outlet*: sub-communicator consists of processes assigned to partitions with elements which faces are conforming with the faces of elements of the inlet of the adjacent patch.

The point-to-point communication between the ROOTs of L4 is performed over the global communicator (MPI_COMM_WORLD), while the communication between the processes within L4 is performed over the local L4 sub-communicator. 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 this data to a file (or transfers it to the dedicated IO processor), or communicates with coupled 1D flow solver. The mapping as well as number of processes in the L4 communicators is not known a priori, and it is done at the beginning of the simulation. 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.

Figure 4.34(left) depicts an example of a solver executing four tasks in the third MCI layer. 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 used for co-processing of the intermediate results, e.g. IO, system calls and etc., while tasks 2-4 solve the 3D flow equations in arterial domains schematically represented at the right of figure 4.34(left).

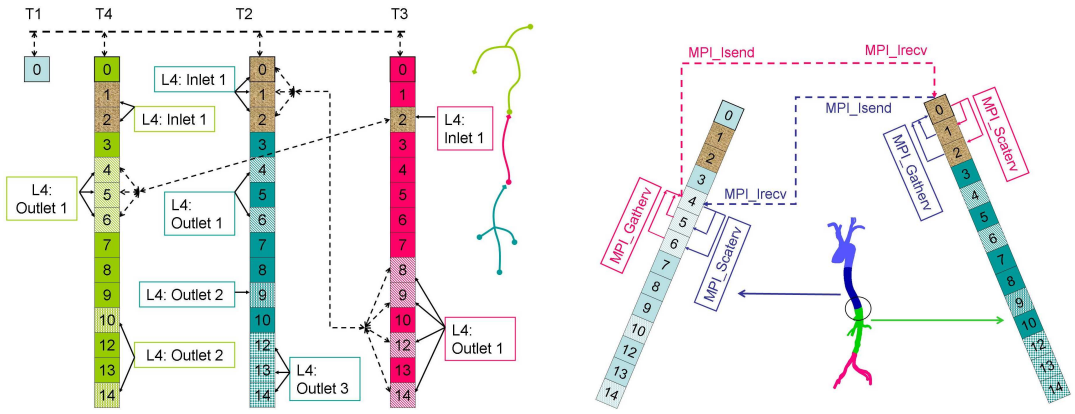


Figure 4.34: Multilayer Communicating Interface. Left: Low level communicator splitting. Here four third-level process sub-groups (T_j) are shown. The low level communicator splitting is performed within each of the T_j sub-group. The inlet and outlet communicators are created. Data exchange between processes from each task T_j is performed through communicators from the fifth-layer of MCI, which combines roots of the third-level communicators. Right: three-step algorithm for inter-patch communication.

In the illustration of figure 4.34(left) each patch has one inlet and different number of outlets, and accordingly different number of process sub-groups (denoted by inlet and outlet sub-groups) are created on the fourth MCI layer. Due to high geometrical complexity of the computational domain, the standard domain decomposition (performed by METIS [3]) may result in some degree of overlap on the fourth-level of process sub-groups, since spectral elements from different outlets or inlets may belong to the same partition, the limiting case would be assigning one process for entire patch. However, when the number of processes assigned to the solution of tightly-coupled problem increases, the possibility of the overlap vanishes. In section 4.3.2.3 we have presented a simple algorithm for communicator splitting in a case of one-to-many mapping between a partition and inlets or outlets. The fifth-layer of MCI is responsible for data exchange across the sub-groups T_j of the third-layer.

Solving flow equations with 2DD, requires imposing interface boundary conditions for the velocity and pressure as explained earlier in this section, which necessitates data exchange across the patch interfaces. The inter-patch communication is implemented in three steps, as illustrated in figure 4.34(right) and explained next. Consider two patches, which share an interface Γ as shown in figure 4.31. As a result of the standard (1DD) domain decomposition within each patch, only a fraction of partitions will be assigned elements which faces belong to Γ; let us denote this partitions as *interface* partitions. Since we have one-to-one mapping between processes and partitions, only processes that are assigned to

the interface partitions should communicate in order to exchange data required by the IPC. For example, in the illustration of figure 4.34(right) processes 4-6 are assigned the interface partitions of the blue patch, while processes 0-2 assigned the interface partitions of the green patch. On the first step, data from the interface partitions, is gathered to the ROOT process of L4 sub-communicator. On the second step, point-to-point communication between the ROOTs of corresponding L4 sub-communicators 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.

We should note that exchanging data through the ROOT processes of L4 sub-groups is an optimization made for simulations on distributed computers to minimize the traffic over the slower network. The alternative approach is to allow each process assigned to the interface partition of one patch to communicate with processes assigned to the interface partition of the neighbor patch, which decreases the intra-patch but increases the inter-patch communication.

Inter-patch conditions

The inter-patch conditions (IPC), required for coupling 3D patches, are computed explicitly and include 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 figure 4.31. The outlet of patch Ω_A marked as Γ^- conforms with the inlet of patch Ω_B marked as Γ^+.

To impose IPC we follow a procedure similar to the discontinuous Galerkin (DG) method [32]. The hyperbolic component of the Navier-Stokes equation dictates the choice of interface condition for the velocity based on the upwinding principle. Assuming that $\mathbf{u} \cdot \mathbf{n} \geq 0$ (with $\mathbf{n}$ pointing outward) at the patch outlet we impose the inlet velocity condition in patch B as

$$\mathbf{u}^{n+1}|_{\Gamma^+} = \mathbf{u}^n|_{\Gamma^-}, \tag{4.12}$$

where the superscripts denote time steps. The velocity flux at the patch A outlet is computed as weighted average of the normal velocity derivatives from both sides of the interface, i.e.,

$$\frac{d\mathbf{u}^{n+1}}{dn}\bigg|_{\Gamma^-} = c_1 \frac{d\mathbf{u}^n}{dn}\bigg|_{\Gamma^-} - (1-c_1)\frac{d\mathbf{u}^n}{dn}\bigg|_{\Gamma^+}, \tag{4.13}$$

where the coefficient c_1 is in the range $0 \leq c_1 < 1$. In the present study we use $c_1 = 0.5$. Alternative choices of numerical fluxes may be considered; for example, imposing the total flux for the velocity at Γ^+ was advocated in [41, 47]. Also, different choices for the flux in the *DG* formulation can be found in [10, 11].

The pressure at the patch outlet is given by

$$p^{n+1}|_{\Gamma^-} = F(t - t_0)\bar{p} + (1 - F(t - t_0))p_{IC}, \tag{4.14}$$

where $\bar{p} = 0.5(p^n|_{\Gamma^-} + p^n|_{\Gamma^+})$, $F(t - t_0) = (1 - e^{-\alpha(t^{n+1} - t^0)})^\beta$ with $\alpha > 0$ and $\beta > 0$ and p_{IC} is the initial conditions for the pressure; in our simulations we used $\alpha = 20$ and $\beta = 2$. The role of p_{IC} is explained bellow. The filter function $F(t - t_0)$ suppresses erroneous

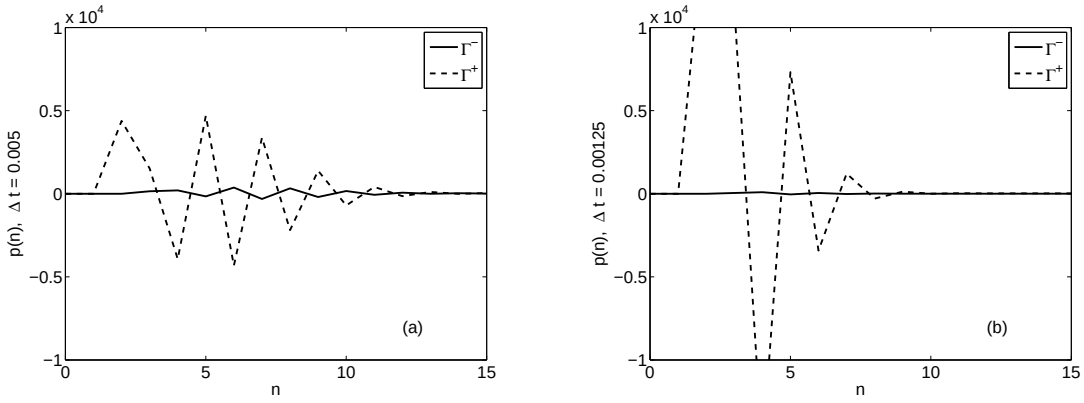


Figure 4.35: Steady flow simulation with 2DD: High amplitude oscillations of the pressure at the patch interface in the beginning of a simulation performed in a convergent pipe domain subdivided as illustrated in figure 4.39 (left). Solution is computed with third-order approximation in space and different size of time steps: (a) $\Delta t = 0.005$, and (b) $\Delta t = 0.00125$. High amplitude oscillations in p_{Γ^+} are reduced by the filter function $F(t-t_0)$ resulting in low amplitude oscillations in p_{Γ^-}.

large pressure oscillations. The pressure oscillations at the inlet are due to the numerical scheme and incompatible initial conditions; e.g., in the beginning of the simulation the initial velocity field may not satisfy the continuity equation $\nabla \cdot \mathbf{v} = 0$. In simulations on a single domain (1DD), the pressure oscillations at the beginning of a simulation do not affect the stability of the solver. However, in the 2DD formulation, the pressure computed at the inlet Γ^+ is imposed as Dirichlet B.C. at the outlet Γ^-, thus oscillations in p_{Γ^+} propagate to the adjacent patch. In the case of multiple interfaces required in complex arterial networks, out-of-phase pressure oscillations at outlets may lead to catastrophic results. Large oscillations

in pressure may also appear when the simulation is restarted; in this case, the last term of Eqn. (4.14) works to reduce the oscillations and, during the first few time steps, it keeps the pressure values at the patch interface close to p_{IC}. In figure 4.35 we show the oscillations of the mean pressure $\tilde{p} = (A_\Gamma)^{-1} \int_\Gamma p dA$, computed at the patch interface for two choices of Δt. The dramatic reduction in the amplitude of the oscillations at Γ^- is a result of applying the filter function $F(t - t_0)$ of Eqn. (4.14). In the case of overlapping domains the values of velocity and pressure from *inside* of a patch are used to impose boundary conditions at the adjacent patch.

The error introduced by the explicit treatment of the interface boundary conditions is controlled by the size of time step. In order to minimize the error, an iterative procedure for computing $\mathbf{u}^{n+1}$ and p^{n+1} may be applied. However, from the computational standpoint, such procedure is inefficient in a sense that the computational time will grow almost linearly with the number of sub-iterations required for convergence of the velocity and the pressure at patch interfaces. In large scale simulations of blood flow in the human arterial tree, computing the solution over one cardiac cycle typically takes 1-3 days at the present time, thus such sub-iterations are computationally prohibitive. We should note, that the typical size of a time step in such simulations is $1E-5$ - $1E-6$ s., and the space discretization error is usually dominant.

An alternative way for enhancing the numerical accuracy is to approximate the boundary condition at the time step $t^{(n+1)}$ by applying a penalty term, i.e.,

$$\mathbf{u}^{n+1}|_{\Gamma^+} = \mathbf{u}^n|_{\Gamma^-} + \alpha_{BC} F(t - t_0)(\mathbf{u}^n|_{\Gamma^-} - \mathbf{u}^n|_{\Gamma^+}) \tag{4.15}$$

or by extrapolation, i.e.,

$$\mathbf{u}^{n+1}|_{\Gamma^+} = \mathbf{u}^n|_{\Gamma^-} + \alpha_{BC} F(t - t_0)(\mathbf{u}^n - \mathbf{u}^{n-1})|_{\Gamma^-}, \tag{4.16}$$

where $0 \leq \alpha_{BC} \leq 1$ is a relaxation parameter and $0 \leq F(t - t_0) \leq 1$. The choice of α_{BC} affects both accuracy and stability, however, the latter dictates its value.

In the SEM approach, the solution for velocity and pressure is performed in *modal* space, hence, the values of velocity and pressure can be transfered and imposed as B.C. in modal space bypassing expensive transformation from modal to physical space and vice-versa. Of course, this can be done only when the computational meshes at Γ^- and Γ^+ are

conforming. Imposing B.C. in modal space has an additional advantage: we can exploit the hierarchical structure of the base functions and reduce communication by 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 computation time associated with imposing IPC. The latter is very important in ultra-parallel simulations.

4.4.2 Non-overlapping domains: Results

In this section we compare the parallel efficiency of two solvers based on the 1DD and 2DD approaches. Subsequently, we investigate the accuracy of numerical simulations of steady and unsteady flow performed with the 1DD and 2DD methods.

From the computational standpoint, one of the advantages of the 2DD approach over the 1DD approach is the minimization of blocking communication between processes, which enhances parallel efficiency. In the first simulation we use a computational domain consisting of 67456 tetrahedral elements elements sub-divided into two equal size patches. Separate runs are performed using the 1DD and the 2DD approaches, and in figure 4.36 we plot the mean CPU-time per time step. The computational saving with the 2DD approach is clear even for this relatively small size problem.

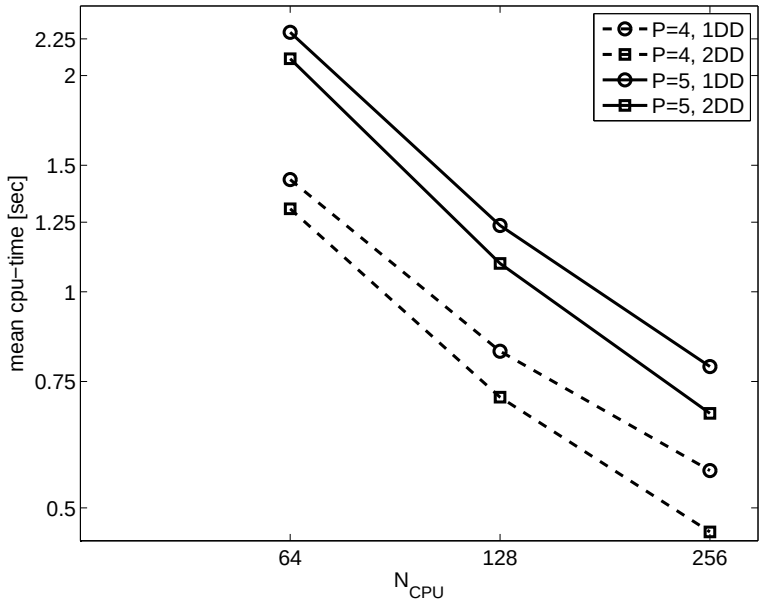


Figure 4.36: Simulation with 1DD and 2DD: performance. Y-axis is the mean CPU-time required per time step. Problem size: 67456 tetrahedral elements, polynomial order $P = 4$ (dash line) and $P = 5$ (solid line). Computation performed on the CRAY XT3 supercomputer.

In figure 6 we plot the CPU-time per time step for the solution of large problem computed with the 1DD and 2DD approach. The relatively large deviation in the CPU-time is due to different number of iterations at different time steps and also due to sharing the communication network with other users during our tests. The computational savings with the 2DD approach is clear and increase with the number of processors.

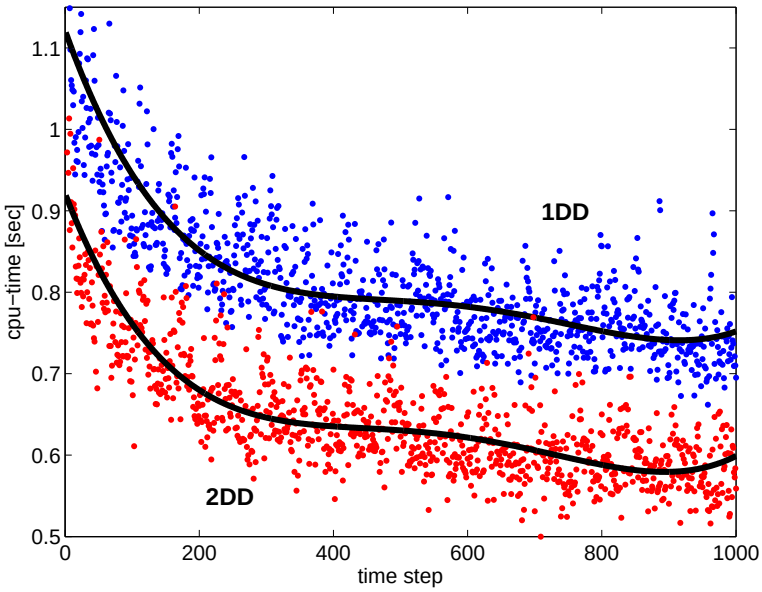


Figure 4.37: Simulation with 1DD and 2DD on 1024 cores of Ranger: The domain is subdivided into two parts: patches C and D (106,219 and 77,966 elements, $P = 4$) in figure 4.38(right). Dots - CPU-time per time step; solid lines - least-square approximation.

The potentially great advantage of the 2DD approach is in simulating very large scale problems. There are two difficulties in the solution of a such problems: (a) limited available memory per processor, and (b) tight communication between thousands of processors. Limited memory requires the use of many processors, which decreases the volume of computation versus volume of communication ratio. Implementation of coarse partitioning of a computational domains into M patches leads to partitioning of the communicator into M non-overlapping groups of processes [44]. The tight communication between processes is performed as intra-group message passing only. The communication between groups is required to exchange data across patch interfaces and it is limited to a small number of processes. This communication is non-blocking and is performed once in a time step. The overall computational efficiency is strongly affected by the parallel efficiency in the solution of a problem within a patch. In figure 4.38 we show the parallel efficiency of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}$ in simulation of a steady flow in aorta. On a coarse level the domain is partitioned into four patches. The parallel efficiency E_p is computed as:

$$E_p(N) = \frac{T_n/T_N}{N/n},$$

where N is the number of processes employed in simulation, n is the reference (minimum)

number of processes used for parallel solution of a given problem and T_N (T_n) is the mean CPU-time per time step required for simulation on N (n) processes. The results of figure 4.38 verify that the overall parallel efficiency depends on the parallel efficiency of each subproblem on the different patches. Here patches A and C suffer from relatively low efficiency. In terms of optimizing the overall performance, this implies that we should optimize the parallel performance on the individual patches – a much simpler task than dealing with tens or hundreds of thousands of processors involved in the solution of the overall problem.

patch	Nel	P	# DOF
A	120,813	6	69,588,288
B	20,797	6	11,979,072
C	106,219	6	61,182,144
D	77,966	6	44,908,416
total	**325,795**	**6**	**187,657,920**

Table 4.4: Solution of large scale problem, computational complexity: flow simulation in the domain of figure 4.38(right). On a coarse level the computational domain is subdivided into four patches A-D. Nel - number of spectral elements. DOF - number of degrees of freedom, computed as a total number of quadrature points: $DOF = Nel * (P+3)(P+2)^2$ required for exact integration of linear terms.

Our last example in this section is simulation of unsteady flow in a domain of 147 arteries, subdivided to 12 non-overlapping patches. The domain consists of 1,244,295 spectral element, and the solution is approximated with $P = 6$, which results in 2.87B degrees of freedom problem. The dimensions of L3 communicator are from 16 to 3072 ranks, and the total number of cores used for the test is 18,576. Simulation performed on Ranger (TACC) showed that the solution of a such large problem can be obtained in about one second per time step. The acceleration techniques described in section 3.5 were not implemented in this test, we project that with acceleration technique the solution can be obtained in about 0.6s to 0.65s at each time step. The goal of the performed test was to demonstrate the capabilities of the two-level approach and MCI.

4.4.2.1 Accuracy verification

Our objective is to evaluate the potential loss of spectral accuracy in the numerical solution obtained using the 2DD approach. The accuracy degrades due to two factors: a) explicit treatment of the IPC; and b) incomplete transfer of the velocity and pressure modes in

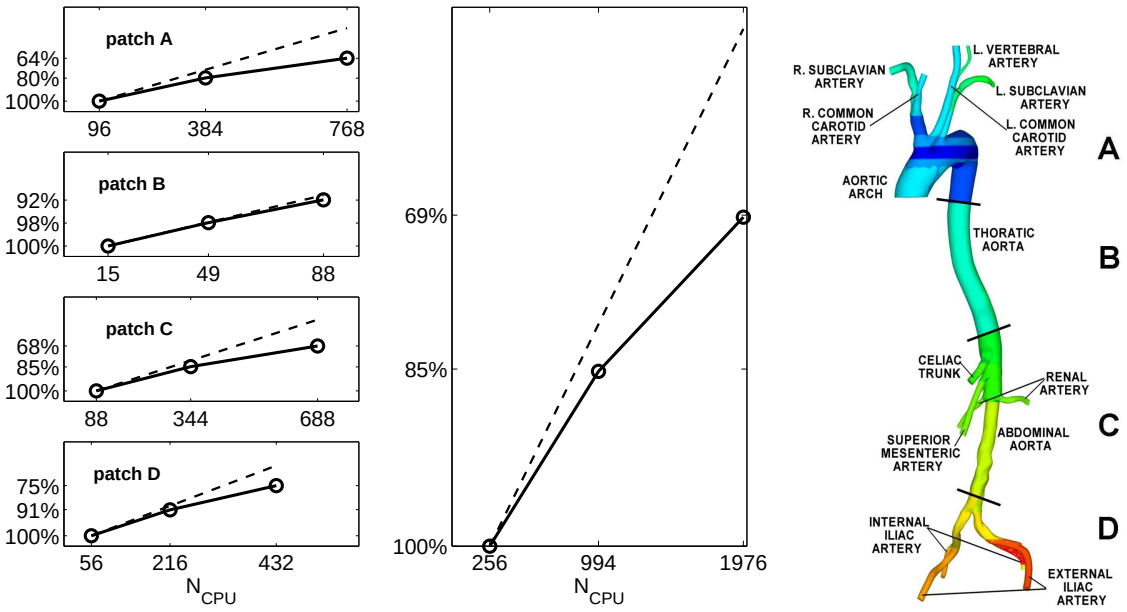


Figure 4.38: Simulation with 2DD: parallel efficiency. Steady flow simulation in the human aorta. The domain is sub-divided into four patches. Details on computational complexity are summarized in table 4.4. Y-axis - parallel efficiency of a solver, E_p. Left: parallel efficiency in solution of tightly coupled system withing a patch. Center: overall parallel efficiency. Right: geometry of a human aorta; large computational domain is subdivided into four patches. Computation performed on the CRAY XT3 supercomputer.

imposing the inter-patch conditions.

We use the following notations:

P - order of polynomial expansion.

P_{VBC} - maximum order of polynomial expansion in imposing the *velocity* IPC.

P_{PBC} - maximum order of polynomial expansion in imposing the *pressure* IPC.

P_{FBC} - maximum order of polynomial expansion in imposing the *velocity flux* IPC.

4.4.2.2 Benchmark Problem

Numerical simulations were performed in a simple computational domain whose shape is similar to a converging blood vessel. Although the domain is axi-symmetric, we use full 3D solver and non axisymmetric mesh to perform the simulations.

The domain consists of two blocks A and B as illustrated in figure 4.39. For reference, we also combined the two blocks into one, i.e., block AB. In the first configuration, the patch interface is normal to the z-axis and in the second configuration it intersects the

z-axis at 80 degrees. First, we performed a steady flow simulation with Reynolds number

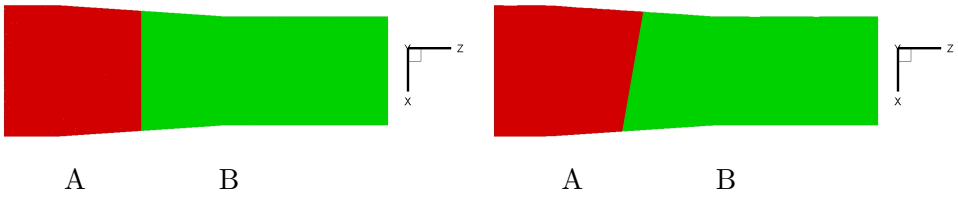


Figure 4.39: Illustration of two configurations of two-level domain decomposition into patches A and B. The interfaces are centered at $z = 5$. Left: interface is normal to the pipe axis. Patch A (red) has 4203 tetrahedral spectral elements and patch B (green) has 8906 elements. Right: interface is at an angle of 80^o to the pipe-axis. The sizes of patches A and B are 7246 and 11331 tetrahedral spectral elements, respectively. Primary flow direction $(z-)$ is from left to right.

$Re = \frac{D_A U_m}{\nu} = 350$ where D_A is a diameter of the inlet of patch A, U_m is the mean velocity at the inlet and ν is a kinematic viscosity. Poiseuille flow is imposed at the entrance of the block A. At the outlet of the patch B a fully developed flow is assumed defined by zero velocity flux and constant pressure. Second, we simulate unsteady flow by imposing the Womersley velocity profile at the inlet of patch A. In this simulation the main characteristics of the flow are $Re = 350$ and Womersley number $Ws = (D_A/2)^2\sqrt{\omega/\nu} = 4.375$, which are typical values for the arterial flow in vessels with diameter of 4 to 5mm.

4.4.2.3 Convergence of flowrate and mean pressure at patch interface

We define the error in mass conservation across the interface as:

$$\epsilon_Q = \left| \frac{Q|_{\Gamma^+} - Q|_{\Gamma^-}}{Q|_{\Gamma^+}} \right|, \tag{4.17}$$

where $Q = |\int_\Gamma \mathbf{v}\cdot\mathbf{n}dA|$ and $\mathbf{n}$ is an unit vector normal to Γ. The error in the mean pressure $\tilde{p}$ is computed from

$$\epsilon_p = \left| \frac{\tilde{p}|_{\Gamma^+} - \tilde{p}|_{\Gamma^-}}{\tilde{p}|_{\Gamma^+}} \right|. \tag{4.18}$$

First, we consider steady flow simulation with relatively low spatial resolution *within* a patch, $P = 3$. In figure 4.40 we plot $\tilde{p}(t)$ at the interface. In the first simulation the pressure IPC were imposed with $P_{PBC} = 1$ (i.e., using vertex modes only) and relatively large size of time step was used i.e., $\Delta t = 0.005$. Low spatial and temporal resolution result in high frequency oscillations in $\tilde{p}(t)$, whose amplitude is of order Δt at Γ^+ and considerably lower

at Γ^-, as shown in figure 4.40(a). The next two simulations are performed with smaller size of the time step $\Delta t = 0.00125$ (figure 4.40(b)) or with higher spatial accuracy in imposing the pressure IPC, i.e., $P_{PBC} = 2$ (figure 4.40(c)); we observe that the oscillations can be removed by either reducing the size of a time step or by increasing P_{PBC}.

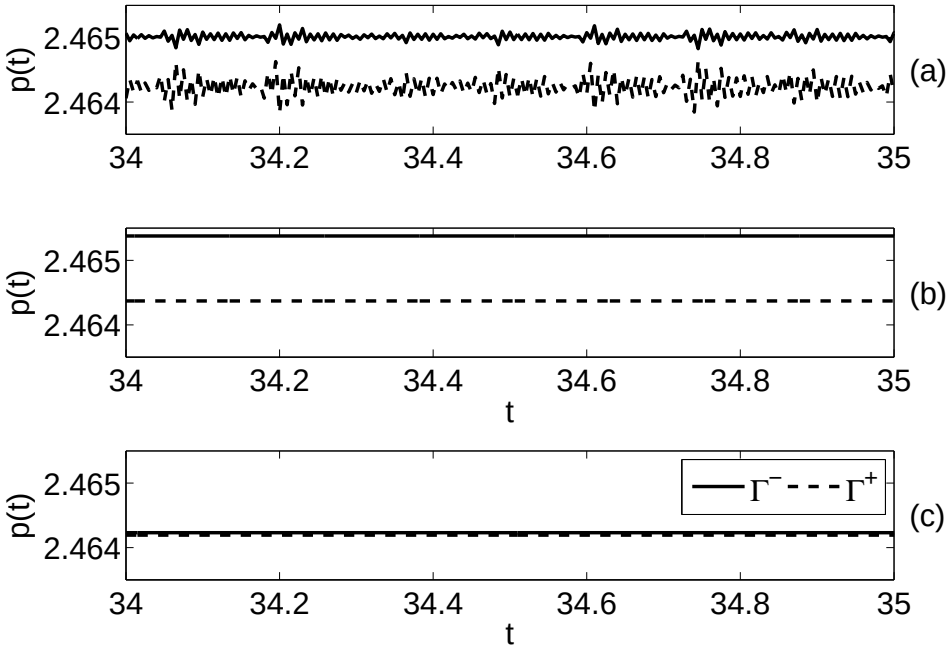


Figure 4.40: Steady flow simulation with 2DD: Simulation performed in domain of convergent pipe, sub-divided as illustrated in figure 4.39 (left). Convergence of the mean pressure at the patch interface. Solution computed with $P = 3$. Velocity IPC is imposed with $P_{VBC} = 3$.
(a) - $\Delta t = 0.005$ pressure IPC is imposed with $P_{PBC} = 1$.
(b) - $\Delta t = 0.00125$ pressure IPC is imposed with $P_{PBC} = 1$.
(c) - $\Delta t = 0.005$ pressure IPC is imposed with $P_{PBC} = 2$.

To investigate further the effect of spatial under-resolution we consider a steady flow simulation in the domain illustrated in figure 4.39(left) performed with higher order of accuracy i.e., $P = 5$ and $\Delta t = 0.00125$. We focus on the effects of spatial under-resolution in imposing IPC. In table 4.5 we summarize details of simulations where the velocity IPC were imposed with $P_{VBC} = 1, 2, 3$ and pressure IPC with $P_{PBC} = 1, 2$. Exponential convergence in ϵ_Q and $\epsilon_{\tilde{p}}$ is observed. Simulations (a), (b) and (c) also show that imposing velocity IPC with higher order of accuracy has significant effect on *both* ϵ_p and ϵ_Q. We should note that the time splitting scheme (2.34b)-(2.34e) introduces an error in mass conservation. For example, the error in mass conservation in simulation (d) was $\epsilon_Q = |(Q_{inlet} - Q_{outlet})/Q_{inlet}| = 6.1e-7$

in patch A and $\epsilon_Q = 1.4e-6$ in patch B, which is comparable to the error introduced by incomplete transfer of modes in imposing the velocity IPC.

Decoupling the solution for the velocity and the pressure allows to approximate the two fields with the same polynomial order. The solution for the Poisson equation for the pressure, supplemented with Dirichlet boundary condition at the outlet and Neumann boundary condition at the Dirichlet-velocity boundaries, is unique and the spurious pressure modes are eliminated [79]. However, imposing outlet pressure boundary condition with $P_{PBC} = P$ or $P_{PBC} = P-1$ may lead to an instability. In our tests, we observed that using $P_{PBC} = P$ or $P_{PBC} = P-1$ (in the domain of figure 4.39) led to unstable simulations. In steady flow simulation, only $P_{PBC} = P$ led to instability. This issue deserves additional investigation in the future. In simulations (c) and (d) (see table 4.5) we observed that using $P_{PBC} = P-1$ gave practically the same result as $P_{PBC} = P-2$ as far as the mass conservation is concerned.

simulation	N	Δt	N_{VBC}	N_{PBC}	N_{FBC}	ϵ_Q	ϵ_P
a	5	1.25E-3	**1**	1	5	3.3E-2	1.0E-2
b	5	1.25E-3	**2**	1	5	8.2E-4	6.8E-4
c	5	1.25E-3	**3**	1	5	1.3E-7	3.1E-4
d	5	1.25E-3	**3**	**2**	5	1.6E-7	1.7E-5

Table 4.5: Steady flow simulation with 2DD: exponential convergence in the error of a flow rate Q and mean pressure $\tilde{p}$ computed across the patch interface. Simulation performed in a domain of convergent pipe, sub-divided as illustrated in figure 4.39(left).

In figure 4.41 we plot the pressure distribution from both sides of the interface. The upper plots show the computed pressure at the inlet of the patch B, $p|_{\Gamma^+}$, while the lower plots show the pressure imposed as IPC, $p|_{\Gamma^-}$ at the outlet of patch A. Under-resolution in imposing velocity boundary condition results in significant discontinuity in the second derivatives of velocity computed at the inlet, and this induces pressure oscillations due to pressure-velocity coupling through the pressure B.C. In contrast, incomplete transfer of pressure modes from Γ^+ to Γ^- is equivalent to applying a cut-off filter and results in considerable smoothing of the pressure field particularly in a case of under-resolved velocity field.

The effect of explicit treatment of velocity IPC on the mass conservation across the interface was investigated also using unsteady flow simulation, where third-order accuracy in space ($P = 3$) and second-order accuracy in time discretisation within each patch was

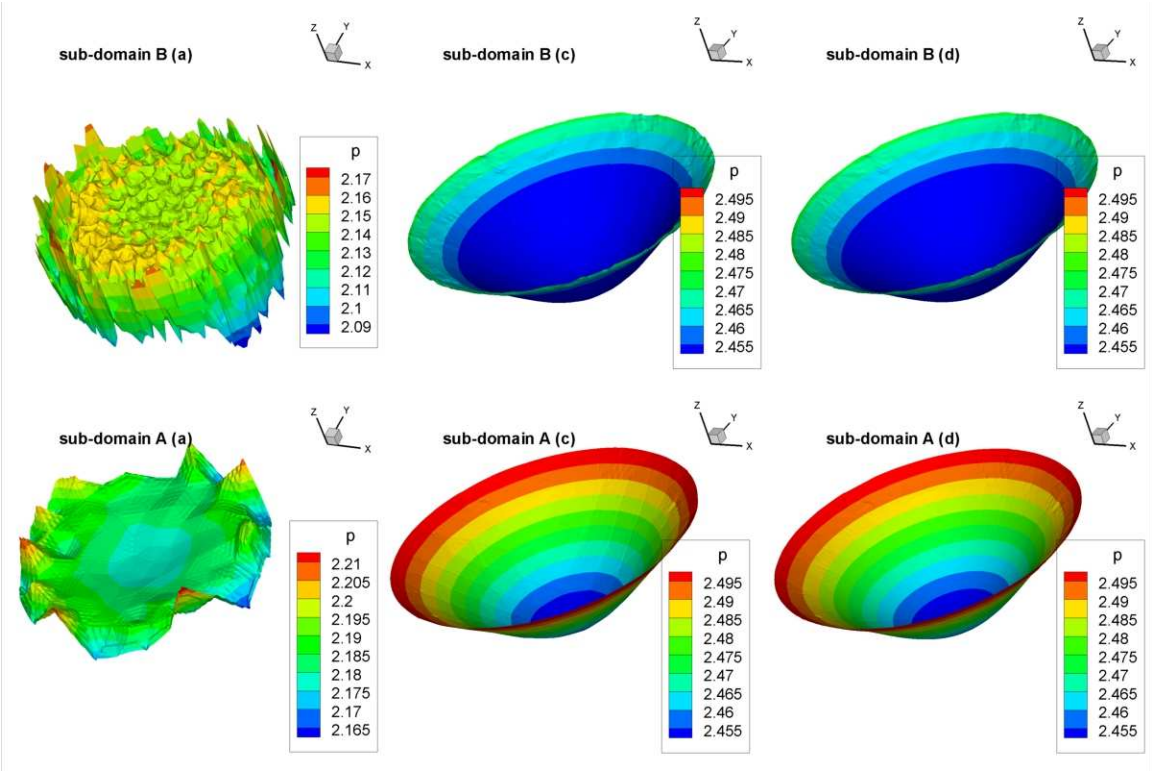


Figure 4.41: Steady flow simulation with 2DD: pressure distribution from both sides of patch interface. Simulation performed in a domain of convergent pipe, sub-divided as illustrated in figure 4.39(left) with $P = 5$ and $\Delta t = 0.00125$. The set-up is consistent with table 4.5. Top plots: $p|_{\Gamma^+}$; bottom plots: plots $p|_{\Gamma^-}$. Left: case (a), $P_{VBC} = 1$, $P_{PBC} = 1$. Center: case (c), $P_{VBC} = 3$, $P_{PBC} = 1$. Right: case (d), $P_{VBC} = 3$, $P_{PBC} = 2$.

employed. Velocity and flux IPC were imposed with full resolution ($P_{VBC} = 3,\ P_{FBC} = 3$) and pressure IPC with $P_{PBC} = 1$. Velocity and flux boundary conditions were imposed using three different methods: a) according to formula (4.12); b) using the penalty formulation (4.15) with $\alpha = 0.5$; and c) using the extrapolation formula (4.16) with $\alpha = 0.5$. The results are summarized in figure 4.42. All three methods are based on first-order (in time) explicit scheme in imposing IPC, however the coefficients (C_i) of the leading term in the truncation error are different as we show in figure 4.42(right).

4.4.2.4 Errors due to inter-patch interface

Here we compare the error in the pressure and vorticity fields obtain with the 1DD and 2DD approaches. In figure 4.43 we show the computational domain and the location where the pressure field was extracted for comparison. In figure 4.44 we compare the pressure computed with the two methods. We observe that the error in the numerical solution is

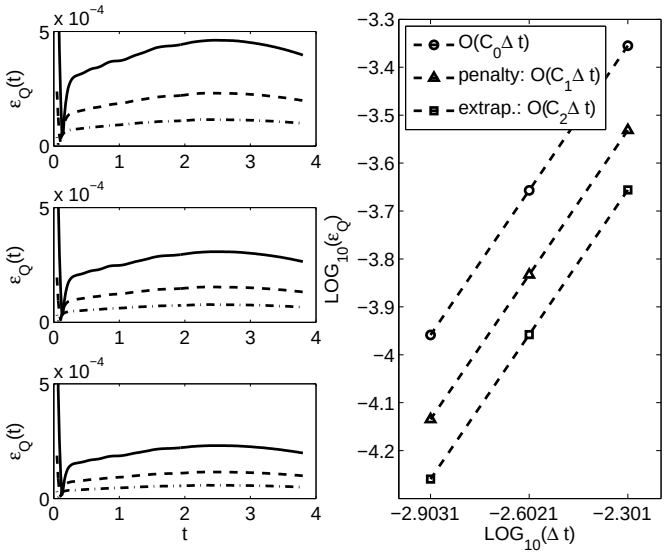


Figure 4.42: Unsteady flow simulation with 2DD: Convergence of flow rates at the patch interface. Simulation performed in a domain of convergent pipe, sub-divided as illustrated in figure 4.39 (left). Left upper plot: $\alpha = 0.0$. Left center plot: penalty formulation, $\alpha = 0.5$. Left lower plot: extrapolation, $\alpha = 0.5$. Solid line $\Delta t = 0.005$, dash line $\Delta t = 0.0025$, dash-dot line $\Delta t = 0.00125$. Right: convergence rate of numerical error ϵ_Q at time $t = 1.9$.

greatest in the vicinity of patch interface and rapidly decays upstream where it is of order Δt, and downstream, where it converges to the imposed value of the pressure at the outlet, i.e., $p = 0$. The localization of numerical error at the vicinity of the patch interface is due to reduced space for imposing IPC and also due to explicit treatment of the IPC.

Next we compare the vorticity field. In figure 4.45 we plot the $y-$ component of the vorticity, ω_y computed with 1DD and 2DD. We define the maximum deviation in the vorticity field by:

$$\epsilon_\omega = \frac{MAX(|\omega(1DD) - \omega(2DD)|)}{\omega_s(1DD)},$$

here the value of a scaling factor $\omega_s(1DD)$ is computed at a point where the difference $|\omega(1DD) - \omega(2DD)|$ is maximum; results are presented in figure 4.46. The maximum value of ϵ_{ω_y} in simulation with $P_{PBC} = 1$ ($P_{PBC} = 2$) is about 4% (2%) and decreases by an order of magnitude a short distance from the interface.

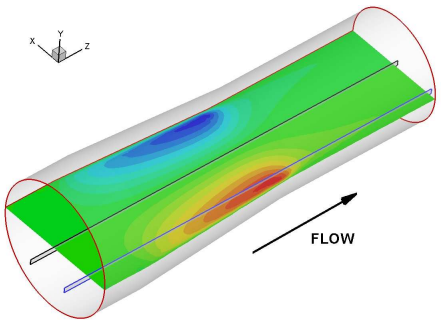


Figure 4.43: Illustration of computational domain and location of slice $y = 0$ and lines $x = 0,\ y = 0$ (black) and $x = -1.6,\ y = 0$ (blue); colors represent the non-dimensional u-velocity.

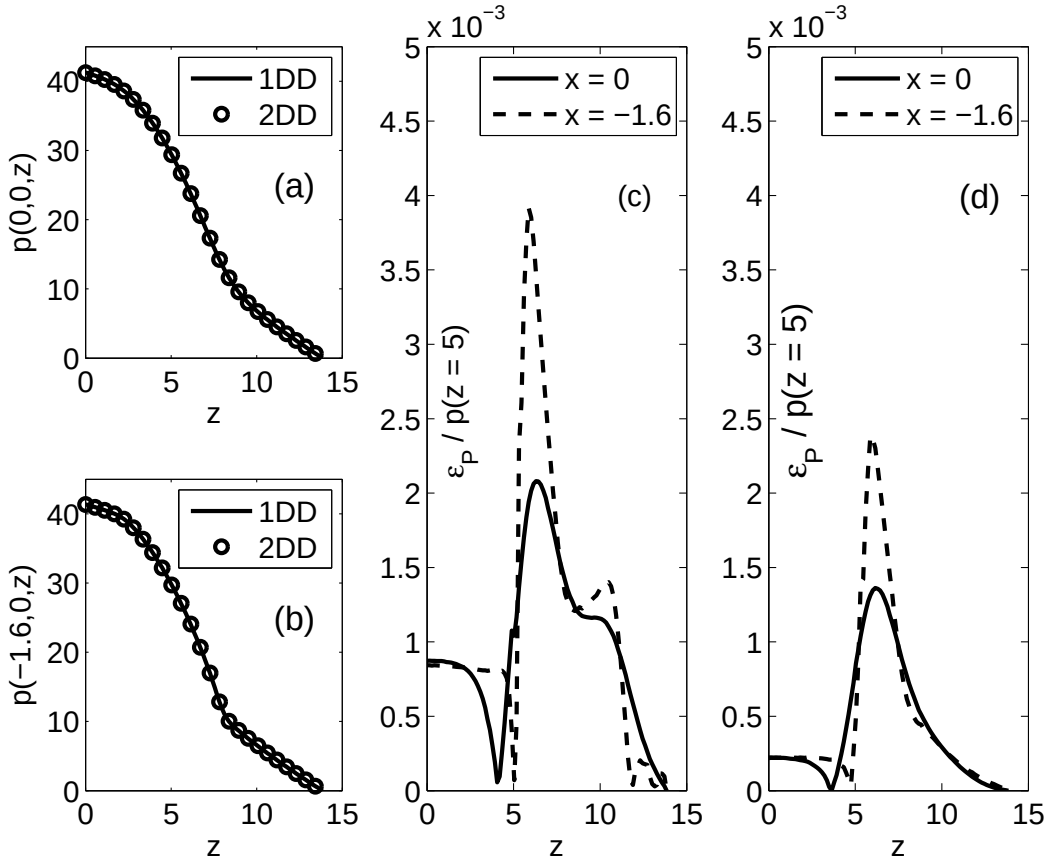


Figure 4.44: Unsteady flow simulation with 2DD in the computational domain of figure 4.39(right). Pressure along lines $y = 0,\ x = 0$ and $y = 0,\ x = -1.6$ as marked in figure 4.43. (a) and (b) non-dimensional pressure values computed with 1DD and 2DD. (c) and (d) normalized difference between the pressure computed with 2DD and 1DD; (c) - $P_{VBC} = 3$, $P_{PBC} = 1$, (d) $P_{VBC} = 3$, $P_{PBC} = 2$. $P = 5$, $\Delta t = 0.0005$.

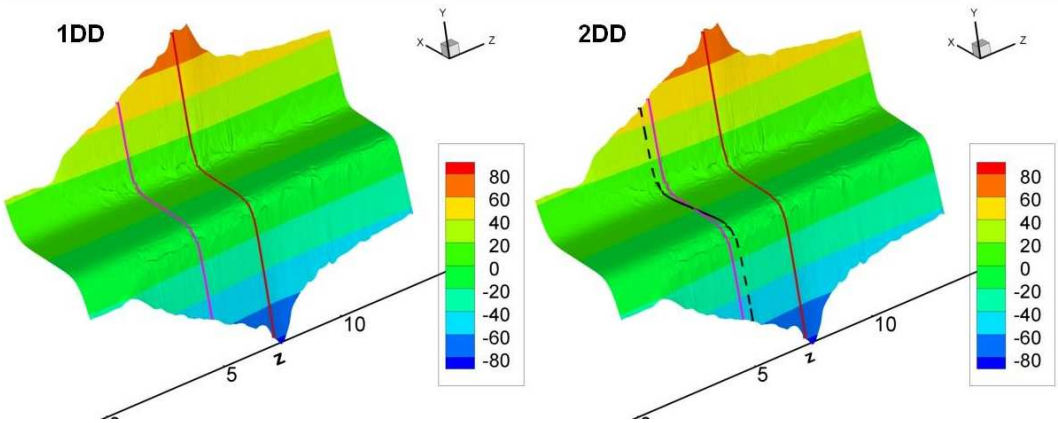


Figure 4.45: Unsteady flow simulation with 2DD: comparison of vorticity filed computed with 1DD and 2DD: Y-component of vorticity field (ω_y) contours at slices $y = 0$. Y-axis is ω_y. Solid lines represent location ($z = 5$ and $z = 7.5$) where ω_y was extracted. Dash line depicts the location of patch interface. $P = 5$, $\Delta t = 0.0005$.

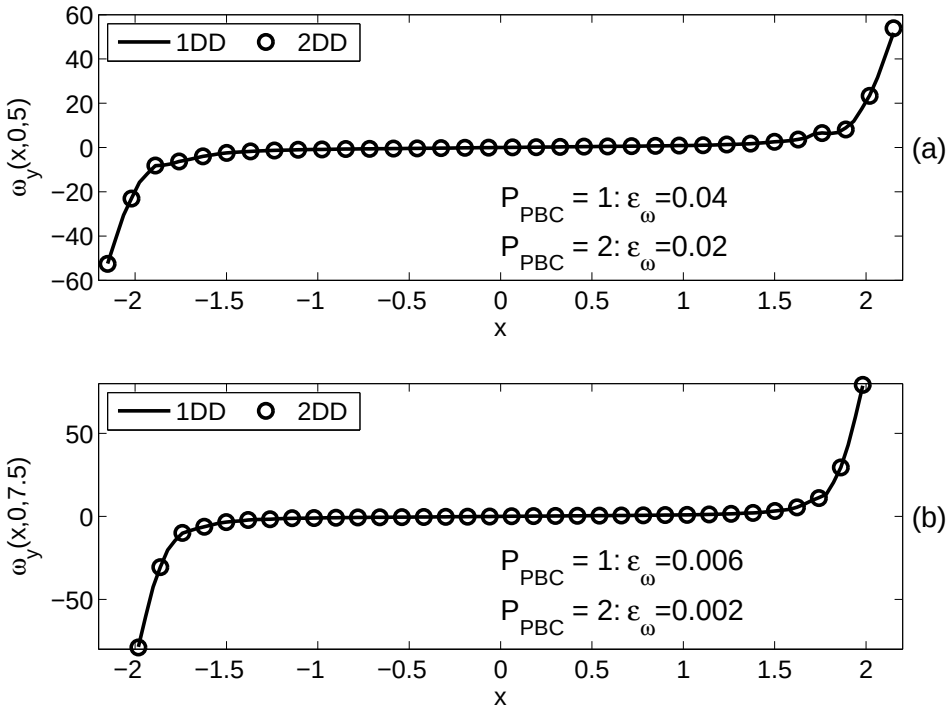


Figure 4.46: Unsteady flow simulation with 2DD: comparison of vorticity filed computed with 1DD and 2DD. Computational domain is illustrated in figure 4.39(right). Y-component of vorticity field, ω_y, extracted at (a) - $y = 0, z = 5$ and (b) - $y = 0, z = 7.5$. ϵ_ω - deviation in ω_y. $P_{VBC} = 3$, $P_{PBC} = 1, 2$, $P = 5$, $\Delta t = 0.0005$.

4.4.3 Overlapping domains: Results

In this section we consider transient and unsteady flow simulations using the two-level method and overlapping patches. We start with evaluating parallel efficiency of $\mathcal{NEKTARG}$, by solving transient flow problem in a pipe-like domain with multiple constrictions. The large computational domain has been subdivided into several overlapping patches as illustrated in figure 4.47. The first simulation is performed on 1024 cores of Ranger and Kraken (CRAY XT5), solution has been integrated for 1000 time steps with $\Delta t = 0.002$, $P = 4$, and initial condition $\mathbf{u} = 0$. In this test we do not implement any acceleration techniques described in section 3.5, LEBP is employed. At the inlet parabolic velocity profile corresponding to $Re = 470$ is imposed, at the outlet fully developed flow is assumed. The number of spectral elements in a single-patch domain is $Nel_1 = 130,880$, while in the cases of 2, 4 and 8 patches the total number of elements is $Nel_2 = 131,994$, $Nel_4 = 134,222$ and $Nel_8 = 138,678$, respectively. The increasing number of elements is due to overlapping regions, each overlapping region has 1114 elements, shared by two neighbor patches. Simulations are performed 10 times on each computer and for each level of coarse discretization. The mean CPU-time required for 1000 time-steps along with the standard deviations are presented in figure 4.48(top). Note, that despite the increasing problem size (due to the overlap), the computational cost is decreasing, as a result of lower volume of communications. In figure 4.48(bottom) we present the strong scaling in simulations with one and four patches. For this test we use relatively low-order polynomial approximation ($P = 4$), which results in the poor scaling in simulations with one patch. However, in simulations with four patches we observe considerably better scalability.

The weak and strong scaling tests have been extended up to 32,768 cores of the BlueGene/P computer of ANL, for that purpose the domain size was increased by adding 16 more patches of equal size. The results are summarized in table 4.6. The excellent scaling obtained on the BlueGene computer is due to factors: a) very good distribution of work load, and b) computer characteristics, specifically extremely low system noise.

Next we perform simulations on CRAY XT5 with 3,8 and 16 patches, using 2048 cores per patch. Unlike BlueGene, CRAY XT5 is more sensitive to the system noise, which typically results in non-uniform CPU-time per time step as presented in figure 4.49. Very good weak scaling is observed. We note that the total number of DOF for 16 patches is about

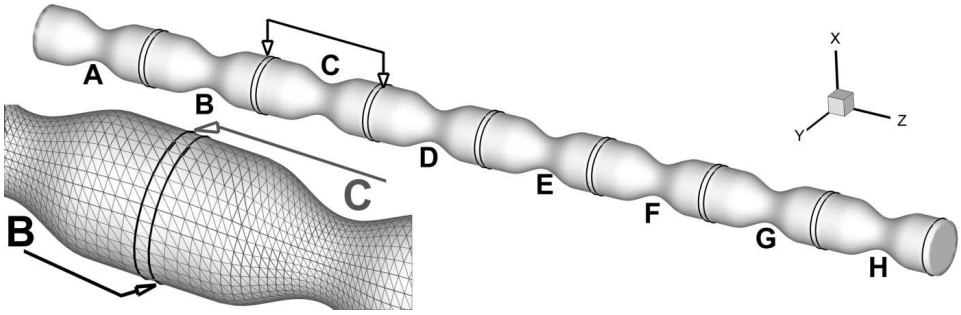


Figure 4.47: Large computational domain is sub-divided into several overlapping patches.

case	Np	# of cores/patch	total # of cores	CPU-time [s]	weak (strong) scaling
A1	3 (A-C)	1,024	3,072	996.98	100% (100%)
A2	8 (A-H)	1,024	8,192	1025.33	97.2% (100%)
A3	16 (A-P)	1,024	16,384	1048.75	95.1% (100%)
B1	3 (A-C)	2,048	6,144	650.67	100% (76.6%)
B2	8 (A-H)	2,048	16,384	685.23	95% (74.8%)
B3	16 (A-P)	2,048	32,768	703.4	92% (74.5%)

Table 4.6: BlueGene/P: flow simulation in the domain of figure 4.47 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, acceleration is not implemented. Simulations have been performed using four cores per node (mode vn).

2.07B. and it takes only about $0.5s$ to solve such a large system, even without implementing acceleration techniques described in section 3.5. The total solution time required for 1000 time steps with 3,8 and 16 patches was 462.3s, 477.2s and 505.05s, respectively. Moreover, considering the very good weak scaling, there is a potential to solve much larger problem within the same wall-clock time.

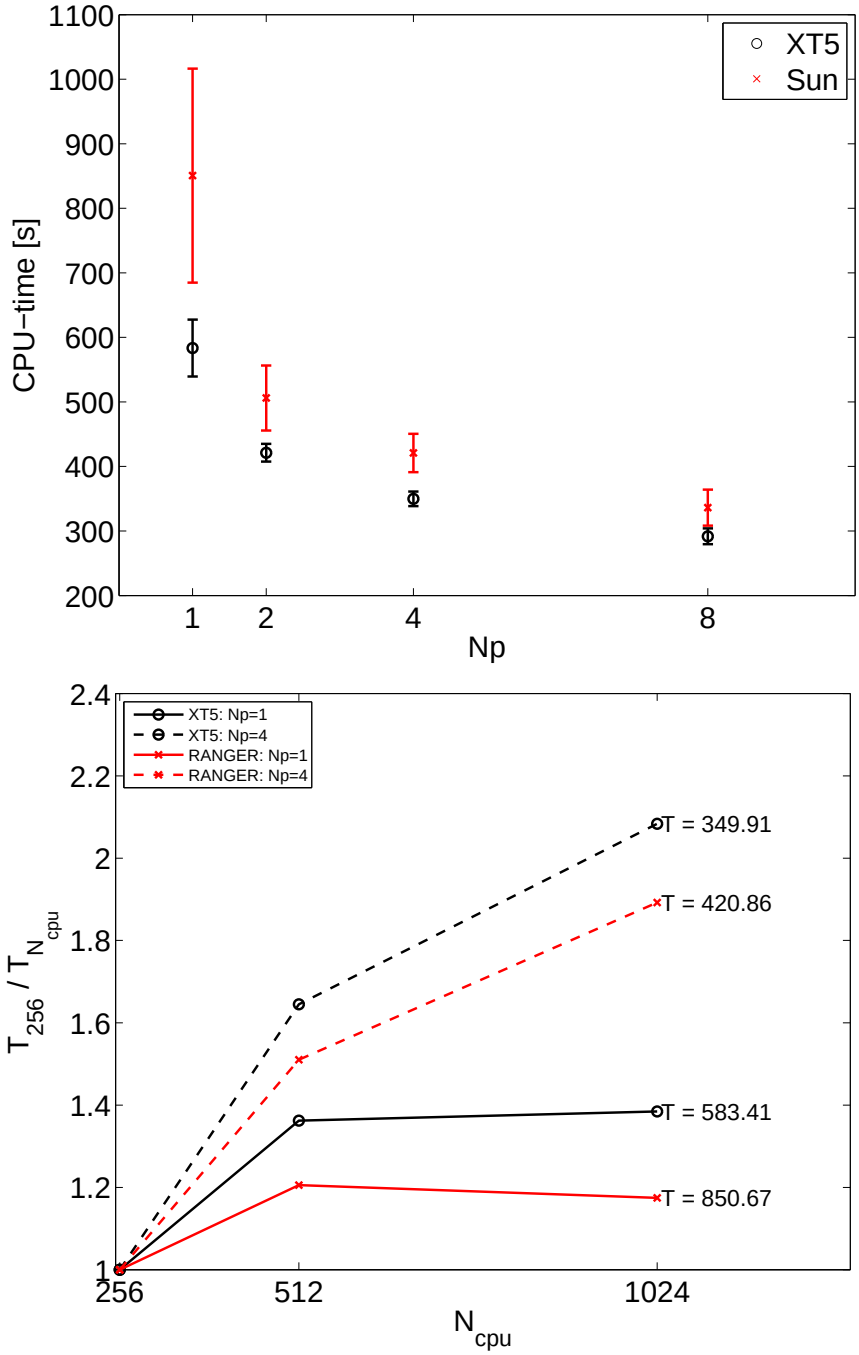


Figure 4.48: Flow simulations in a domain of figure 4.47. Top: simulations with 1, 2, 4 and 8 patches with 1024, 512, 256 and 128 cores per patch, respectively. Mean CPU-time and standard deviation, CPU-time - time required for 1000 time-steps, Np - number of patches, N_{CPU} - number of cores. The measurements are based on 10 simulations on each computer and for each coarse discretization. Bottom: Strong scaling in simulations with 1 and 4 patches, computing cores are equally subdivided between the patches. $Re = 470$, $P = 4$, $\Delta t = 0.002$. Simulation is performed on CRAY XT5 (Kraken, NICS), and Sun Constellation Linux Cluster (Ranger, TACC).

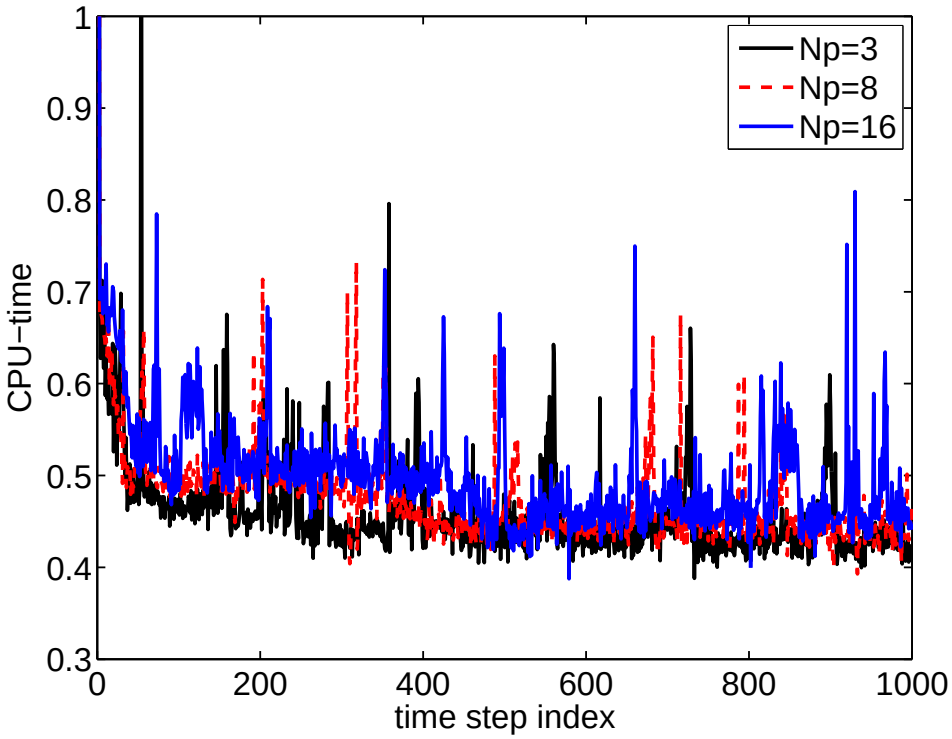


Figure 4.49: Flow simulations in a domain of figure 4.47: weak scaling. Simulation with 3 (A-C), 8 (A-H) and 16 patches (A-P) using 2048 cores per patch. Np - number of patches. $Re = 470$, $P = 10$, $\Delta t = 0.0005$. Simulation is performed on 6,144, 16,384 and 32,768 cores of CRAY XT5 (NICS).

Accuracy verification

To evaluate the accuracy of the two-level domain decomposition with overlapping patches, we first consider unsteady simulation in a domain of convergent pipe, which we used in the previous section, but this time the domain is discretized (on the outer level) into two overlapping patches as presented in figure 4.50. In figure 4.51 we plot the difference between the pressure computed with 2DD and 1DD. We observe that decoupling the pressure and velocity interfaces reduces the error associated with the IPC (compare results of figure 4.44(d) and 4.51(c,d)).

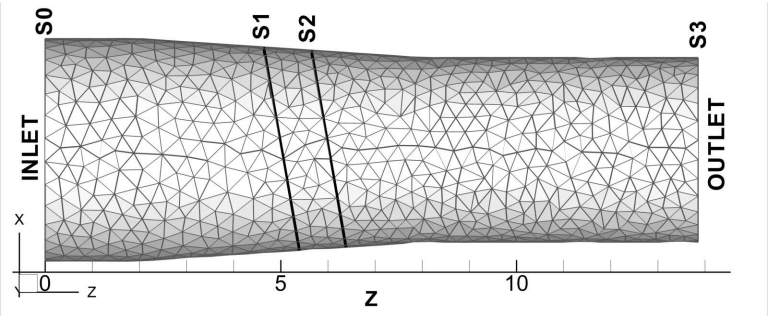


Figure 4.50: Domain of converging pipe: computational mesh and inter-patch interfaces ($S_1,\ S_2$). S_0 (S_2) - inlet (outlet) of sub-domain Ω_A, S_1 (S_3) - inlet (outlet) of sub-domain Ω_B.

Next we perform steady flow simulations in a channel with backward facing step illustrated in figure 4.52. In figure 4.53 contours of the streamwise ($w-$) component of velocity vector are presented for a) simulation with 1DD; b,c) and d) simulation with 2DD and overlapping patches as specified in the captions of figure 4.53. Very good agreement between the simulation results is observed. In figure 4.54 we plot the pressure contours for the aforementioned three simulations. Pressure field presented in plot (b) slightly varies from the pressure presented in plots (a, c) and (d) (see pressure contour around $z = 9$). The larger deviation is due to insufficient resolution in imposing the interface boundary conditions; specifically, truncation error corresponding to the choice of $P_{VBC} = 3,\ P_{PBC} = 3$. In simulation with interface located between S_1 and S_5 the inter-patch interface is located in the region of relatively high velocity and pressure spatial gradients, while in the simulation corresponding to figures 4.54(c,d) the interface is located in the region where velocity and pressure are very smooth, consequently the truncation error in IPC is lower. In order to "zoom-in" into the differences between the velocity and pressure computed with 1DD and 2DD we extract data along a line $x = 1.25, y = 2.5$. The results are presented in figures

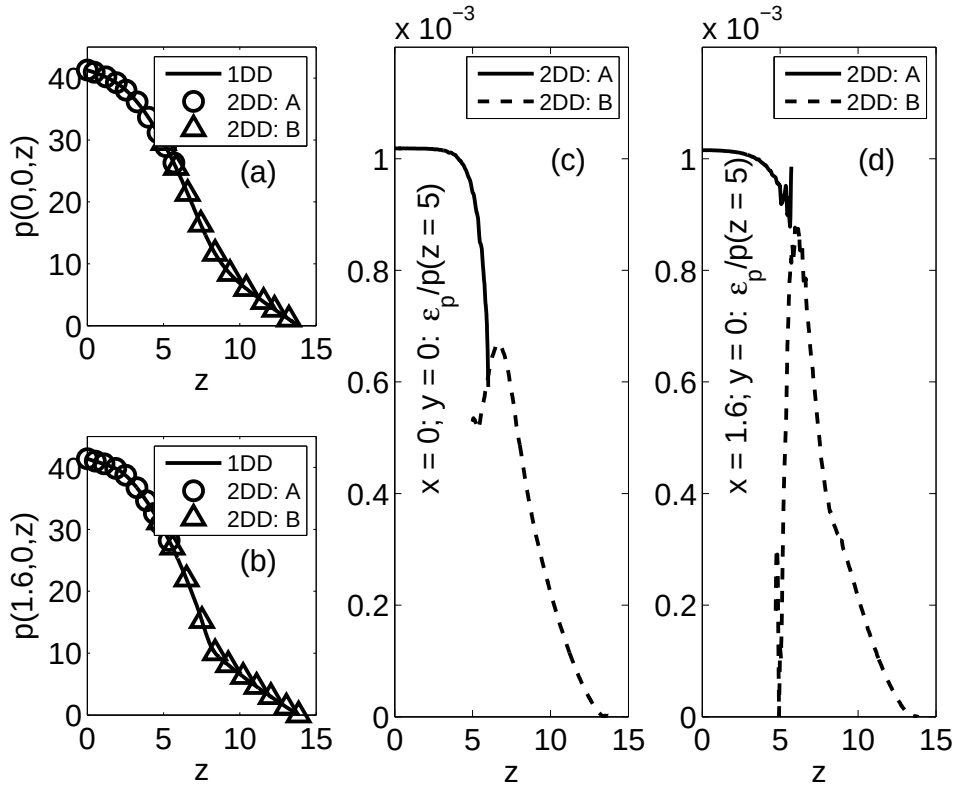


Figure 4.51: Unsteady flow simulation with 2DD and overlapping patches (see figure 4.50. Pressure along lines $y = 0$, $x = 0$ and $y = 0$, $x = 1.6$ as marked in figure (a) and (b) non-dimensional pressure values computed with 1DD and 2DD. (c) and (d) normalized absolute value of difference between the pressure computed with 2DD and 1DD: (c) - $y = 0$, $x = 0$; (d) - $y = 0$, $x = 1.6$. $P_{VBC} = 3$, $P_{PBC} = 2$, $P = 5$, $\Delta t = 0.0005$.

4.55 and 4.56. Figures 4.55(a) and 4.56(a) 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. In simulations corresponding to 4.55(b) and 4.56(b) the interface is located at the end and also outside the recirculation region, and as we observe it leads to about one order of magnitude lower difference between the data from 1DD and 2DD simulations.

Next, let us compare the computational efficiency in simulating the step-flow with 1DD and 2DD. In table 4.7 we summarize the results.

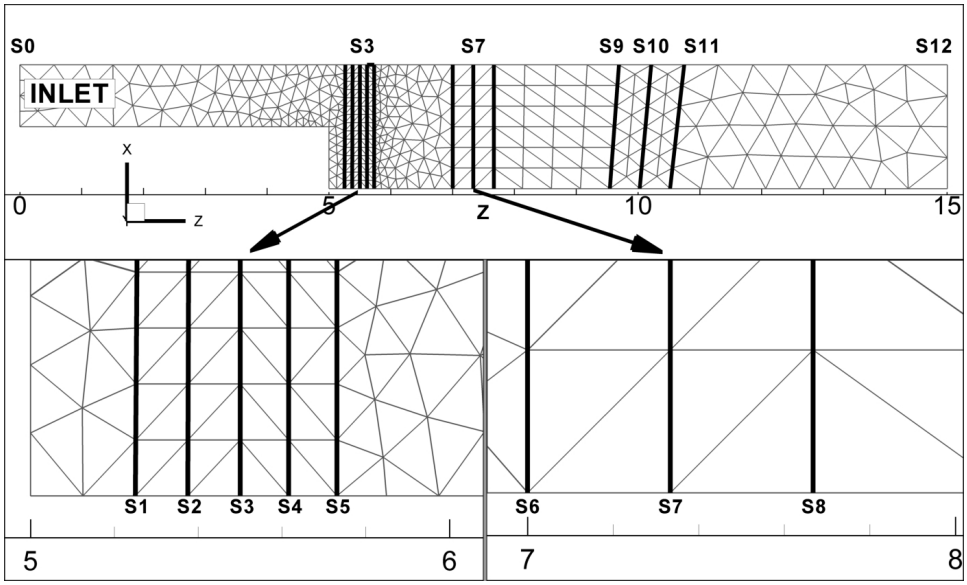


Figure 4.52: Domain of a 3D channel with backward facing step: computational mesh and inter-patch interfaces. S_0 ($S_1 2$) - inlet (outlet) of domain Ω. $S_j, j = 1, ..., 12$ - possible inter-patch interfaces. The width of the channel is 5 length units.

overlap	Nel	N_{CPU}	CPU-time $P = 3$	CPU-time $P = 5$
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

Table 4.7: Steady flow simulation in 3D channel with backward facing step: performance on CRAY XT5 (NICS). Nel - number of elements in two patches ($Nel(\Omega_A) + Nel(\Omega_B)$. N_{CPU} - number of processes assigned to patches. The first line correspond to 1DD simulation, and the next lines to simulations with 2DD and different overlapping regions.

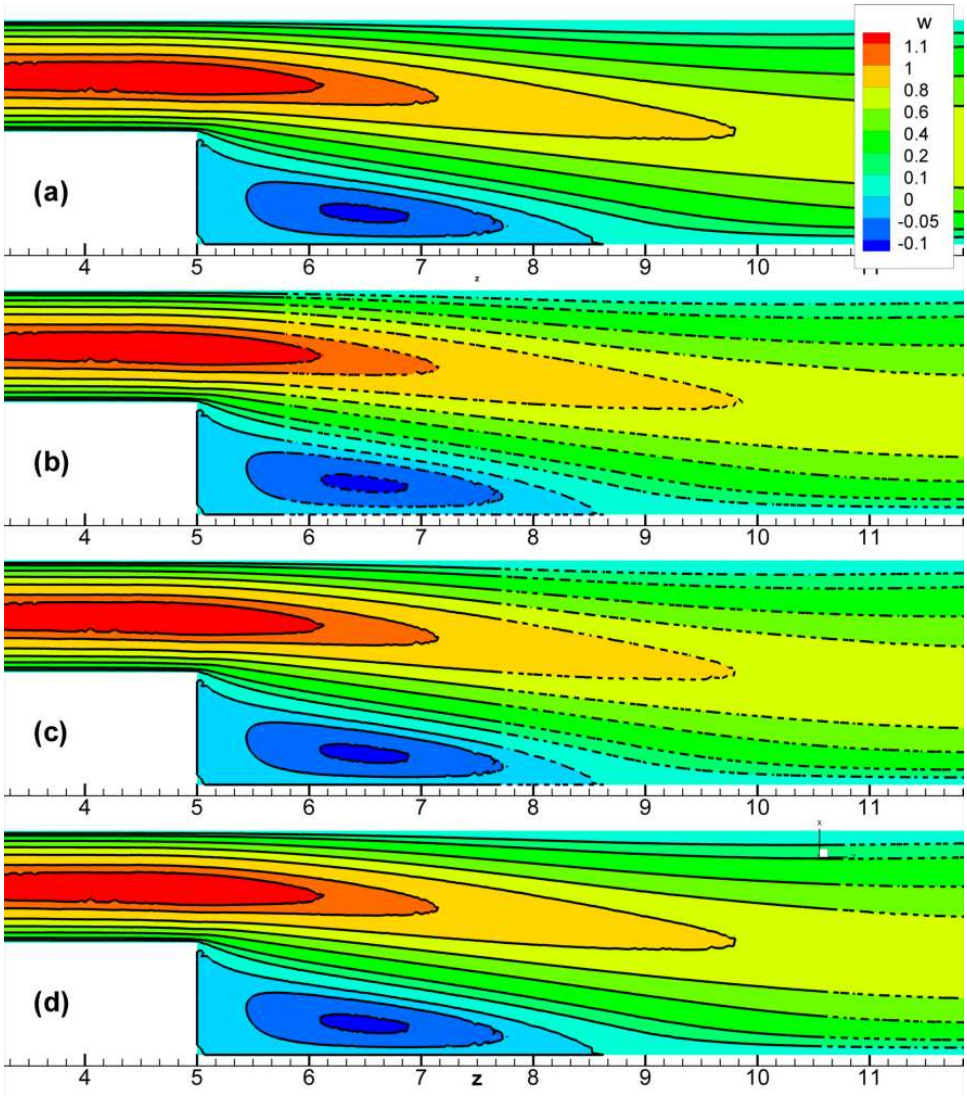


Figure 4.53: Steady flow simulation in 3D channel with backward facing step with 2DD and overlapping patches: contours of w-velocity components: $w(x, 2.5, z)$. a) 1DD; b) 2DD; patch Ω_A is located between S_0 and S_5, patch Ω_B is located between S_1 and S_12; c) 2DD; patch Ω_A is located between S_0 and S_8, patch Ω_B is located between S_6 and S_{12}; d) 2DD; patch Ω_A is located between S_0 and S_{11}, patch Ω_B is located between S_9 and S_{12}; Coarse discretization is illustrated in figure 4.52. $P = 5$, $P_{VBC} = 3$, $P_{PBC} = 3$, $\Delta t = 0.002$, $Re = 72$.

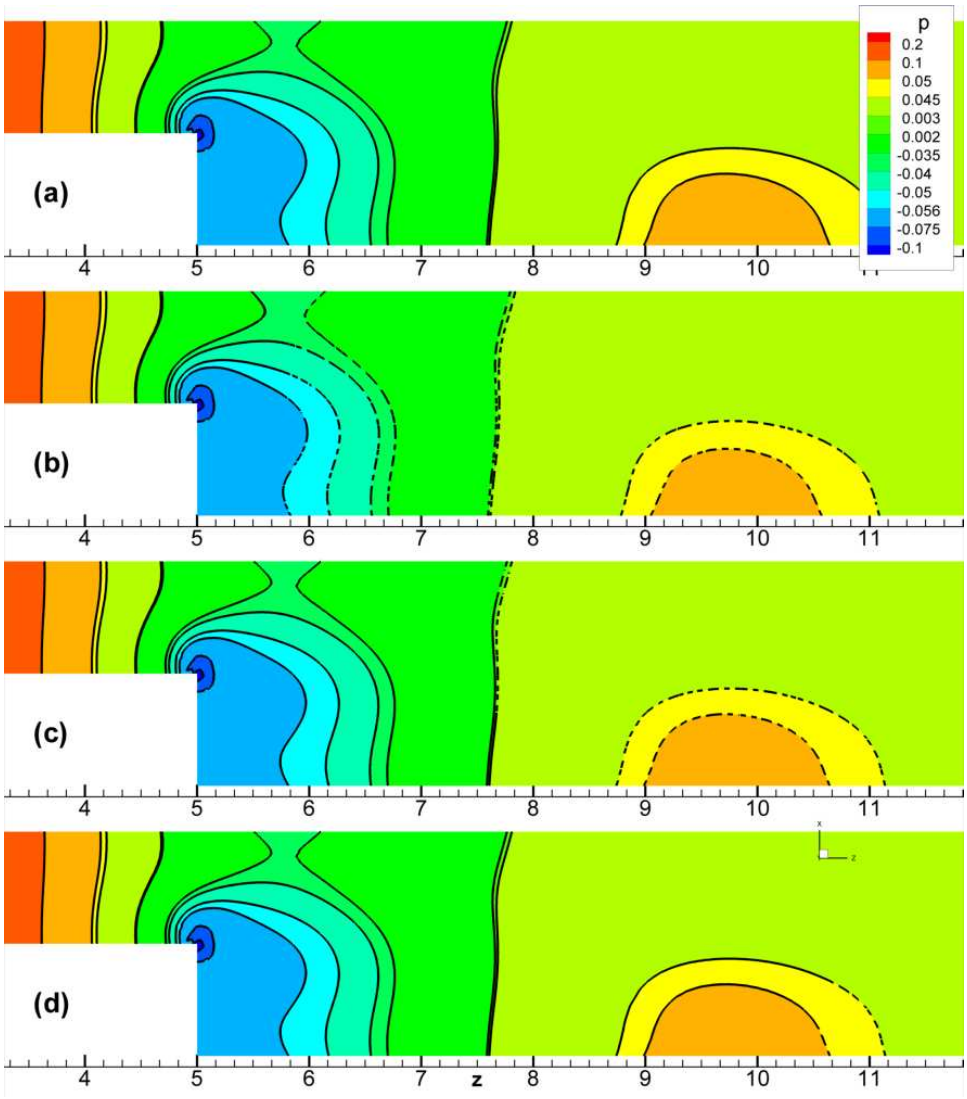


Figure 4.54: (in color) Steady flow simulation in 3D channel with backward facing step with 2DD and overlapping patches: pressure contours: $p(x, 2.5, z)$. a) 1DD; b) 2DD; patch Ω_A is located between S_0 and S_5, patch Ω_B is located between S_1 and S_{12}; c) 2DD; patch Ω_A is located between S_0 and S_8, patch Ω_B is located between S_6 and S_{12}; d) 2DD; patch Ω_A is located between S_0 and S_{11}, patch Ω_B is located between S_9 and S_{12} Coarse discretization is illustrated in figure 4.52. $P = 5,\ P_{VBC} = 3,\ P_{PBC} = 3,\ \Delta t = 0.002,\ Re = 72$.

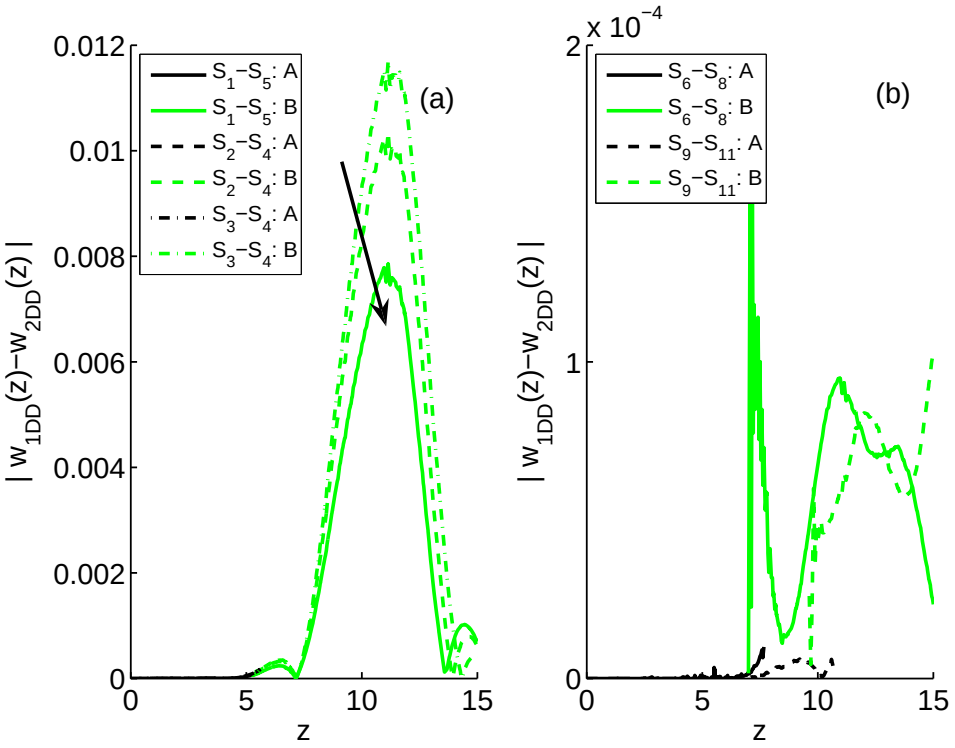


Figure 4.55: Steady flow simulation in 3D channel with backward facing step with 2DD and overlapping patches: difference in $w-$velocity component for various sizes and location of overlap. The data is extracted along $x = 1.25,\ y = 2.5$. a) overlapping region is located close to the step; solution is obtained to three different width of the overlap. b) overlapping region is located at the end of the recirculation region (solid line); and behind the recirculation region (dash line); (see figure 4.52). Arrow indicates increase in the overlapping. $P = 5,\ P_{VBC} = 3,\ P_{PBC} = 3,\ \Delta t = 0.002,\ Re = 72$.

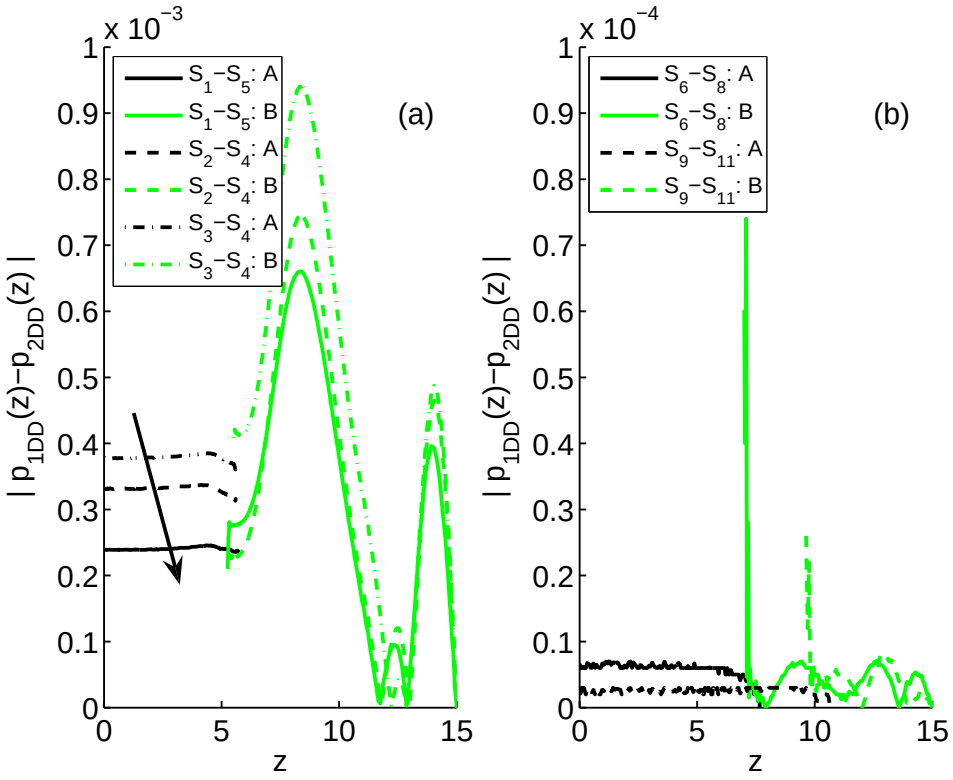


Figure 4.56: Steady flow simulation in 3D channel with backward facing step with 2DD and overlapping patches: difference in pressure for various sizes and location of overlap. The data is extracted along $x = 1.25,\ y = 2.5$. a) overlapping region is located close to the step; solution is obtained to three different width of the overlap. b) overlapping region is located at the end of the recirculation region (solid line); and behind the recirculation region (dash line); (see figure 4.52). Arrow indicates increase in the overlapping. $P = 5,\ P_{VBC} = 3,\ P_{PBC} = 3,\ \Delta t = 0.002,\ Re = 72$.

4.4.4 Arterial Flow simulations on TeraGrid: Results

In this section we present results of flow simulation with $\mathcal{NEKTARG}$ on TeraGrid - a supercluster of supercomputers geographically distributed across the USA. The main objective here is to show that high latency associated with long-distance communication can be efficiently hidden by overlapping computation and communication and also by performing full-duplex communication, and proper scheduling of communications.

Simulation of arterial flow in human aorta was performed. The computational domain was decomposed initially into four sub-domains as presented in figure 4.38(right), however only the first three sub-domains were used for the performance analysis (the sub-domain D was omitted). In space, fourth-order polynomial approximation for the velocity components and pressure was employed; in time, second-order semi-implicit numerical scheme was implemented. The size of computational sub-domains in terms of number of spectral elements (Nel) and degrees of freedom (DOF) is summarized in table 4.8. The computation

sub-domain	Nel	DOF
A	120,813	30,444,876
B	20,797	5,240,844
C	106,219	26,767,188
total	247,829	62,452,908

Table 4.8: Arterial flow simulation, numerical challenge: size of computational sub-domains. Nel - number of spectral elements; DOF - number of degrees of freedom per one variable.

was carried out on computers from the National Center for Supercomputing Applications (NCSA) and San Diego Supercomputer Center (SDSC) integrated by high-performance network connections. The application was executed on 225 and 128 processes of the NCSA and SDSC supercomputers, respectively. At NCSA two subjobs (S_1 and S_2) were submitted and only one subjob was submitted at SDSC. In table 4.9 computational details are summarized. Since the number of computational tasks exceeds the number of sub-jobs the S_1 group of processors was subdivided and two new communicators (T_1 and T_2) were derived. Processors grouped in communicators T_2, T_3 and T_4 were assigned for parallel solution of three 3D flow problems on the inner level. On the outer level, the boundary conditions at the sub-domain interfaces were imposed by exchanging values of the velocity and pressure fields. For the data exchange the ROOT of T_2 communicates with the ROOT of T_3 and the ROOT of T_3 communicates with the ROOT of T_4. The performance of the inter-site

sub-domain	site	N_{CPU}	S-group	T-group
O	NCSA	1	S_1	T_1
A	NCSA	192	S_1	T_2
B	NCSA	32	S_2	T_3
C	SDSC	128	S_3	T_4

Table 4.9: Arterial flow simulation on TeraGrid: Simulation performed on NCSA and SDSC supercomputers. The 3D computational domain is decomposed into 3 parts, parallel solution in each sub-domain was performed on N_{CPU} processes, additional processor is dedicated for co-processing. Second layer of MCI consists of three processor groups (S_j), third layer consists of four groups (T_j).

communication between the processes from T_3 and the T_4 was the primary point of interest in the performed simulation.

The schematic representation of the numerical algorithm implemented in $\mathcal{NEKTARG}$ is as follows:

```
for (time_step = 1:N)
MPI_Wtime(t1)
 1: MPI_Irecv(P,V,dV/dn)                    [L5]
 2: MPI_Gatherv(V), MPI_Isend(V)            [L4]
 3: Compute(dV/dn)                          [L4]
 4: MPI_Gatherv(dV/dn), MPI_Isend(dV/dn)    [L4,5]
 5: Compute flow rate and average
    pressure at all inlets/outlets          [L4,5]
 6: MPI_Wait(P), MPI_Scatterv(P)            [L5,4]
 7: MPI_Wait(V), MPI_Scatterv(V)            [L5,4]
 8: MPI_Wait(dV/dn), MPI_Scatterv(dV/dn)    [L5,4]
MPI_Wtime(t2)
 9:  Solve(NL)                              [L3]
 10: Solve(P)                               [L3]
 11: MPI_Gatherv(P), MPI_Isend(P)           [L4,5]
 12: Solve(V)                               [L3]
MPI_Wtime(t3)
endfor
```

In the square parenthesis we specify which layer(s) of the MCI is involved in communication

at each steps. In the communication between ROOTs of T_2 and T_3 the velocity values (V) are transfered from T_2 to T_3 while the pressure (P) and the velocity flux (dV/dn) are transfered in the opposite direction (see figure 4.34(right)). Similar data transfer takes place in communication between T_3 and T_4. Computational domains A and C are not physically connected thus no communication is required between processes in T_2 and T_4. During step 5, as presented in the numerical algorithm, flowrate values and average pressure are computed at the all inlets and outlets by processes of MCI level 4. The computed values are gathered in the ROOT of corresponding T_j communicator and then passed using MPI_Isend to the processor of T_1 via MCI level 5 communicator.
During steps 9, 10 and 12 the tightly coupled 3D problems are solved in parallel. The steps 10 and 12 are characterized by high volume of communication between processes of MCI level 3, no inter-site communication is required at these steps.

Simulations have been performed using two message passing interfaces designed for parallel computing on distributed processors: i) MPICH-g2, and ii) MPIg. Unlike MPICH-G2, MPIgs new multithreaded design allows applications executing on mulitcore systems, like those found on the TeraGrid, to effectively hide inter-site latency. In order to compare performance of $\mathcal{NEKTARG}$ using MPIg and MPICH-G2 libraries we monitored the CPU-time on the ROOT of T_j communicators. In figure 4.57 we compare the overall performance of $\mathcal{NEKTARG}$. The superiority of MPIg library over MPICH-G2 is clear. In table 4.10 we provide more details on the $\mathcal{NEKTARG}$ performance. We compare the CPU-time required to execute steps 1-8 ($t2 - t1$) and steps 9-12 ($t3 - t2$) monitored over the last 5 steps (the data monitored during the first 195 steps is similar). The data leads to the following observations:

- The $t3 - t2$ time required for solution of a problem in sub-domain C is considerably higher then the time required for solution of the problems in the sub-domains A and B. The difference is explained by different size of computational sub-domains and also by different number of iterations required at steps 10 and 12 where Poisson problem for the pressure and Helmholtz problem for the three velocity components are solved. Better load balancing can be achieved by increasing the number of processors in the T_4 communicator.

- The $t2 - t1$ CPU-time consumed by steps 1-8 and monitored on T_2 and T_3 is con-

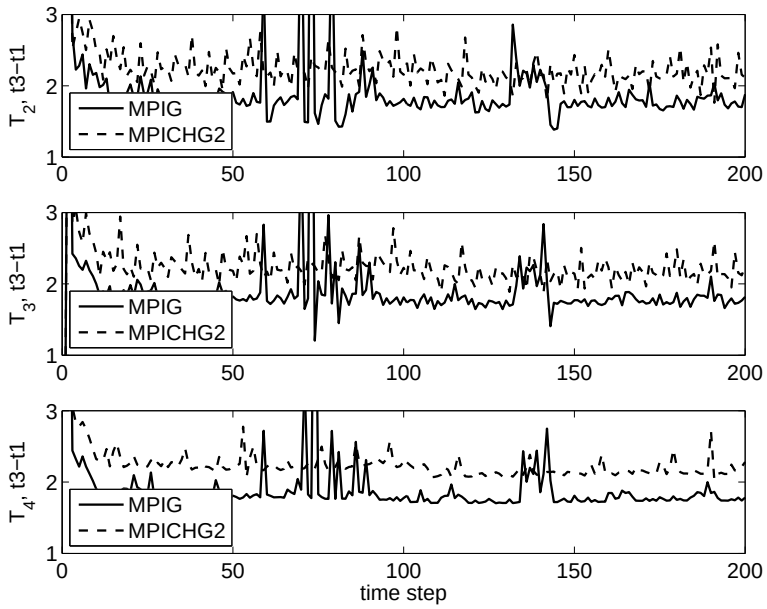


Figure 4.57: Arterial flow simulation, performance: Cross-site simulation using MPIg and MPICH-G2 libraries. CPU-time (in sec.) required for solution of a flow equation in each sub-domain.

siderably higher then the CPU-time monitored on T_4. This is a consequence of two factors:

(i) the velocity flux dV/dn, is transfered from the sub-domain C to B (and from B to A) but not vise versa. The message length for dV/dn is also the largest comparing to the other messages. The CPU-time (in seconds) required for steps 6, 7 and 8 monitored on the ROOT of T_3 for the last 5 time steps is as follows:

```
        MPICH-G2:                          MPIg
 STEP 6    STEP 7   STEP 8    |   STEP 6   STEP 7   STEP 8
2.31E-4;  2.87E-4;  0.8668;    |   9.3E-5;  4E-6;    0.3545;
2.32E-4;  2.95E-4;  1.3460;    |   9.1E-5;  3E-6;    0.3984;
2.36E-4;  2.99E-4;  1.0580;    |   9.6E-5;  4E-6;    0.3409;
2.31E-4;  2.91E-4;  1.0599;    |   9.2E-5;  3E-6;    0.3659;
2.30E-4;  3.42E-4;  1.1591;    |   9.1E-5;  4E-6;    0.4322;
```

that is the time $(t2 - t1)$ is almost entirely spent on the step 8. Also the considerable difference between performance of the code linked to MPIg and MPICH-G2 libraries is observed.

(ii) aforementioned load imbalance: processors of T_3 wait for data (dV/dn) that must be sent from T_4. Again, by minimizing the $(t3 - t2)$ time on the T4 sub-group we effectively can reduce the waiting time spent by processes of T_3 (and consequently by T_2). However, processes of T_4 have zero waiting time for messages received from T_3, thus measured $(t2-t1)$ time difference reflects the true cost of inte-rsite communication plus the cost of additional minor computational effort required for processing the received messages (including MPI_Scatterv, executed on withing an inlet group of MCI L4) and on preparing the outgoing messages (including MPI_Gatherv executed on MCI L4).

- On the step 11 of the numerical algorithm the values of pressure are transfered from T_4 to T_3 over a wide area network with MPI_Isend. Sending data on step 11 through the fifth level communicator of MCI overlaps with computation and communication (step 12) performed on the third level of the MCI. The difference in $(t3-t2)$ measured on the ROOT of T_4 which communicate with the ROOT of T_3 reflects the superiority of the multithreaded MPIg library over MPICH-G2.

In summary, distributed computing opens an opportunity to extend the range of large mechanical simulations. The major obstacle in solution of coupled problems on distributed computers is high latency cost associated with inter-site communication. Use of multi-threaded libraries for message passing for non-blocking communication, such as MPIg, provides a true overlap between communication and computation.

T_2

time step	MPIg $t2-t1$	MPICH-G2 $t2-t1$	MPIg $t3-t2$	MPICH-G2 $t3-t2$
196	0.388	0.533	1.364	1.575
197	0.432	0.375	1.347	1.590
198	0.374	0.857	1.334	1.606
199	0.399	0.549	1.300	1.606
200	0.466	0.514	1.416	1.599

T_3

time step	MPIg $t2-t1$	MPICH-G2 $t2-t1$	MPIg $t3-t2$	MPICH-G2 $t3-t2$
196	0.707	0.890	1.089	1.060
197	0.670	1.369	1.051	1.078
198	0.693	1.081	1.040	1.073
199	0.742	1.083	1.025	1.037
200	0.716	1.182	1.101	1.059

T_4

time step	MPIg $t2-t1$	MPICH-G2 $t2-t1$	MPIg $t3-t2$	MPICH-G2 $t3-t2$
196	0.022	0.123	1.737	2.123
197	0.022	0.122	1.723	2.040
198	0.022	0.122	1.761	2.035
199	0.022	0.122	1.719	2.092
200	0.022	0.122	1.760	2.159

Table 4.10: Arterial flow simulation on TeraGrid: Simulation performed on NCSA and SDSC supercomputers. The 3D computational domain is decomposed into 3 parts, parallel solution in each sub-domain was performed on N_{CPU} processors, additional processor is dedicated for co-processing. Second layer of MCI consists of three processor groups (S_j), third layer consists of four groups (T_j).

4.5 Blood flow circulation in Circle of Willis

The dynamics of blood flow in the human brain depends upon a complex network of vessels under a variety of temporal and spatial constraints. Abnormalities in the delivery of blood to the brain clearly underlie the pathophysiology of stroke, vasospasm, traumatic brain injury, vascular dementias, and probably conditions such as migraine and hydrocephalus. Clinical decisions are often made on the basis of steady state conditions (e.g., mean intracranial pressures, mean cerebral blood flow, etc), but there is clearly a risk that ignoring the range of spatial and temporal scales present may limit understanding, and hence clinical effectiveness. Considerable interest attends the recent observation that a frequency dependent pulsation absorber may be a part of normal physiology, dampening the effect of cardiac pulsation on microvessels (Zou *et al.* 2008). A detailed understanding of such a phenomenon categorically depends on development of new techniques to grasp the importance of dynamic processes at a variety of different scales.

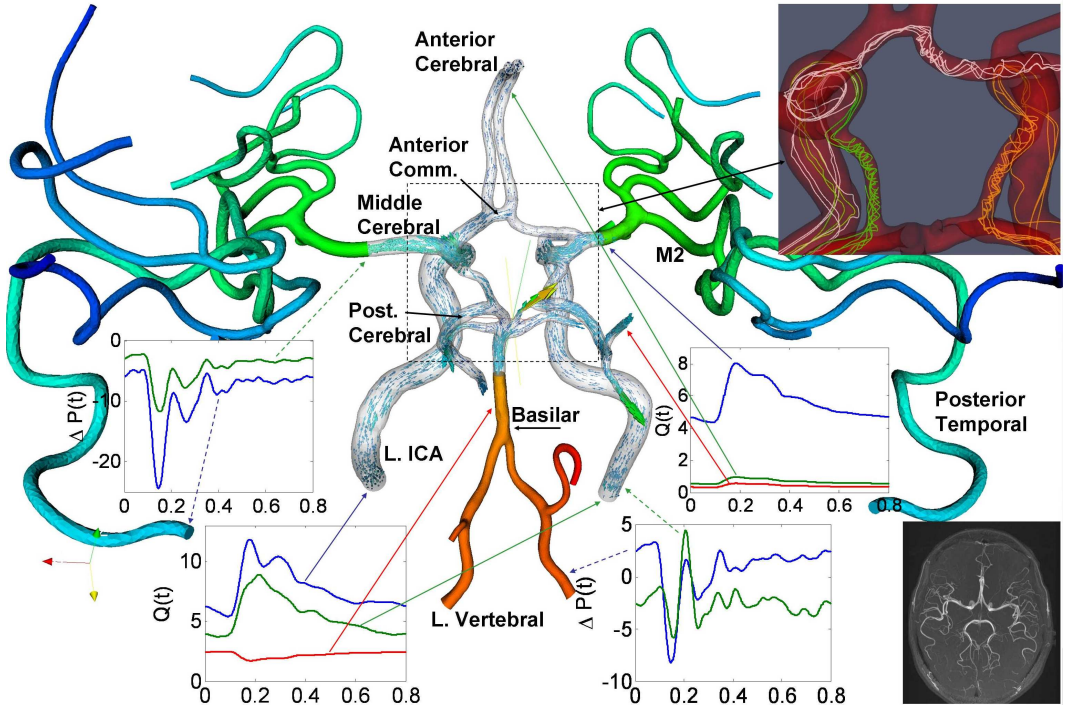


Figure 4.58: (in color) Brain blood flow simulation in complete Circle of Willis: Geometrical model of 65 cranial arteries. Colors represent pressure, arrows represent velocity fields, XY plots depict the flow rate in ml/s and pressure drop $\Delta P = P - P_{ref}$ in $mmHg$, where P_{ref} is the average pressure at ICA inlet. Top right: instantaneous streamlines showing swirling flow in communicating arteries. Bottom left: MRA image of the cranial arterial system.

The range of length and times scales which need be considered spans several orders of magnitude, from flow in large arteries to subcellular processes. Interactions of blood flow in the human body occur between different scales, in which the large-scale flow features are being coupled to cellular and sub-cellular biology, or at similar scales in different regions of the vascular system. The human brain is a fascinating object of study in this regard since 20% of cardiac output must be distributed to tissues with exquisite regulation from a pulsatile cardiac pump, with the unique boundary condition that the intracranial volume is bounded by the skull, a rigid exoskeleton. The typical anatomy includes delivery of blood by two internal carotid (ICA) and two vertebral arteries, which branch and link to form a ring - the Circle of Willis (CoW, presented in figure 4.58), which can potentially provide alternative supply to any area of the brain if one or more of the supply trunks is interrupted. Abnormalities in the CoW are not uncommon, affecting up to 50% of the population, according to Lippert & Pabst (1985). However, the effectiveness of the normal CoW arrangement, or the consequence of an abnormality, relies on downstream responses, and thus critically on interaction between events observable on different scales. Development of tools to simulate the events *simultaneously at many scales* becomes critical to understanding, and perhaps clinical effectiveness. We have focused on hydrocephalus, where interactions between pressures and flows in large compartments (the ventricles) may have a clinical manifestation through changes in flow pattern in microvessels (Zou *et al.* 2008).

We can begin to probe differences between dynamics in the normal state and hydrocephalus by looking at normal versus abnormal flows in an unusual subject with hydrocephalus and a CoW abnormality. The goal here is to get the first insight into brain flow patterns, understand the capabilities of different arterial flow models. Application of these techniques to more subtle abnormalities may clarify the true dynamic abnormality in hydrocephalus, and perhaps many other vascular conditions where changes in the topology and geometry of the vascular tree may directly impact risk of later severe clinical events such as ischemic stroke and hemorrhage. Examples include abnormal arteriovenous fistulae and complex connections such as arteriovenous malformations (Friedlander 2007), atherosclerotic narrowing of vessels (Amin-Hanjani *et al.* 2005), and moyamoya syndrome (Scott *et al.* 2004; Smith & Scott 2005), where a consequence of large vessel occlusion in childhood results in development of a dense network of microvessels to replace the function of

an occluded part of the vascular tree. Therapeutic goals in these cases involve occluding abnormal channels, stenting partially occluded channels, and providing alternative routes for blood to revascularize the tissue, respectively. Planning for pharmacological, surgical or endovascular treatment of these abnormalities of blood vessels could all benefit from insights derived from realistic models of the circulatory dynamics in the diseased arterial network. Preliminary work on the use of quantitative MR angiography to predict stroke appears promising (Amin-Hanjani *et al.* 2005).

Full 3D simulation of the intracranial flow and its interaction with the brain tissues at *all* length scales is not feasible, and hence we should follow the MaN-MeN-MiNapproach described in section 4.1. In this work we concentrate on blood flow simulations at the MaNscale, and present results of 1D simulations and of high-resolution 3D blood flow simulations in CoW. High resolution 3D flow simulation in tens of brain arteries and bifurcations present a significant challenge from the standpoint of parallel computing and numerics, hence we shell employ all advanced methods described in the previous sections. We use patient-specific geometrical models of the arterial networks, measured *in-vivo* flowrates for the inlet boundary conditions and the RC-type outflow boundary condition as a closure problem for the 3D simulation. We also make use of the described in section 4.4.1 two-level domain decomposition and multilevel communicating interface.

In this section we consider intracranial flow simulation based on clinical data corresponding to the following cases:
(a) healthy subject with complete CoW,
(b) patient with hydrocephalus and incomplete CoW (Patient No. 1), and
(c) patient with hydrocephalus and incomplete CoW (Patient No. 2).

The geometry of complete CoW of a healthy subject has been reconstructed from high-resolution DSCTA and MRI images, and geometry of other two arterial trees was obtained from MRI images. To reconstruct the arterial geometry we used "in-house" developed software gOREK, presented in section 4.2. The intracranial arterial trees corresponding to the three aforementioned cases are presented in figure 4.59. Following the two-level domain decomposition strategy, we subdivide the large computational domains into 2 to 4 patches as also illustrated in 4.59 by coloring each patch into a different color. The summary on spectral discretization is provided in table 4.11. Simulations of the unsteady 3D flow

have been done on Sun Constellation Linux Cluster (Ranger, TACC) and CRAY XT4 (Kraken, NICS). The number of computer processes (cores) used for the three cases was 3344, 1040 and 1404 correspondingly. The simulations have been performed with second-order semi-implicit time splitting scheme and nondimensional time step was in the range $\Delta t = [1E-3 \;\; 5E-5]$, the small size of a time-step was required during the systolic peak in order to satisfy the CFL conditions. The approximate simulation time for one cardiac cycle was 27 hours.

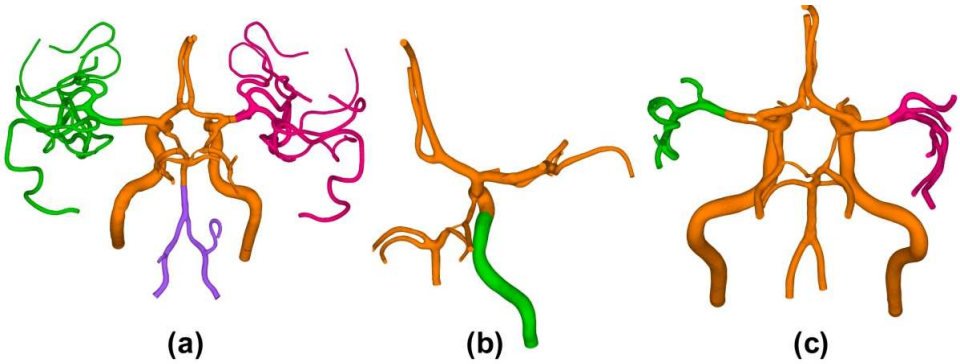


Figure 4.59: (in clolor) Domain of cranial arteries of: a) healthy subject with complete CoW, b) a patient with hydrocephalus and incomplete CoW (Patient No. 1, only the right part of CoW is shown), and (c) a patient with hydrocephalus and complete CoW (Patient No. 2). The geometry obtained from DSCTA and MRI images; colors represent different patches.

case	patch	orange	blue	green	pink	total
(a)	Nel	162,909	44,632	128,508	123,201	**459,250**
(a)	# DOF (P=5)	63,860,328	17,495,744	50,375,136	48,294,792	**180,026,000**
(b)	Nel	112,495		19,161		**131,656**
(b)	# DOF (P=4)	28,348,740		4,828,572		**33,177,312**
(c)	Nel	149,411		26,432	31,048	**206,891**
(c)	# DOF (P=4)	37,651,572		6,660,864	7,824,096	**52,136,532**

Table 4.11: CoW simulation: computational complexity. Nel - number of spectral elements. DOF - number of degrees of freedom per one variables, computed as a total number of quadrature points: $DOF = Nel * (P+3)(P+2)^2$ required for exact integration of linear terms.

4.5.1 1D and 3D arterial flow simulations

The 3D simulations provide deep insight into flow patterns, Wall Shear Stress, and may be used for particle tracking instead of dye angiograms. However, sometimes the only informations required is the direction of the mean flow, arterial pressure drops and flow distribution. Such requirements might be fulfilled by performing computationally inexpensive one dimensional simulations. The 1D model has gained popularity owing to its ability to predict, with reasonable accuracy, pulse wave propagation. The 1D model is computationally inexpensive and can be applied successfully in simulations where blood flow does not exhibit strong 3D effects (e.g. in straight vessels with a diameter $< 2mm$). It can also be applied to simulate the pulse wave propagation in large vessels; however, only large-scale features can be predicted. However, no information on the local flow dynamics can be obtained using the 1D model, for example it is impossible to simulate a flow in aneurysms using the 1D model. Overviews and examples of 1D modeling of blood flow are available in the literature [81, 13, 100]. The 1D model, considered here, incorporates elastic properties of the arterial walls; a recent study accounts also for the variability of the elastic properties in the 1D model using stochastic modeling [104]. The flow equations considered by the 1D model are:

$$\frac{\partial A}{\partial t} + \frac{\partial AU}{\partial x} = 0$$

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial x} = \frac{f}{\rho A},$$

where, x is the local to each arterial segment 1D coordinate axis, $A(t,x)$ is the cross-section area, $U(t,x)$ and $p(x,t)$ are the averaged over cross-section axial velocity and pressure, ρ is the blood density, and $f = -2\mu\pi\alpha/(\alpha-1)U$ is the friction force per unit length, μ is the blood viscousity and α is the nondimensional correction factor, that depends on the assumed velocity profile. In our study we used $\alpha = 1.1$ we have also verified that the sensitivity of $U(t,x), p(t,x)$ and $A(t,x)$ on α is very low. The closure for the system is provided by the pressure-area relation:

$$p = \frac{\beta}{A_0}\left(\sqrt{A} - \sqrt{A_0}\right), \quad \beta = \beta_0\frac{\sqrt{\pi}hE}{1-\sigma^2},$$

where h is the arterial wall thickness, E is the Young's modules, A_0 is the reference area, and β_0 is a scaling parameter. The physiological parameters in this study were chosen

according to data provided in [13].

In the following we compare the predictive capabilities of the 1D and 3D models. We use patient-specific geometry and flowrates measured (with PC-MRI) at internal carotid and vertebral arteries. The geometry and the blood flowrates at internal carotid and vertebral arteries have been clinically acquired in the Children's Hospital in Boston, department of neurosurgery.

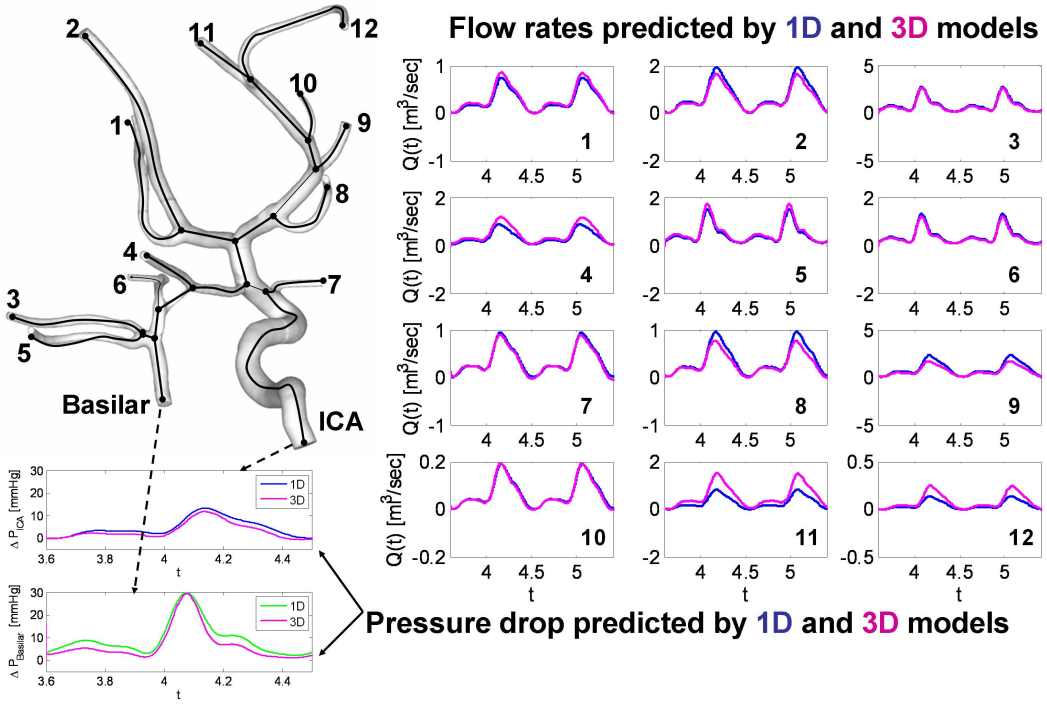


Figure 4.60: (in color) Brain blood flow simulation in incomplete CoW with one- and three-dimensional models. (a) Arterial geometry, (b) flow rates and (c) pressure drop at (i) ICA and (ii) basilar artery.

For the first experiment we chose a relatively small part of arterial network of patient with multiple brain disorder (patient No. 1) who also has an incomplete CoW. The geometry of the part of CoW we have used for simulations is presented in figure 4.60. To compare results of 1D and 3D simulations we performed skeletonization of the model shown in figure 4.60; then, setting pressure at all outlets to $p = 0$, using sufficiently large value of β and imposing the same *patient specific* flowrates at the two inlets, we performed 1D and 3D simulations. The large β was required to stiffen the arterial walls. The zero pressure boundary conditions and stiff arterial walls were rather simplifications of the model required for the preliminary comparative analysis of the 1D and 3D models. The 1D simulation over

eight cardiac cycles took only 20 minutes on a single processor, while about 24 hours/per cycle were required for the 3D simulation on 256 cores of CRAY XT4. In figure 4.60 we compare the pressure drop between the terminal branches and the two inlets and the flowrates computed at terminal branches using the 3D and the 1D models; the similarity in the waveforms is remarkable in most of the arterial domain. The discrepancies between the flowrates predicted by the 1D and 3D, usually appear at the vessels located farther from the inlets (outlets 1, 2, 11 and 12 in figure 4.60). The primary reasons for such deviation are: a) the 1D model does not take into account the angle at which a particular vessel bifurcates; and b) the shape of the velocity profile at any point of the arterial tree is assumed to be the same, which introduces errors in modeling the wall friction in arteries with swirling or reversal flow.

In the first experiment we used a rigid arterial wall models for both the 1D and 3D simulations. In the next example we employ the same 3D flow model but now the 1D model predicts also the elastic behavior of the blood vessels. We compare the results predicted by the 3D and 1D simulation with various levels of elasticity ($\beta_0 = 1,\ 8$). The purpose of this study is to estimate the deviation in prediction of flowrates and pressure drop by the rigid 3D and *elastic* 1D models. To this end, we reconstruct from (MRI images) the geometry of cranial arterial vasculature of a patient treated for hydrocephalus(patient No. 2). This patient has a complete CoW. The flowrates at the inlets have been acquired using PC-MRI technique. The 3D model and flowrates measured at the internal carotid and vertebral arteries are presented in figure 4.61. To perform the 1D simulation we measure the length and the average area of each arterial segment using the reconstructed 3D geometry and construct the corresponding 1D arterial tree. To impose outlet boundary conditions we employ the RC boundary conditions for the 3D simulations and the RCR model for the 1D simulation. The parameters of the RCR model have been adjusted similarly to the method used in section 4.3.1 (also see captions to figure 4.11). In table 4.12 we summarize the parameters for the 1D and 3D simulations.

To examine the effect of stiffening the arterial walls we perform 1D simulations with different β_0 parameters: $\beta_0 = 1$ and $\beta_0 = 8$. In figure 4.62 we compare the flowrates and pressure difference predicted by the 3D and 1D simulations. Results of the 3D simulation and of the 1D simulation with $\beta_0 = 1$ reveal significant discrepancy in the amplitude of the flowrate and pressure oscillations during a cardiac cycle, although, the two simulations

showed remarkable agreement in predicting the time-average flowrate and pressure drop. In 1D simulation with relatively stiff arterial walls ($\beta_0 = 8$) convergence of the data predicted by 1D model to data predicted by the 3D model is observed. Clearly, the discrepancy in the results is due to different *modeling* approaches, specifically, modeling of interaction between the *elastic* arterial walls and flow. Use of stiff arterial walls for simulation can be appropriate in some situations, for example it is well known that aging as well as some pathophysiological processes in the human body may lead to calcification of arterial walls. However, the large discrepancy in the 3D and 1D results suggest that elastic properties of the walls should not be ignored. In figure 4.63 we present the area fluctuations during one cardiac cycle. Note that in the case of $\beta_0 = 1$ the area fluctuations is less then two per cent, which is an acceptable range from the standpoint of physiology. Such small area fluctuations typically lead to conclusion that the wall flexibility can be ignored and the rigid wall model is sufficiently accurate. However, as we learn from the last numerical experiment such small *local* area fluctuations have significant contribution to the capacitance of the relatively large arterial network and long range effects on the flow and pressure waveforms in *distant* regions. Due to elasticity of the blood vessels the arterial system exhibits the Windkessel effect, where the blood, ejected by heart at the systolic phase, is partially stored in the vessels due to cross-sectional area (volume) expansion, during the diastolic phase the vessels contract slowly releasing the excess blood volume. It is clear that long elastic arterial segments and also networks of *multiple* elastic arterial segments have the capability of storing relatively large amount of blood, consequently reducing the amplitude of flow oscillations at the regions located farther from the heart. Reduction in the amplitude of mean flow translates to lower peak Reynolds number. From the standpoint of fluid dynamics, high Reynolds number flows typically trigger secondary flows and even transition to turbulence. This highlights the importance of choosing the correct and preferably patient-specific elasticity parameters for flow simulation in large arterial networks, and also an importance of modeling elastic behavior of the vessels even if the *local* area fluctuations are very small.

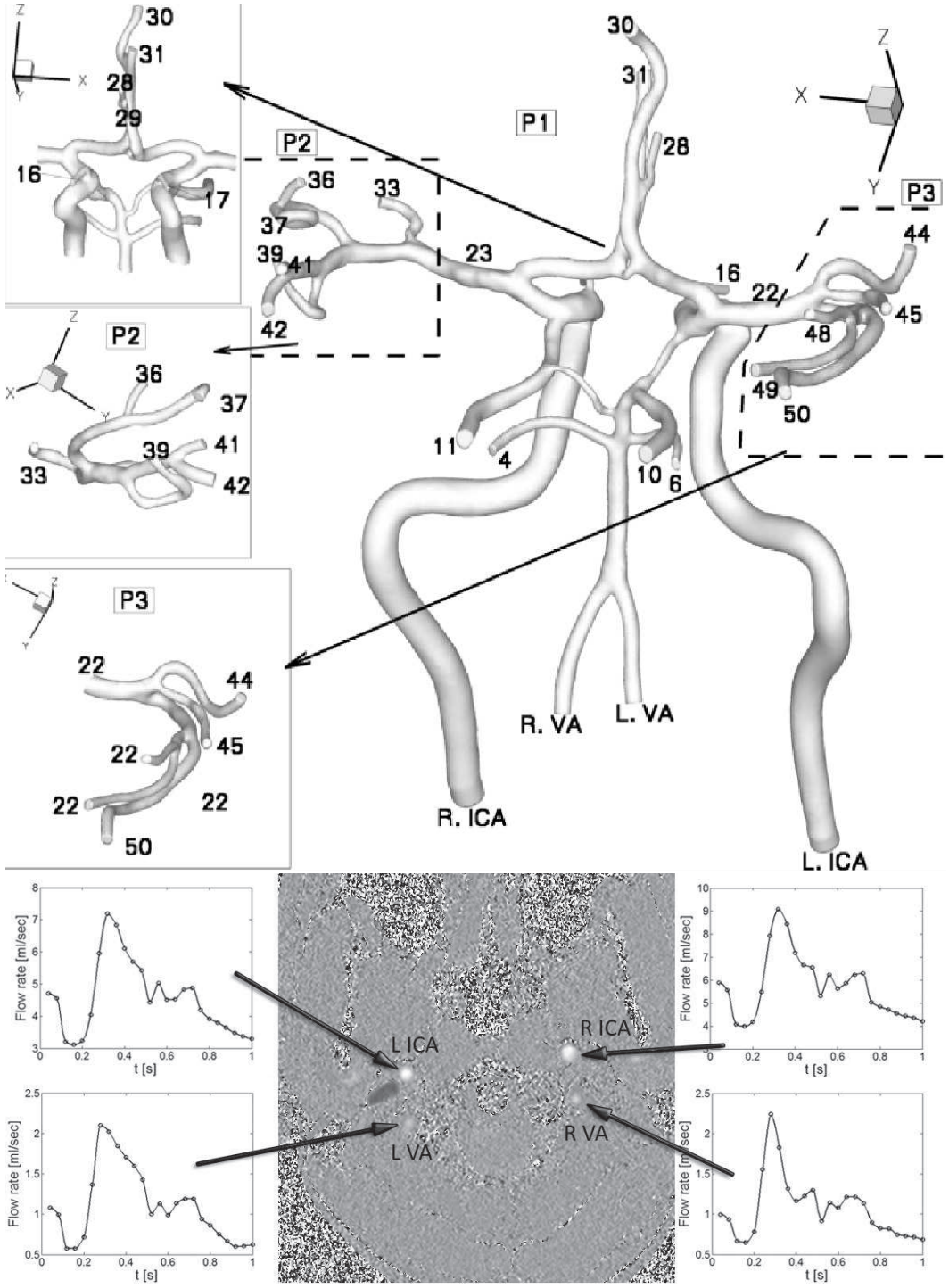


Figure 4.61: Brain blood flow simulation in complete CoW of a patient with hydrocephalus (patient No. 2). Arterial geometry and measured by PC-MRI flowrates at four inlets. Numbers correspond to IDs of the arterial segments. The arterial tree is subdivided into three patches - P1, P2 and P3.

segment	$A_o[10^{-6}m^2]$	$L[m]$	$\beta[10^6]$	$R[10^9 Pasm^{-3}]$	$C[10^{12}m^3Pa^{-1}]$	
1	5.77	0.148	111.03			
2	5.77	0.148	111.03			
3	3.37	0.0214	449.26			
4	3.46	0.086	286.57	36.8	5.43	
5	4.85	0.001	311.98			
6	3.46	0.086	286.57	36.8	5.43	
7	3.37	0.0066	449.26			
8	3.37	0.0045	303.25			
9	0.842	0.0124	606.5			
10	3.46	0.086	286.57	36.8	5.43	
11	3.46	0.086	286.57	36.8	5.43	
12	1.67	0.0151	412.19			
13	1.67	0.0151	412.19			
14	13.6	0.1703	69.674			
15	15.7	0.1703	60.278			
16	1.81	0.0071	541.79	73.6	2.72	
17	0.985	0.0055	997.89	73.6	2.72	
18	13.6	0.0067	69.55			
19	15.7	0.0067	60.508			
20	12.6	0.005	150.45			
21	12.6	0.005	150.45			
22	6.42	0.0144	210.42	36.8	5.43	
23	6.42	0.0131	210.42	36.8	5.43	
24	4.3	0.0117	257.18			
25	4.3	0.0117	257.18			
26	5.23	0.0114	188.05			
27	6.33	0.0117	155.2			
28	1.72	0.0093	417.61	36.8	5.43	
29	2.22	0.0059	324.1	36.8	5.43	
30	3.4	0.0249	289.33	36.8	5.43	
31	3.14	0.0161	312.94	36.8	5.43	
32	1.67	0.0032	412.19			
33	1.72	0.011	417.61	36.8	5.43	
34	5.9	0.008	166.73			
35	2.27	0.018	316.52			
36	1.63	0.005	441.14	36.8	5.43	
37	2.43	0.014	295.31	36.8	5.43	
38	5.9	0.008	166.73			
39	1.37	0.019	524.99	36.8	5.43	
40	3.53	0.008	278.51			
41	1.67	0.005	429.13	36.8	5.43	
42	3.33	0.007	294.97	36.8	5.43	
43	4.52	0.0022	217.32			
44	3.8	0.0221	258.63	36.8	5.43	
45	2.54	0.0171	282.33	36.8	5.43	
46	4.52	0.0154	217.32			
47	2.54	0.0092	282.33			
48	6.16	0.0123	159.66	36.8	5.43	
49	2.01	0.0243	357.32	36.8	5.43	
50	3.8	0.0329	258.63	36.8	5.43	

Table 4.12: Flow simulations in the arterial tree of patient No. 2: parameters and dimensions.

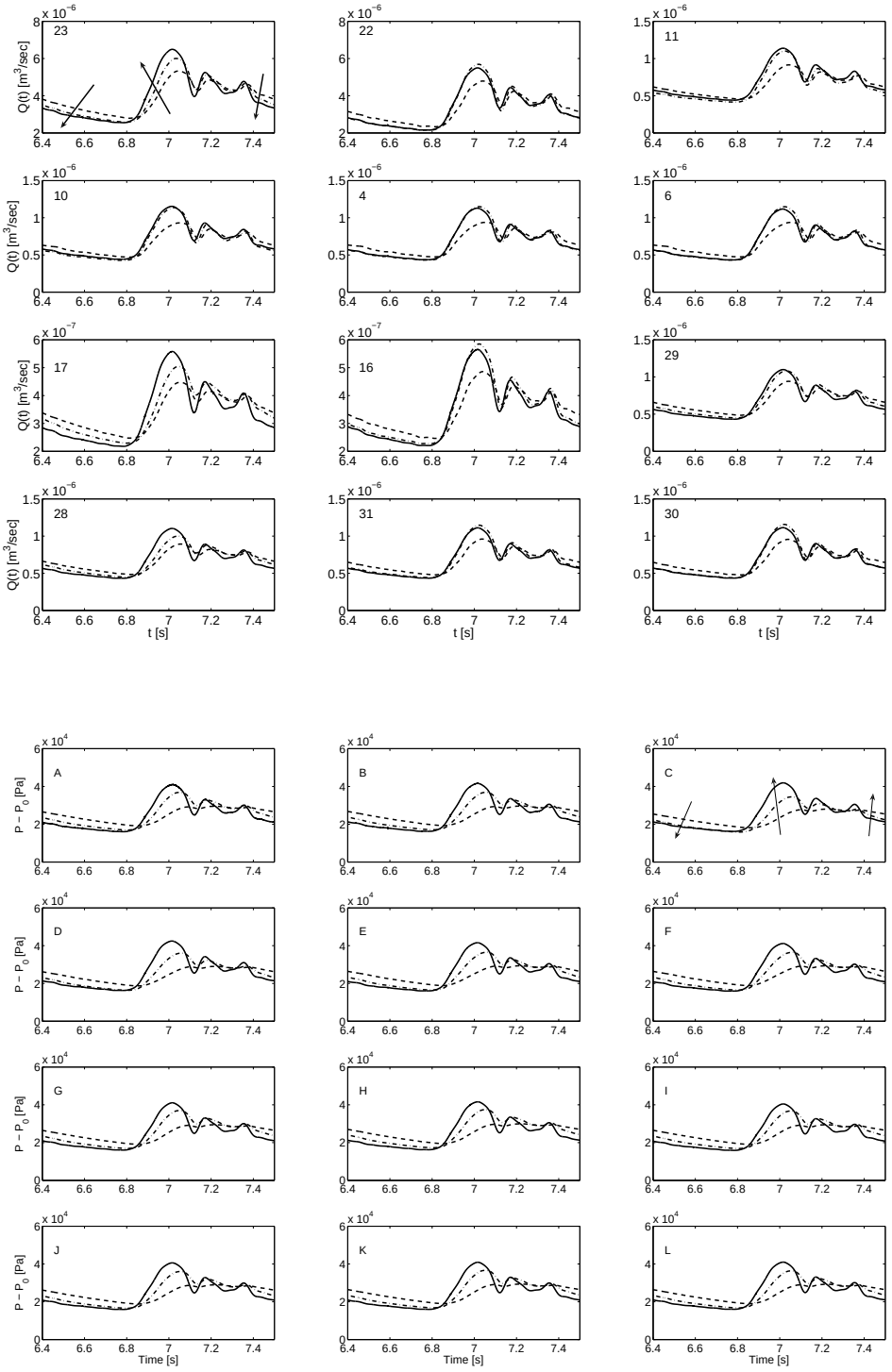


Figure 4.62: Brain blood flow simulation in complete CoW with one- and three-dimensional models: comparison of flowrates and pressure drop computed at different arteries. Input data corresponds to patient No. 2.

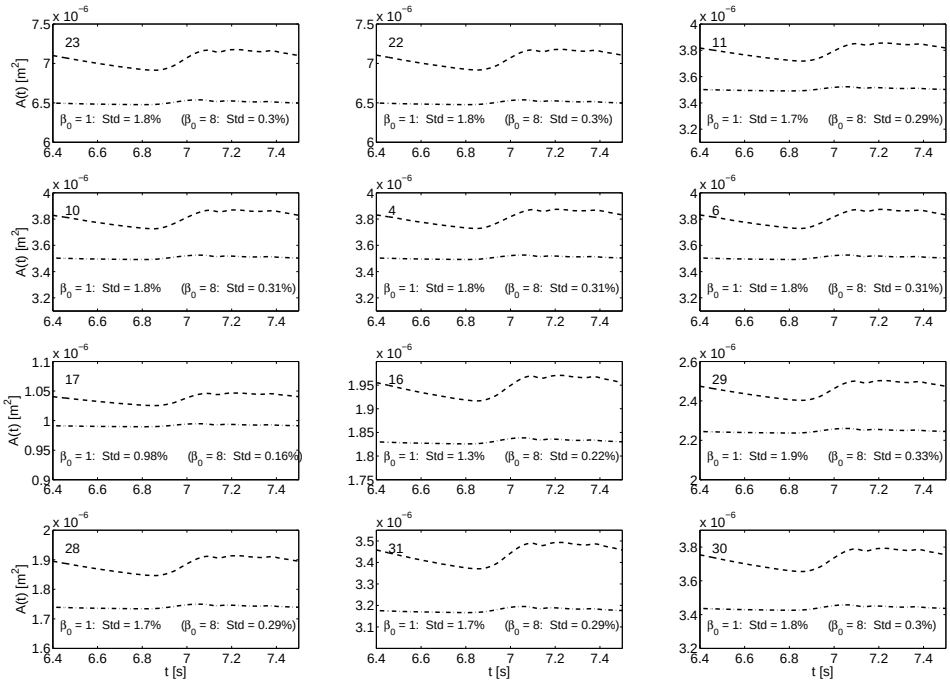


Figure 4.63: Brain blood flow simulation in complete CoW with one- and three-dimensional models: cross sectional area fluctuations. Dash line - corresponds to 1D simulation with elastic walls ($\beta_0 = 1$); dot-dash line - corresponds to 1D simulation with stiffer walls ($\beta_0 = 8$). "Std." denotes standard deviation. Input data corresponds to patient No. 2.

4.5.2 3D arterial flow simulations

We performed three 3D simulations of cranial blood flow using arterial geometry and inlet flow waves corresponding to the following cases: (1) a healthy subject with complete CoW, (2) a patient with hydrocephalus and incomplete CoW (Patient No. 1), and (3) a patient with hydrocephalus and incomplete CoW (Patient No. 2). The objectives here are to get a first insight into the flow patterns in the CoW of a healthy subject and of a patient with hydrocephalus and to define goals for the future investigations.

Flow simulation in CoW of healthy subject.

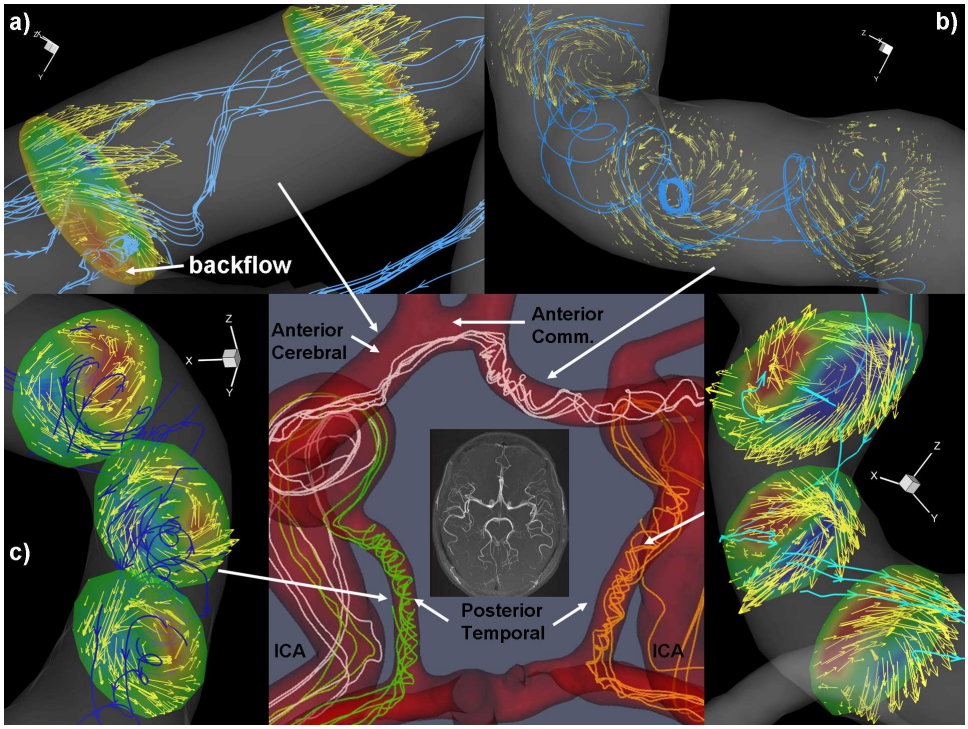


Figure 4.64: (in color) Brain blood flow simulation in complete Circle of Willis: development of secondary flows in the communicating arteries. a) swirling and backflow in the left Anterior Cerebral artery; colors represent streamwise velocity component; b), c) and d) swirling flow in the right Anterior Cerebral artery, left and right Posterior Communicating arteries, colors (in c) and d)) represent w-velocity (in-plane) component. Arrows represent direction of a flow.

Results of the flow simulation in the complete CoW are presented in figures 4.58 and 4.64. In figure 4.58 we plot flowrates and pressure drop at selected arteries $p(t) - p_{ref}(t)$, where $p_{ref}(t)$ is the pressure computed at inlet of the left ICA. In figure 4.64 the swirling flow is illustrated by instantaneous streamlines and vectors of the velocity field. The unsteadiness

and phase shift in the flow waves imposed at inlets result in alternating flow direction in the communicating arteries. Alternating mean flow direction was also observed in 1D simulations.

Flow simulation in incomplete CoW of a patient with hydrocephalus.

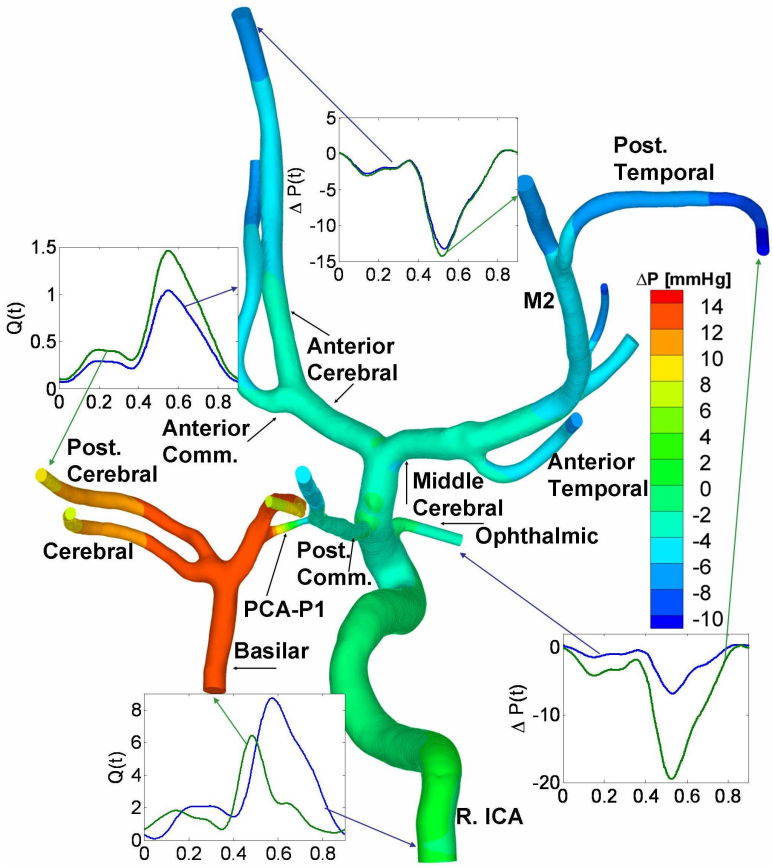


Figure 4.65: (in color) Brain blood flow simulation in incomplete Circle of Willis of a patient with hydrocephalus: Geometrical model of 23 cranial arteries; branches of the left ICA are not shown. XY plots depict the flowrate in ml/s and pressure drop, colors represent pressure difference at $t = 0.6s$ $\Delta p = p - p_{ref}$ in $mmHg$, where p_{ref} is the average pressure at ICA inlet. The constant flowrate observed at the ICA at time interval of $0.2s$ to $0.35s$ is not typical for a healthy subject.

In simulation of the flow in the incomplete CoW (patient No. 1) a relatively high pressure drop was observed through segment PCA-P1 (see figure 4.65). The PCA-P1 is very narrow artery, it connects the arterial branches of the internal carotid artery to those of the basilr artery. The internal carotid and basilar arteries are the main source of the blood supplied to the CoW. Due to the incompleteness of CoW, the PCA-P1 is the only

arterial segment communicating blood between the two sources, hence the flow dynamics in this artery is affected primarily by the blood demand from different parts of the brain. This points to the need to perform a detailed study to understand its nature, and to answer the question if the high Δp is a consequence of the pathological processes and what is the role of the boundary condition modeling. High Δp was also observed in other simulations performed using various settings for the outflow pressure boundary conditions. The accuracy provided by the *RC* model depends on the resistance values [43]. These values are certainly patient-specific, and additional care should be taken when we consider a pathological case. In our simulations the resistance was of order $10^{10}\ Pa\ s\ m^{-3}$, which is comparable to the data reported by Stergiopulos [87]. In the simulations of flow circulation in a patient with hydrocephalus, relatively high discrepancy between the prescribed flowrate ratios at posterior cerebral arteries and numerically computed ones (about 38%) was observed. To enhance the accuracy of the method the resistance parameters must be increasing, which is consistent with the fact that patients with hydrocephalus have higher peripheral resistance than healthy subjects due to high intracranial pressure. Doubling the resistance values resulted in lower differences (17%) between the prescribed by the *RC* method flowrates and the computed ones, however, the high pressure gradient at PCA-P1 artery was still present. We concluded that the high Δp is due to the phase shift in the flowrate waveforms imposed at the two inlets. This finding may affect diagnosis of a vascular disorder in the cranial system. One way to validate our results is to perform dye angiogram to compare the flow distribution in the cranial arteries with results of the simulation. The dye angiogram requires X-ray scanning and is performed only when *specifically clinically* indicated due to two reasons: a) a considerable amount of radiation, and b) the invasiveness of putting catheters into the vessels of the leg and brain; this is painful procedure and usually requires sedation and sometimes general anesthesia, and carries a risk of stroke. A non-invasive and harmless technique to measure intracranial flow is the Doppler Ultrasound, however, in the case of our patient, this method is not applicable due to high skull-bone thickness. Currently, we are working on implementing high-resolution PC-MRI to obtain the 3D velocity field in the major cranial arteries. This technique has the potential of replacing dye angiograms, but it requires variable velocity encoding to extract data from different vessels and different slices. Development of secondary flows in the inclomplete CoW model was also observed. In figure 4.66 we plot the flow patterns close to the junction where the ICA bifurcates to

the Middle Cerebral and Anterior Cerebral arteries, which is a typical region for developing aneurysms [103].

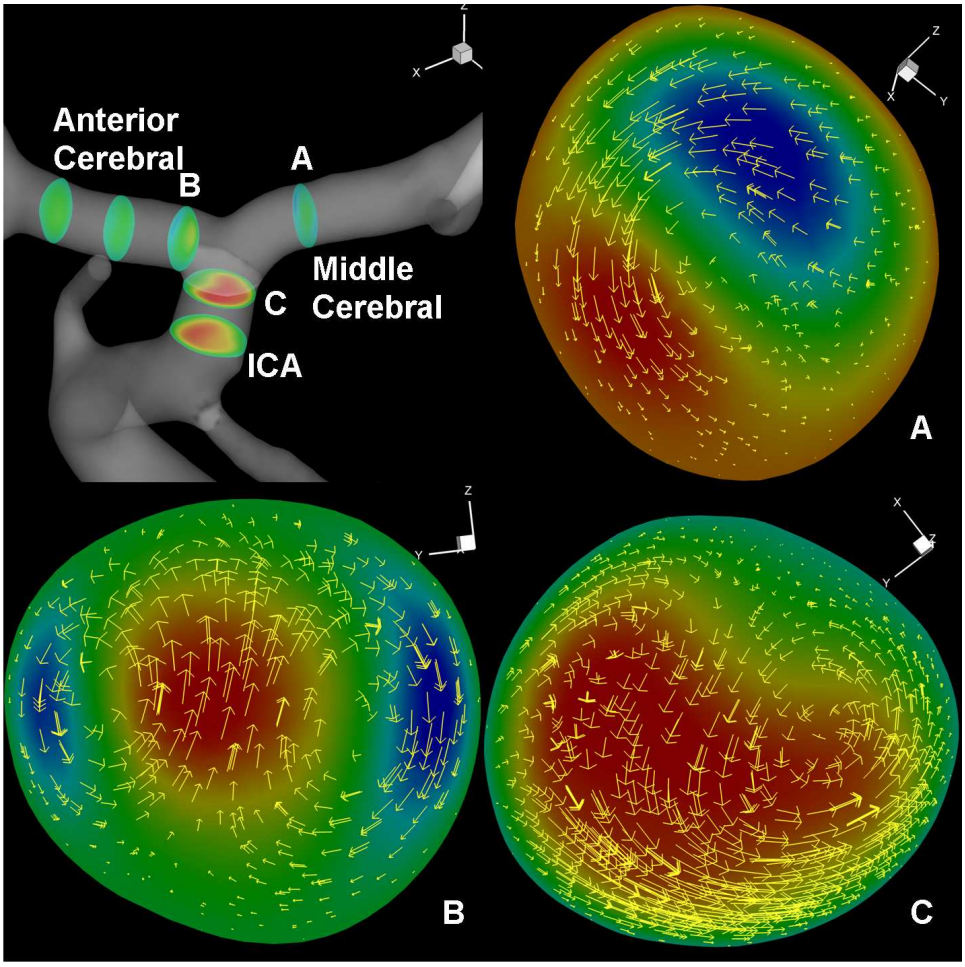


Figure 4.66: (in color) Brain blood flow simulation in incomplete Circle of Willis: development of secondary flows in the Middle Cerebral (A), Anterior Cerabral arteries (B) and Internal Carotid (C) arteries; colors represent v-velocity component in (A) and w-velocity component in (B,C). Arrows represent direction of a flow.

Flow simulation in complete CoW of a patient with hydrocephalus.

Unsteady flow simulations in complete CoW of patient No. 2 have revealed similar flow patterns: swirling flow, alternating flow directions in communicating arteries. In figure 4.67 the flow patterns at the junctions of the posterior communicating artery are presented. We recall that role of communicating arteries is to redistribute the blood supplied through the internal carotid and basilar arteries. It was observed that the changes in the flow direction occur at the end diastolic phase, when the flow rates at inlets of CoW are low. The flow

direction changes and presence of secondary flows are associated with high temporal and spatial gradients in the wall shear stress. The flow patterns depicted here do not necessarily point to a certain pathology, but rather provide an insight into the complexity of a blood flow in CoW. Further research is required in order to correlate particular flow patterns with pathologies such as aneurysm developing and/or rapture.

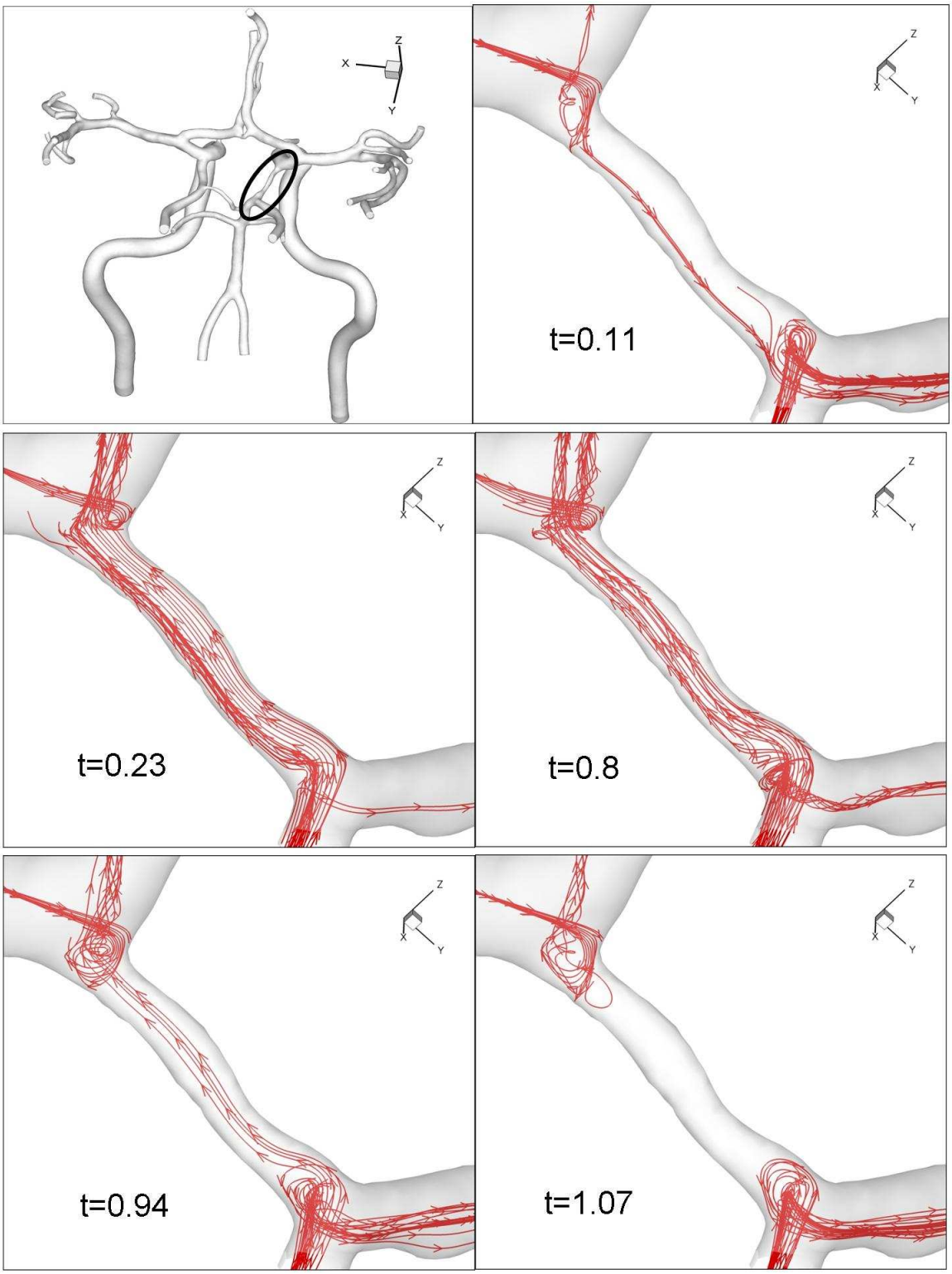


Figure 4.67: (in color) Brain blood flow simulation in complete Circle of Willis of a patient with hydrocephalus: development of secondary flows at the junctions of the posterior communicating artery (highlighted in the top left plot). Lines and arrows represent instantaneous stream lines and flow direction. Time interval of one cardiac cycle is $T = 1.1s$.

4.6 Discussion and outlook

Chapter 4 has been devoted to fundamental challenges in flow simulation in complex arterial networks: arterial geometry reconstruction, integration of clinically measured inlet/outlet flowrates as boundary conditions in numerical simulations, and fast scalable parallel numerical solvers for very large (in terms of number of unknowns) 3D unsteady flow problems.

Outflow boundary condition.
Patient-specific simulations require information about the geometric model as well as the flow rates at the terminal vessels. Several *models* for imposing pressure boundary condition exist, however, the implementation of these models is not simple and typically requires very expensive tuning of several parameters to obtain the correct flow rate distribution at all outlets. Here, we present a simple *method* on how to impose the measured flow rates at tens of outlets, which is accurate, stable and scalable, i.e., it can be used in simulations of full-scale arterial trees. The procedure is very simple as the flow rate ratios determine resistance ratios, and appropriately large values of resistances can then be selected arbitrarily to guarantee the accuracy of the method. The flow rates can be obtained using simple non-invasive Ultrasound based measurements. It is important to note here that monitoring flow rates at different vessels must be correlated in time in order to retain the correct phase shift in numerical simulations.

An *alternative method* to impose patient-specific flow rates at multiple outlets relies on implementation of the impedance boundary condition. This method requires measurements of *both* the flow rate and the pressure at each of the terminal vessel in order to compute patient specific impedance. These measurements also must be correlated in time to ensure a proper phase shift. Accurate pressure measurements in multiple arteries are rare since they require invasive procedures. In addition, implementation of the impedance boundary condition in numerical simulations of a 3D flow in large computational domains is computationally inefficient.

Two-level domain decomposition and Multilevel Communicating Interface.
Currently none of the existing domain decomposition methods for the Navier-Stokes equations scales well beyond a few thousand processors, and hence their anticipated performance on petaflop-size systems will be very poor. In Chapter 4 a two-level decomposition method that can potentially scale well to hundreds of thousands of processors has been presented.

The main advantage of the suggested method is in splitting a large domain, where the solution is approximated with C^0 polynomial expansion, into a set of patches of manageable size. Continuity of the solution at patch interfaces is obtained by implementing a *DG*-like approach both for the advection and diffusion contributions. There is a potential loss of accuracy due to the patch interfaces but more elaborate conditions can be applied to enhance the accuracy as we have demonstrated here for a benchmark problem. Use of overlapping domains, improves the accuracy of the solution in the neighborhood of interfaces, and also enhances stability.

From the computational standpoint, solution of a set of small tightly coupled problems is more efficient than solution of a single much larger problem. In particular, the new two-level method we presented matches well with the new petaflop hybrid-type architectures, consisting of a few hundreds of "fat nodes", with each node containing several thousand processors. We can envision, therefore, a natural mapping of patches and nodes, with all intensive data transfers taking place efficiently within each node. The overall scalability of the method depends on the strong scaling within a patch and the weak scaling in terms of the number of patches. This dual path to scalability provides great flexibility in *balancing accuracy and parallel efficiency.*

The Multilevel Communicating Interface allows efficient (from the standpoint of parallel computing) coupling between different simulations. Here we considered coupling between several 3D fluid domains, however, the method is general and can be applied for multi-physics and multiscale simulations. The MCI also allows efficient parallel strategy for simulations on distributed computers.

Intracranial flow simulations.

Simulations of flow in the complete Circle of Willis (CoW) of a healthy subject and in the complete and incomplete Circle of patients with hydrocephalus show development of strong secondary flows in the communicating arteries and internal carotid artery. The simulation in the incomplete CoW predicts high pressure drop along the PCA-P1 artery, connecting the basilar with the internal carotid. Simulation of flow in the incomplete CoW requires setting the peripheral resistance to high values in order to ensure proper net blood flow to different arteries. The high pressure variation correlated with the phase shift at the inlet flowrates in the patient with hydrocephalus points to a new direction in diagnosing cranial vasculature disorders.

The discrepancies and similarities between results obtained with 1D and 3D models point to the need of fundamental verification and validation of the numerical models. The relatively small cross-sectional area fluctuations predicted by the 1D models can not be considered as an automatic "excuse" to use rigid arterial wall modeling, particularly in flow simulations in very large arterial networks.

We envision that the 1D models will be the default computational models to provide the first estimate of blood circulation and also in surgical planning. The role of the 3D simulations is to complement the 1D models and provide accurate information regarding 3D flow patterns, Wall Shear Stress, oscillatory shear index, low residence time zones, etc. The ability to follow vascular lesions over time with less invasive means could be of clinical utility immediately. More sophisticated quantitative blood flow techniques may find near-term value in clinical medicine. We envision that CFD modeling of blood flow will replace many invasive and dangerous clinical tests. While the present results are encouraging, more work is required to develop a truly multiscale framework by coupling the three-level networks, namely MaN-MeN-MiNin a seamless integration. In summary, the numerical methods and simulation results presented in this Chapter can be considered as a first building blocks in developing an efficient and scalable biomechanics gateway for analyzing the biophysics of brain pathologies, such as hydrocephalus, in a systematic way on the emerging petaflops computer platforms of the next decade.

Chapter 5

A study of transient flow in stenosed carotid artery

5.1 Introduction

We employ Computational Fluid Dynamics (CFD) to investigate blood flow in a carotid artery, which has an occlusion in the cross-section area of its internal branch. The common carotid artery (CCA) transports blood from the arch of aorta (left CCA) or from the brachycephalic artery (right CCA) to the internal carotid artery (ICA) and external carotid artery (ECA); the latter carries blood primarily to the facial tissues while ICA supplies blood to the brain. Interruption in the blood flow to the brain may lead to a stroke. There are two major kinds of strokes: (1) an ischemic stroke, caused by a blood clot that blocks or plugs a blood vessel or artery in the brain; and (2) a hemorrhagic stroke, caused by a bleeding vessel. According to the American Heart Association, the carotid disease is the major risk factor for ischemic stroke, caused by detachment of the plaque, which may clot vital brain vessels. Atherosclerotic plaques inside an arterial wall result in a *local* occlusion of the artery lumen - a stenosis. The stenosis may trigger transition to turbulence, and onset of turbulence downstream of severe occlusions has been observed in laboratory experiments [31]. Turbulence may damage the vessel wall, causing plaque to build up at the site of damage. Consequently, this build-up further increases turbulence and hence the level of turbulence intensity correlates with the degree of stenosis. It is now well established that the transition to turbulence is expected to be dependent on flow pulsatility and on

the geometry of the arterial wall. In particular, the effect of pulsatility on transition to turbulence is common in different arterial flows and many studies support this view [53].

Detection of arterial narrowing can be performed using catheter arteriography; its pathophysiological relevance has been demonstrated empirically by close correlation between the degree of stenosis and the subsequent risk of stroke. However, arteriography is an invasive procedure, and hence there is a need in developing non-invasive methods. Presently, diagnosis of an arterial occlusion is routinely performed by non-invasive techniques: computer tomography (CT), magnetic resonance (MR) or color Doppler ultrasound (CDUS) scanning. Audible sounds (carotid bruit), detected in the proximity of the carotid artery during auscultation may also indicate presence of stenosis [30, 12]. Weak correlation between the carotid bruit and stenosis was reported in [73] but it was emphasized that the detection and diagnosis of the degree of occlusion cannot rely on auscultation only. In particular, high-frequency components of pressure oscillations may be attributed to wall movement and/or to turbulence. In our studies we considered a *rigid* arterial wall model and hence the high-frequency oscillations predicted by our high-accuracy CFD simulations are of hydrodynamic origin, i.e., due to the turbulence *solely*, which is triggered by the stenosis.

Flow in a stenosed carotid artery has been studied experimentally [17, 50, 14] and numerically [36, 89, 86]. Here, we apply the Proper Orthogonal Decomposition (POD) to analyze pulsatile transitional *laminar-turbulent* flows in a carotid arterial bifurcation. POD was introduced in fluid mechanics in [58] for analyzing complex flow regimes, and a comprehensive review on application of the POD method in turbulent flow analysis can be found in [19]. Our focus is on the onset of turbulence and subsequent re-laminarization. Using high-accuracy CFD results, we demonstrate the capability of an extension of POD to identify the transitional and intermittent regimes in a stenosed carotid artery. We show that the behavior of the POD eigenvalue spectrum are directly related to the presence of turbulence and hence it can be used for its detection during the cardiac cycle. We also propose a method on how to extend the POD analysis to clinical measurements, e.g., by phase-contrast magnetic resonance imaging (PC-MRI) or CDUS.

The Chapter is organized as follows. In section 5.1.1 we briefly discuss turbulence modeling and the computational set-up in our CFD simulations. In section 5.2 we discuss the patterns of transitional flow, and in section 5.3 we present the *time-* and *space-window* POD for detecting turbulence. In section 5.4 we discuss the applicability of the window-POD

in the clinical setting by comparing high-resolution results with those mimicking medical images obtained with moderate resolution. In section 5.5 we conclude with a discussion and a brief outlook on the application of our method in the analysis of clinically measured data. In the Appendix we provide details on the resolution studies, the arterial geometry reconstruction, and the inlet/outlet boundary conditions.

5.1.1 Modeling transitional flow in carotid artery

Numerical simulations of blood flows in complex geometries are carried out mostly by employing the Reynolds-averaged NavierStokes (RANS) approach. Stroud *et. al.* [89] employed a low-Reynolds number two-equation turbulence model for a realistic carotid bifurcation but using its two-dimensional projection. In three-dimensional geometry of arteries, the presence of high-degree stenosis and flow pulsatility greatly increase the probability that turbulence may be sustained even at relatively low Reynolds numbers [53]. Since the standard RANS models are derived for fully developed high Reynolds number turbulence, they are not appropriate for modeling transitional flow. Low Reynolds number RANS approaches have been validated using mostly simple geometry benchmarks and cannot give reliable results for cardiovascular flows. In general, RANS models cannot predict complex features of such flows because they may (a) erroneously generate turbulence, and (b) underestimate back-flow regions due to artificially increased dissipation by overestimating the turbulent viscosity. A need, therefore, exists to extend computational studies of transitional flows through stenosed arteries by direct numerical simulation (DNS, simulation without applying *ad hoc* models) in three-dimensional (3D) realistic geometries.

To perform three-dimensional DNS of flow through carotid artery we employed the high-order spectral/*hp* element code $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$. Spectral/*hp* element spatial discretization provides high accuracy and is suitable for complex geometries; it is particularly effective in capturing intermittent laminar-turbulent regimes since it does not suffer from artificial dissipation. Other details can be found in the Appendix. In the following, we present the computational setup.

5.1.2 Geometry, computational domain and grid generation

A geometric model of the carotid artery was obtained from in-vivo MRI images shown in Fig. 5.1(a) processed by an in-house software package to generate a model of the arterial

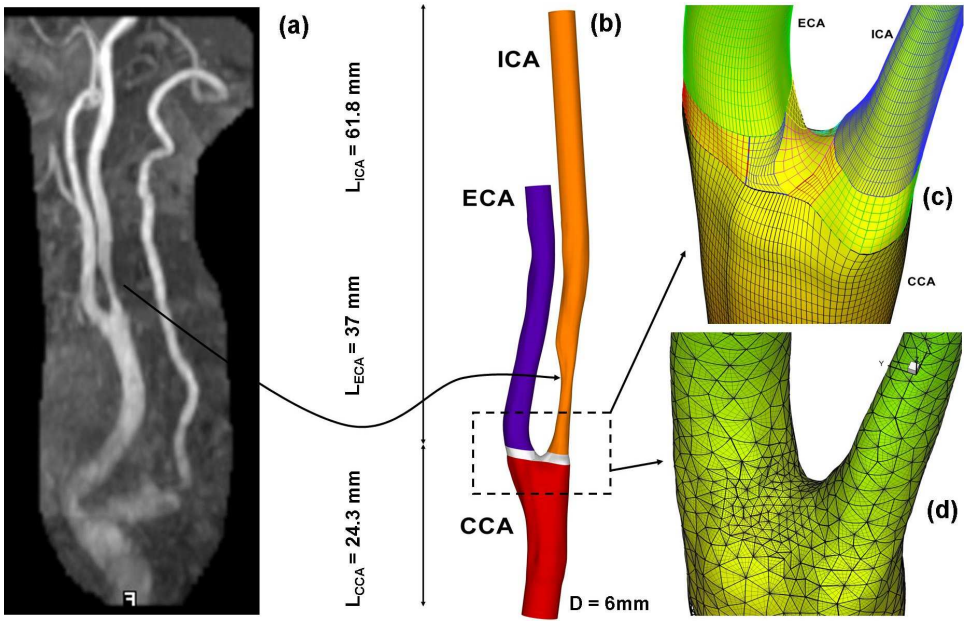


Figure 5.1: (in color) Reconstruction of arterial geometry from MRI images: (a) - MRI of carotid artery (courtesy of Prof. M. Gomori, Hadassah, Jerusalem); (b) - Geometrical model of a carotid artery; colors represent different arterial segments. (c) - Patches of parametric surface representation, colors represent different patches. (d) - computational mesh, consistent with third-order polynomial approximation.

wall. Editing of the arterial geometry and subsequent mesh generation was performed using Gridgen - a commercial mesh generator developed by Pointwise [6]. In Fig. 5.1(b) we show the geometric model reconstructed from MRI images. In the current study we use a mesh with 22,441 tetrahedral spectral elements of variable size, and eighth-order polynomial approximation ($P = 8$) within each element, corresponding to $24,685,100$ degrees of freedom (DOF) per variable. For consistent integration and for preventing aliasing – an important issue in resolving accurately the onset of turbulence – the 3/2-rule was applied in the calculation of the nonlinear terms in the Navier-Stokes equations [51]. Thus, the total number of quadrature points in the computational domain was above 37 millions. A systematic resolution study was performed using $h-$ and $p-$ refinement techniques. Specifically, the spectral/hp element method provides a dual path to convergence (i.e., decay of numerical error) as follows: (a) h-convergence, with the accuracy of the solution depending on the size of elements; and (b) p-convergence with the accuracy depending on the order of polynomial approximation. To enhance the accuracy, a local mesh refinement (h-refinement) was applied downstream of the ICA narrowing. Also, the flat faces of the surface elements were projected on smooth curved boundaries. In Fig. 5.1(c) we plot the parametric surface

mesh (2D structured grid) at the carotid bifurcation and in Fig. 5.1(d) we show the computational grid projected on the curved surface. For temporal discretization, a second-order semi-implicit time splitting scheme was implemented.

5.1.3 Problem formulation

The computational domain consists of the common, internal and external carotid arteries (CCA, ICA and ECA, respectively). A fully developed velocity profile was prescribed at the proximal end of the CCA using superposition with the well-known Womersley analytical profile for each Fourier mode of the flow rate curve. To this end, the flow rate $Q(t)$ shown in Fig. 5.2 was approximated by $N = 20$ Fourier modes as $Q(t) = A_0 + \sum_{k=1}^{N}[A_k \sin(k\omega t) + B_k \cos(k\omega t)]$; here ω is the main frequency and A_0, A_k, B_k are the coefficients of the Fourier expansion. At the distal end of the ICA and ECA, we used time-dependent RC-type boundary condition for the pressure, supplemented with the zero Neumann boundary condition for velocity. The degree of stenosis, S, in the ICA is defined as the ratio of the reduction in cross-sectional area due to stenosis, i.e., $S = 1 - (D_{min}/D_{ICA})^2$, where D_{ICA} is the ICA effective diameter measured downstream the stenosis, and the effective diameter D_{min} is computed from the area at the stenosis throat: $A_{min} = \pi D_{min}^2/4$. The Reynolds ($Re = U_m D/\nu$) and Womersley ($Ws = 0.5D\sqrt{\omega/\nu}$) numbers are based on the parameters at the CCA inlet. Here, U_m is the average (in space and time) velocity, $\omega = 2\pi/T$ where T is the one cardiac cycle time interval, and ν is the kinematic viscosity. In our simulations: $S \approx 77\%$, $D = 6\ mm$, $\nu = 3.283\ mm^2/s$ and $T = 0.9$ s. Consequently, $Re = 350$ and $Ws = 4.375$. The time step was chosen to satisfy the CFL stability constraint and is very small due to high spectral resolution, in the range of $\Delta t \in [5.2 \times 10^{-8},\ 10^{-6}]\ s$. This small time step allows accurate resolution of the high frequency flow oscillations. To ensure sufficient temporal resolution for the POD study, we save the velocity field with intervals of about 8.0×10^{-4} seconds. Starting with low-order spatial approximation and zero velocity initial field, we performed numerical simulation over one cardiac cycle. Then, the spatial and temporal resolution were increased and the flow was simulated during two additional cardiac cycles; for analysis we used the results collected over the last cardiac cycle.

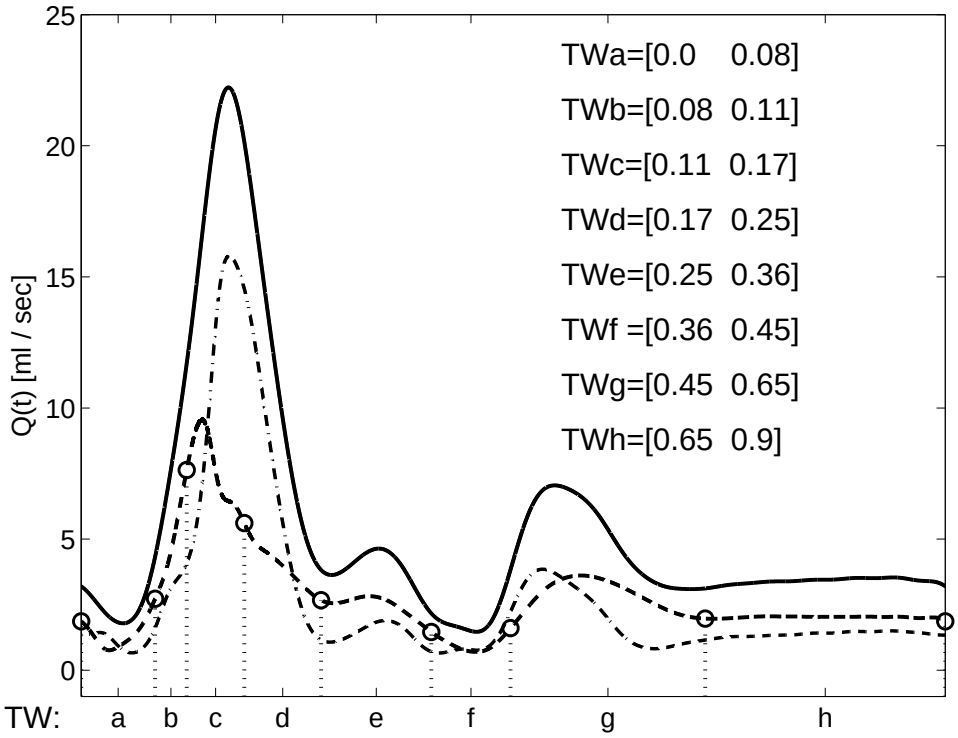


Figure 5.2: Waveform flow rates imposed in the CCA (solid), ICA (dash) and ECA (dash - dot) arteries and Time-Windows selected for POD data analysis.

5.2 Flow patterns

Back-flow: Atherosclerotic plaques usually occur in regions of branching and marked curvature, at areas of geometric irregularity, and also in regions where blood flow undergoes sudden changes in magnitude and direction resulting in back-flow regions. Specifically, back-flow regions occur in the vicinity of carotid bulb or stenosis. These regions are characterized by very low wall shear stress (WSS) and high pressure. The oscillating shear index (OSI), as defined by [54], is a dimensionless local measure that quantifies the time a articular wall region experiences back-flow during the cardiac cycle. More specifically, high values of OSI can reveal the location of the back-flow boundary where WSS is zero. Regions of both low WSS and high OSI are thought to be susceptible to intimal thickening and plaque formation that promotes atherogenesis at these important sites within the carotid arteries. In figure 5.3(left), we show back-flow regions detected in the carotid bulb and the immediate proximal vicinity of stenosis; these back-flow regions are quite stable throughout the systole phase. Figure 5.3(right) shows typical instantaneous pathlines; while in CCA and ECA pathlines are quite organized, those in ICA exhibit disorder in the poststenotic region featuring a swirling pattern. In figure 5.3 we also show complex cross-stream secondary flows.

Jet flow and onset of turbulence: Sherwin & Blackburn [77] performed 3D DNS to study instabilities and transition to turbulence in a straight tube with a smooth axisymmetric constriction, which is an idealized representation of a stenosed artery. They reported that steady flow undergoes a Coanda-type jet formation and turbulent transition exhibiting hysteretic behavior with respect to changes in Reynolds number. (The Coanda effect is understood as the tendency of a jet stream to adhere to a curved boundary wall.) The blood flow along the curved wall is accompanied with decrease of the pressure on the wall, dropping below the surrounding pressure and resulting in the attachment of the fluid flow to the wall. The wall jet consists of an inner region, which is similar to a boundary layer, and an outer region wherein the flow resembles a free shear layer. These layers interact strongly and form a complex flow pattern. The laminar Coanda jet has the tendency to follow the circumferential wall only for small angles. The rapid increase of pressure along the wall occurs due to the influence of the curvature of the wall and the adverse pressure gradient in the direction of flow leading to flow separation. In general, the wall jet flows

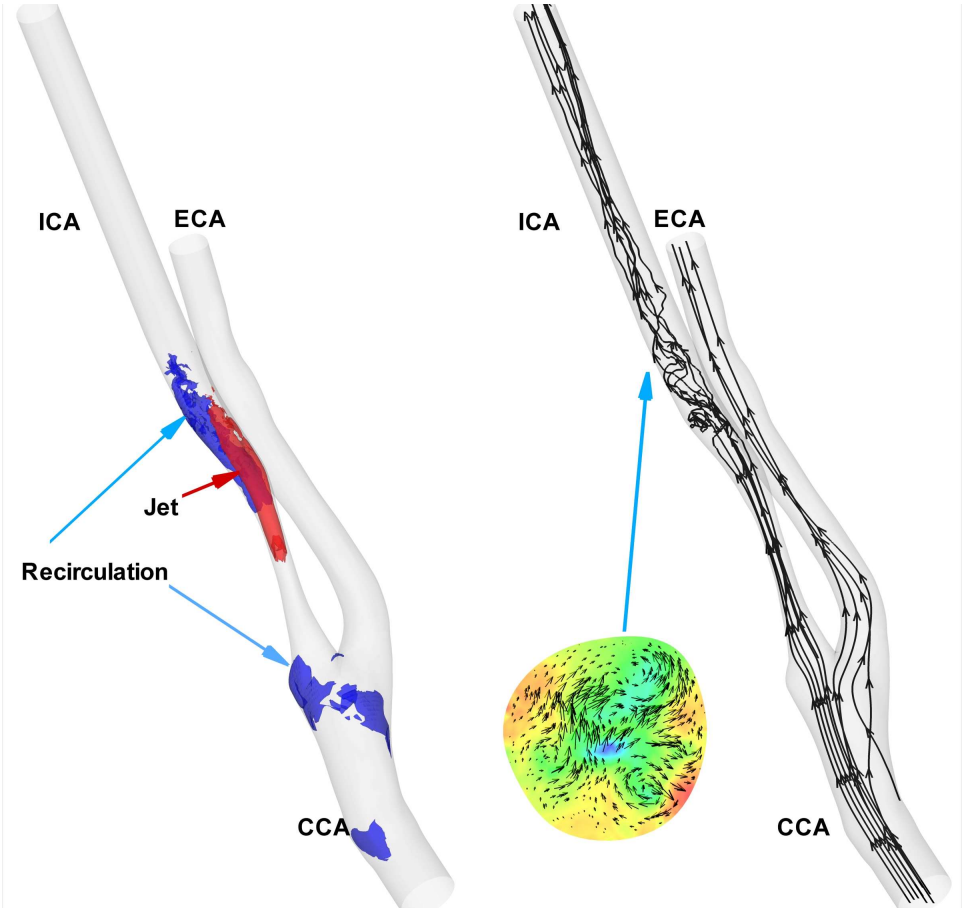


Figure 5.3: (in color) Flow patterns; left: iso-surfaces in a high-speed region (jet, red), blue iso-surfaces - back-flow regions; right: instantaneous path-lines of swirling flow and cross-stream secondary flows.

are turbulent owing to the *low* critical Reynolds number. Experimental and theoretical results have demonstrated that the instability of the entire wall jet is controlled by the outer region. Bajura & Catalano [16] investigated experimentally transition to turbulence in planar wall jets, using flow visualization in a water tunnel. They observed the following stages in natural transition: (i) formation of discrete vortices in the outer shear layer; (ii) coalescence of adjacent vortices in the outer region, coupled with the rolling up of the inner shear layer; (iii) eruption of the wall jet off the surface of the flat plate into the ambient fluid (the "lift-of" stage); and (iv) dispersion of the organized flow pattern by three-dimensional turbulent motions.

Figures 5.4 and 5.5 demonstrate the jet-like effect created by the stenosis as predicted in our calculations. In Fig. 5.4 we show the contours of the Ω_y-vorticity (transverse) that can be linked to the rolling up along the wall (see coordinate axes in Fig. 5.4f), for

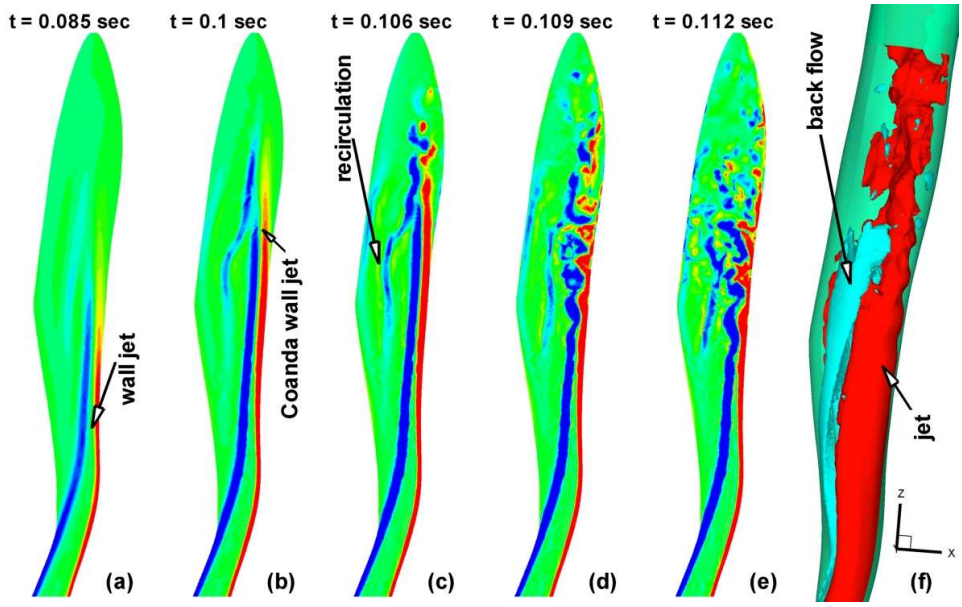


Figure 5.4: (in color) Unsteady flow in carotid artery: transition to turbulence. (a-e) cross-flow vorticity contours Ω_y extracted along $y = -1.2$ in ICA. (f) region of ICA where flow becomes unstable, colors represent iso-surfaces of w-velocity (streamwise, along z-direction), $Re = 350$, $Ws = 4.375$.

different stages of transition that show the wall jet breakdown. These results illustrate the onset of turbulence due to shear layer type instabilities of the Coanda wall jet in the post-stenotic region. Specifically, in Fig. 5.4a, the laminar state of the incoming flow is confirmed by the straight path traces. The jet outer region (marked by a blue trace) and the adjacent recirculation back-flow region (marked by a light-blue trace) form a free shear layer. In Fig. 5.4b, the jet moved downstream along the wall showing the early stage of interaction with the adjacent recirculation region. Figure 5.4c shows the perturbed shear layer at the leading edge of the jet. The tilted vorticity trace in Fig. 5.4d provides evidence of the stage of vortices coalescence in the outer region and the rolling up of the inner shear layer. The tilted wall jet rapidly breaks down leading to dispersion of the organized flow pattern (Figs. 5.4d and e). It should be noted that the breakdown gradually propagates upstream, a phenomenon that was predicted by Sherwin & Blackburn [77] using DNS in a simplified geometry. Figure 5.5 shows iso-surfaces of the longitudinal w-velocity (streamwise) component, depicting the stages when the coherent wall jet structure (Fig. 5.5a, green) collapses in the post-stenotic region (Fig. 5.5e). We note that the recirculation region marked by the blue iso-surfaces, does not collapse, which implies that this region is quite stable.

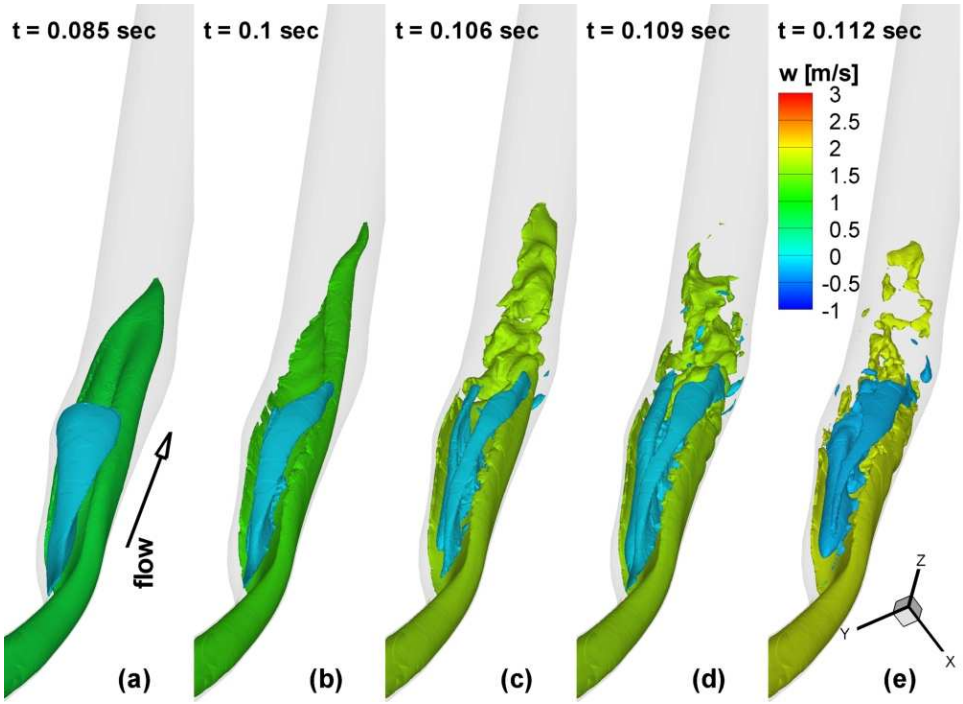


Figure 5.5: (in color) Wall jet formation and breakdown. Streamwise w-velocity iso-surfaces, blue indicates a back-flow recirculation region. $Re = 350$, $Ws = 4.375$.

Transient turbulence followed by re-laminarization: The transient turbulence regime changes the hemodynamic factors (e.g., wall shear stress, WSS). Specifically, it causes the WSS vary significantly through the cardiac cycle; i.e., to become very high proximally to the stenosis throat in the ICA where the vulnerable plaque is usually built-up. For detection of turbulence in time and space, time traces of instantaneous axial velocity have been monitored along the ICA at several axial stations indicated in Fig. 5.6(left). From Fig. 5.6c, the flow disturbances, as they appear during the cardiac cycle, reveal that the turbulent state appears during the systolic phase and is localized in the post-stenotic region, with re-laminarization occurring farther downstream. To link the transition process to the time frame of the cardiac circle, in Fig. 5.6(right bottom plot) we also show the physiological flow rate waveforms in the CCA and ICA imposed in our calculations. The waveform curves consist of a brief systolic phase (acceleration and deceleration) and a longer diastolic phase with some increase in flow rate around $t \approx 0.55$. As follows from the time traces in Fig. 5.6c, the early turbulent activity in the post-stenotic region begins at the mid-acceleration phase of the cardiac cycle. In the early part of deceleration there is intense turbulent activity; past the mid-deceleration phase, the intensities die out and the flow begins to re-laminarize;

an exception is a short-term oscillation at $t \approx 0.55$. The transient turbulent regime lasts about 90% of a systole time, i.e., about 0.15 seconds or nearly 17% of the cardiac circle.

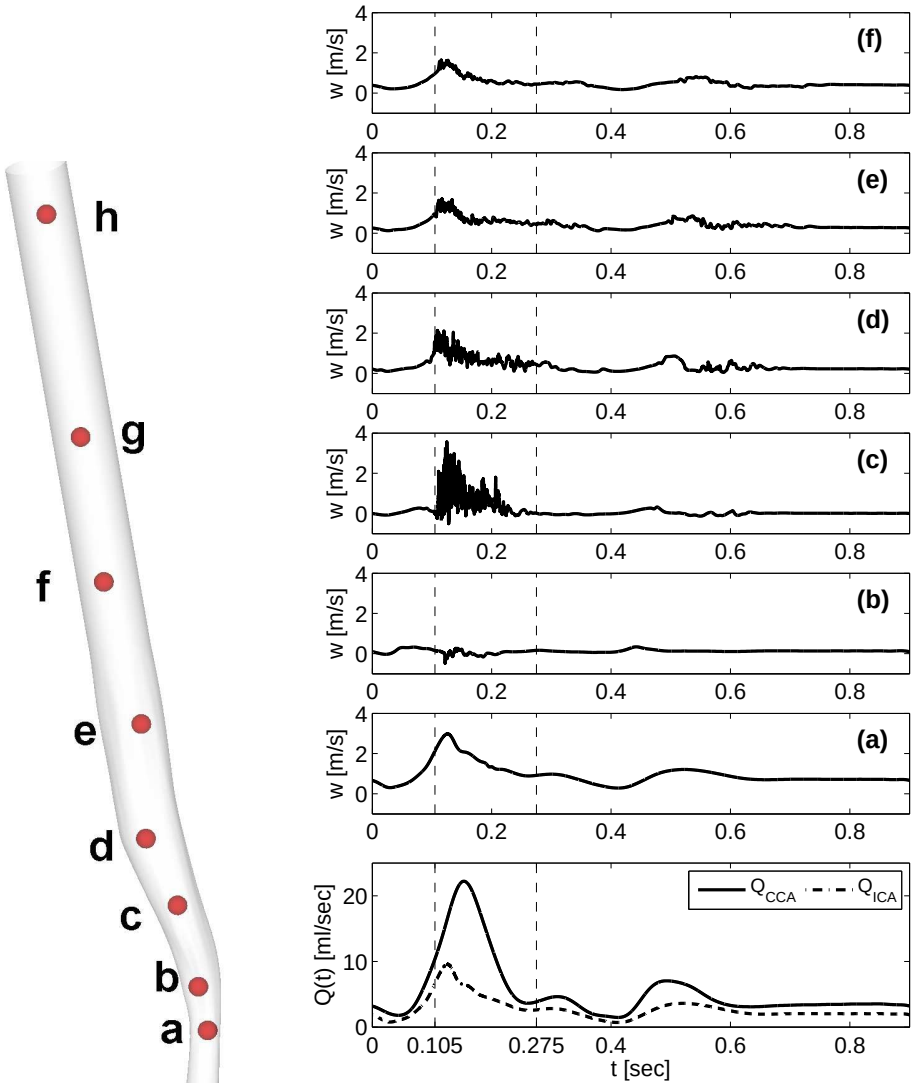


Figure 5.6: (in color) Wall jet formation and breakdown. Streamwise w-velocity iso-surfaces, blue indicates a back-flow recirculation region. $Re = 350$, $Ws = 4.375$.

5.3 Application of Proper Orthogonal Decomposition

In this section we apply Proper Orthogonal Decomposition to detect turbulence in the stenosed carotid artery. The analysis is performed using two approaches: (a) POD based on the data collected over a *complete* cardiac cycle, and (b) *windowed* POD, that is the decomposition is performed over specific intervals of the cardiac cycle and also in different spatial 3D and 2D sub-domains. POD analysis performed over different time intervals shows clear correlation between transition to turbulence and the POD eigenspectrum. Analysis of the temporal and spatial modes of POD provides additional information regarding the onset of turbulence.

5.3.1 Eigenvalue spectrum of POD

POD is a very effective method for identifying an energetically dominant set of eigenmodes in an evolving system. It was originally introduced in fluid dynamics for identification of coherent structures in a turbulent velocity field. Sirovich [82] introduced the *snapshot* method to analyze large data sets.

For a set of data $\mathbf{u}(\mathbf{x},t)$, represented as a function of physical space $\mathbf{x}$ and time t, POD determines a set of orthogonal basis functions of space $\phi_i^k(\mathbf{x})$ and temporal modes $a_k(t)$; here $i = 1,\ 2,\ 3$ is the coordinate index and $k = 1, 2, ..., M$ is the mode index. The basis is sought so that the approximation onto the first K functions: $\hat{u}_i(\mathbf{x},t) = \sum_{k=1}^{K} a_k(t)\phi_i^k(\mathbf{x}),\ \ K \leq M$ has the largest mean square projection. We shall use the method of snapshots to compute the POD modes in a time interval T. Here the inner product between every pair of velocity fields (snapshots) $C(t,t') = T^{-1}\int_\Omega \mathbf{u}(\mathbf{x},t)\mathbf{u}(\mathbf{x},t')d\mathbf{x}$ is the temporal auto-correlation covariance matrix $C(t,t')$ used as the kernel. The temporal modes $a_k(t)$ are the eigenvectors of the $C(t,t')$ matrix and are calculated by solving an eigenvalue problem of the form: $\int_T C(t,t')a_m(t')dt' = \lambda_m a_m(t)$. Using orthogonality, the POD spatial modes $\phi_i^k(\mathbf{x})$ are calculated by $\phi_i^k(\mathbf{x}) = T^{-1}(\lambda_m)^{-1}\int a_k(t)u_i(\mathbf{x},t)dt$.

A spatio-temporal mode $a_k(t)\phi_i^k(\mathbf{x})$ represents a basic flow structure which has its own contribution to the total flow field. The eigenvalue of a single mode represents its contribution to the total kinetic energy of the flow field $\int_\Omega \langle u_i(\mathbf{x})u_i(\mathbf{x})\rangle d\mathbf{x}$ which is equal to the sum over all eigenvalues. Therefore, the eigenspectrum of the decomposition can be regarded as the primary information indicating the importance of each individual mode from the

energetic point of view. The modes with the lowest numbers are the most energetic modes and correspond to coherent flow structures. Specifically, we define

$$E_M = \sum_{k=1}^{M} \lambda_k, \qquad E_{n,m} = \sum_{k=n}^{m} \lambda_k, \tag{5.1}$$

where λ_k are the eigenvalues, E_M is the total kinetic energy and $E_{n,m}$ is the kinetic energy associated with modes from mode n to mode m.

For stationary turbulent flows, the time-averaged field can be calculated and we can compute how many modes contribute to the time-averaged energy. The remaining modes can be attributed to the turbulent field. In the POD analysis of the velocity field at high-Reynolds number obtained in [60], the time-averaged flow field contained about 96% of the total kinetic energy which was primarily caried by mode $k = 1$. Thus, the remaining fluctuating modes (k$\geq$2) were considered as "turbulent". To extract the "turbulent" contribution of the flow field in our simulations, we take advantage of the hierarchical feature of the POD decomposition. Specifically, in our calculations the energy associated with the first two POD modes, computed over a cardiac cycle, is about 96%, which we consider as the ensemble- (or phase) averaged field, and tale the rest POD modes (k$\geq$3) to represent the "turbulent" velocity field.

The presence of fluctuations is not sufficient evidence for the presence of turbulence, which is characterized by specific statistical properties, and hence the distribution of energy between different scales is a commonly used criterion. In figure 5.7 we show the normalized eigenspectrum obtained at two different Reynolds numbers: $Re = 350$ and $Re = 70$. We observe a big difference between the spectra: the eigenspectrum at $Re = 350$ decays slowly, while that for a laminar flow ($Re = 70$) decays rapidly. It is noteworthy that at $Re = 350$, the energy of the higher modes ($k = 5$ to 200) decays with a *power law* (slope $s \approx -1.1$). POD analysis of high-Reynolds number turbulent flows was applied in [60] to flow around a wall-mounted hemisphere and by Yakhot and Liu (unpublished) to a wall-mounted cube. Both studies clearly showed the power law decay rate $s = -3/4$ of the POD high eigenvalues. The fact that the same power law was obtained in two different POD studies is intriguing, however, the precise value of the exponent is probably not universal. Fourier analysis applied to turbulent fields shows that the energy spectrum follows a power law k^{-s} with $1 < s < 3$ depending on the turbulence nature. For homogeneous isotropic turbulence, the

POD eigenmodes are simply Fourier modes. The flow in the carotid is neither homogeneous nor isotropic but a power law energy decay provides additional evidence for the presence of turbulence.

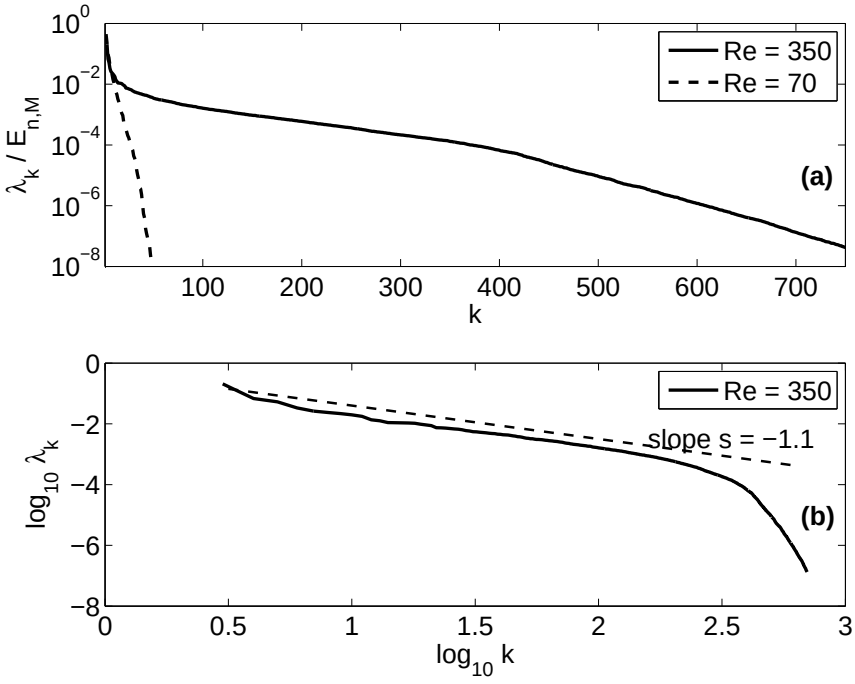


Figure 5.7: POD eigenspectra. $Re = 70$ and $Re = 350$; $Ws = 4.375$. The values of n and m are: $Re = 70$ - $n = 2$ and $M = 64$, $Re = 350$ - $n = 3$ and $M = 1125$.

5.3.2 Detection of turbulence by POD analysis

As we already mentioned, the POD modes are hierarchical, with the first mode representing the main flow pattern while the higher modes add finer features. In figure 5.8(left) we plot selected temporal POD modes computed over one cardiac cycle at $Re = 350$. The first mode, $a_1(t)$, essentially follows the flow rate waveform imposed at the CCA inlet (figure 5.2). As in the time traces of the instantaneous axial velocity shown in figure 5.6(b-f), fluctuations of the higher modes indicate a transient turbulent regime throughout the systole phase. A short-duration weak oscillation around $t \approx 0.55$ that were recorded in time-traces in figure 5.6d are captured here by the high temporal modes. It is caused by a brief transition to turbulence due to a secondary acceleration-deceleration hump in the flow rate. In figure 5.8(right), we show iso-surfaces of the velocity magnitude in the post-stenotic region reconstructed from the turbulent modes ($k = 3, 4, ..., M$) at two different

phases, peak-systole ($t = 0.12$) and end-diastole ($t = 0.0$). The imprints of turbulence are clearly seen in the plot corresponding to $t = 0.12$, consistent with the turbulence activity detected by the high temporal modes.

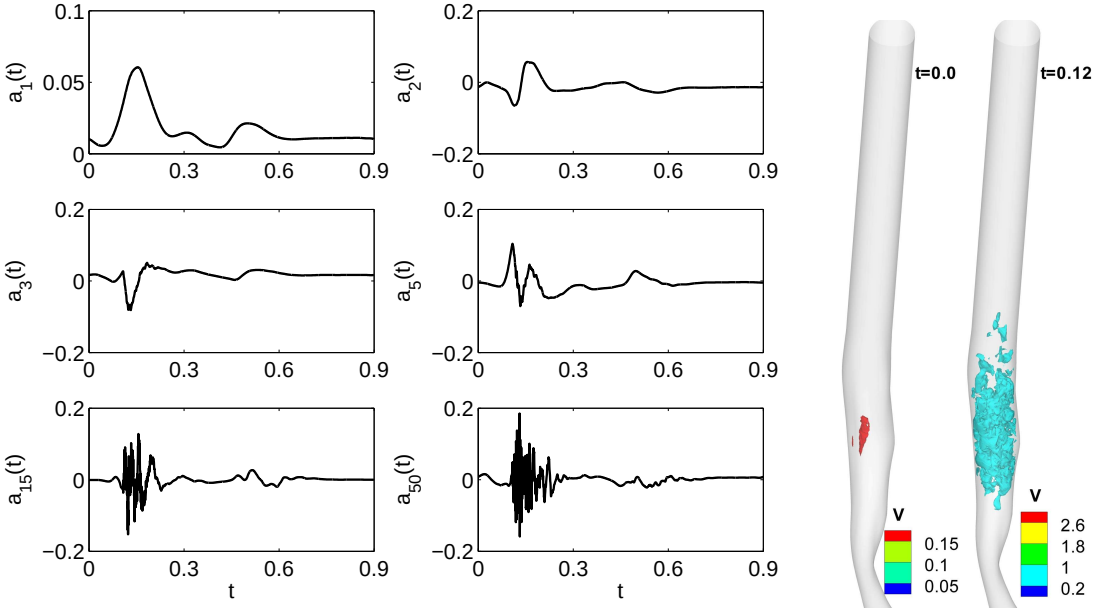


Figure 5.8: Left: temporal POD modes of velocity obtained over one cardiac cycle. Right: velocity field reconstructed from high-order POD modes $\breve{u}_i(t, \mathbf{x}) = \sum_{k=3}^{M}[a_k(t)\phi_i^k(\mathbf{x})]$ at time instances $t = 0.0$s and $t = 0.12$s (systolic peak). Colors represent the corresponding iso-surfaces of v=$|\breve{\mathbf{u}}|$. Only the ICA branch is shown. $M = 1125$, $Re = 350$, $Ws = 4.375$.

Based on the aforementioned observations we draw the following conclusions:

- The energy spectra of the velocity fields computed over the entire cycle do not reveal an intermittent laminar-turbulent regime.
- The transition to turbulence may be detected by analyzing the high-order temporal POD modes.
- The contribution of higher POD modes to the velocity field is not uniform in space, indicating *coexistence* of laminar and turbulent states.

5.3.3 Time- and space-window POD

The overall conclusion of the previous section is that POD analysis performed over a complete cardiac cycle and over the entire computational domain is inadequate to characterize

space-intermittent turbulence in the transient flow regime. To this end, we suggest an alternative approach to analyze the intermittent and mixed flow regimes. Our goal is to measure the turbulent energy of flow in different *regions* of the domain and over different *time intervals*. In order to *quantify* the intensity of the turbulent flow we perform *windowed-POD* analysis in time and space. By analyzing the behavior of the temporal modes we may determine time intervals within which the flow is turbulent. Then, by performing POD analysis at different sub-domains, we can focus on regions where the energy associated with higher POD modes is significant. To characterize transitional flow we divide the cardiac cycle into eight *time-window* intervals denoted by $a \div h$ as illustrated in figure 5.2; the time-windows have been chosen to represent different stages of the transient regime. We will refer to the time-windows as TWa÷TWh.

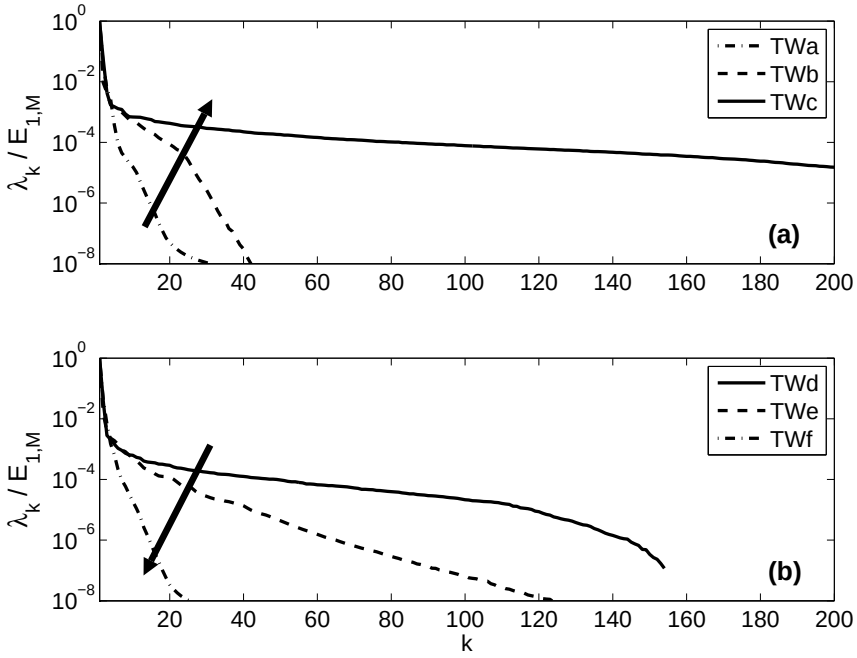


Figure 5.9: POD: Eigenspectra obtained over different time-windows (see figure 5.2).

In figure 5.9 we plot the POD eigenspectra computed over six consecutive time-windows; the spectrum slope provides an indication of a turbulent or laminar regime. The spectra distinguish clearly the presence of transition from laminar to turbulent regime shown in figure 5.9a and followed by re-laminarization in figure 5.9b. The arrows in figure 5.9 denote the spectrum evolution in time: transition to turbulence is denoted by an upward-directed arrow, the downward-directed arrow refers to re-laminarization. The spectra obtained over

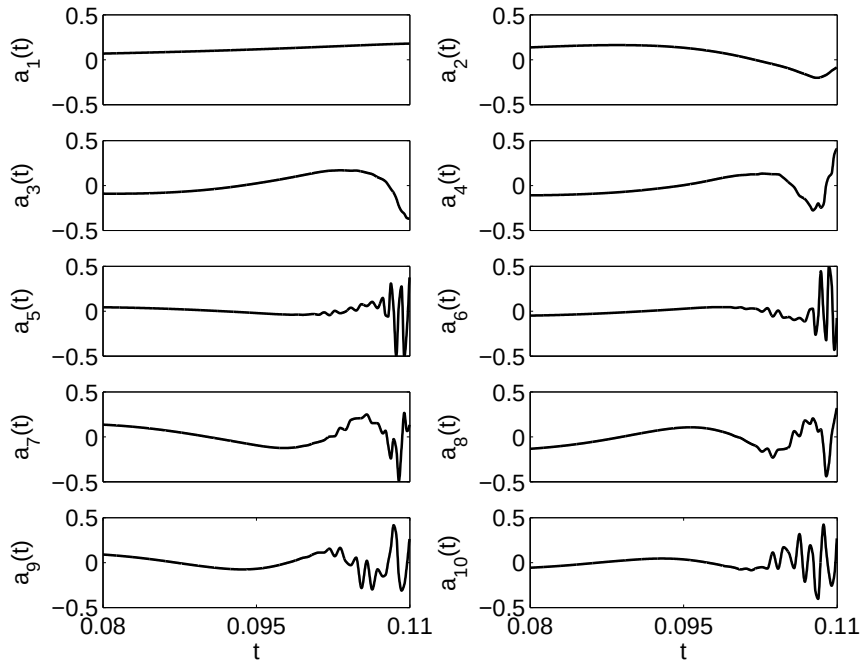


Figure 5.10: POD temporal modes $a_i(t), \quad i = 1, ..., 10$ corresponding to the time window TWb (see figure 5.2).

the TWc and TWd time-windows, covering the transitional regime, display a slow decay. The TWf and TWa time-windows belong to an end-diastole phase of the cardiac cycle; they are very similar and show a fast decay featuring the laminar regime. The results in figure 5.9 present a sequence of transient events in the post-stenotic flow, from laminar to turbulent, reverting towards laminar, back again towards turbulent, and so on. To refine the POD analysis, in order to capture the transition to turbulence, we analyze individual temporal POD modes within a certain time-window. The first ten temporal modes corresponding to TWb are plotted in figure 5.10. The appearance of high frequency components at certain time intervals signifies the onset of turbulence.

Time-window POD analysis applied to transient flows reveals significantly more information than the full-cycle analysis. Moreover, unlike the complete cycle POD analysis, it is less expensive from the computational point of view. However, time-window POD is not sufficient to detect and characterize the spatially intermittent distribution of kinetic (turbulent) energy. As we have already observed, turbulent flow is not present over the entire domain of the stenosed carotid artery (see figure 5.8(right)). Although visualization of certain POD modes helps to capture regions with high turbulent energy, it is not sufficient to

quantify turbulence in each region, and alternative approaches should be applied. To this end, we employ *space-window* POD to detect regions with high kinetic energy. We analyze the eigenspectrum in three sub-domains of the ICA shown in figure 5.11: 1) the stenosis throat (sub-domain AB); 2) the post-stenotic region, from 12 to 22 mm downstream of the stenosis throat (sub domain CD); and 3) the post-stenotic region, from 32 to 42 mm downstream of the stenosis throat (sub domain EF).

Figure 5.11(a,b) shows fast decay of the POD eigenspectra computed in sub-domain AB where no turbulence was detected. In figure 5.11(c,d) we plot the spectra computed in sub-domain CD. The energy spectra in figure 5.11c reveal onset of turbulence and subsequent flow re-laminarization. The spectra obtained over TWc and TWd depict slow decay, typical for turbulent flow. The POD spectra in figure 5.11(e,f) show the same scenario of transition/re-laminrization although the turbulence here is very weak because it experiences a decay and eventually re-laminarization downstream of the stenosis (compare TWc and TWd curves in plots (e,f) with those in plots (c,d)).

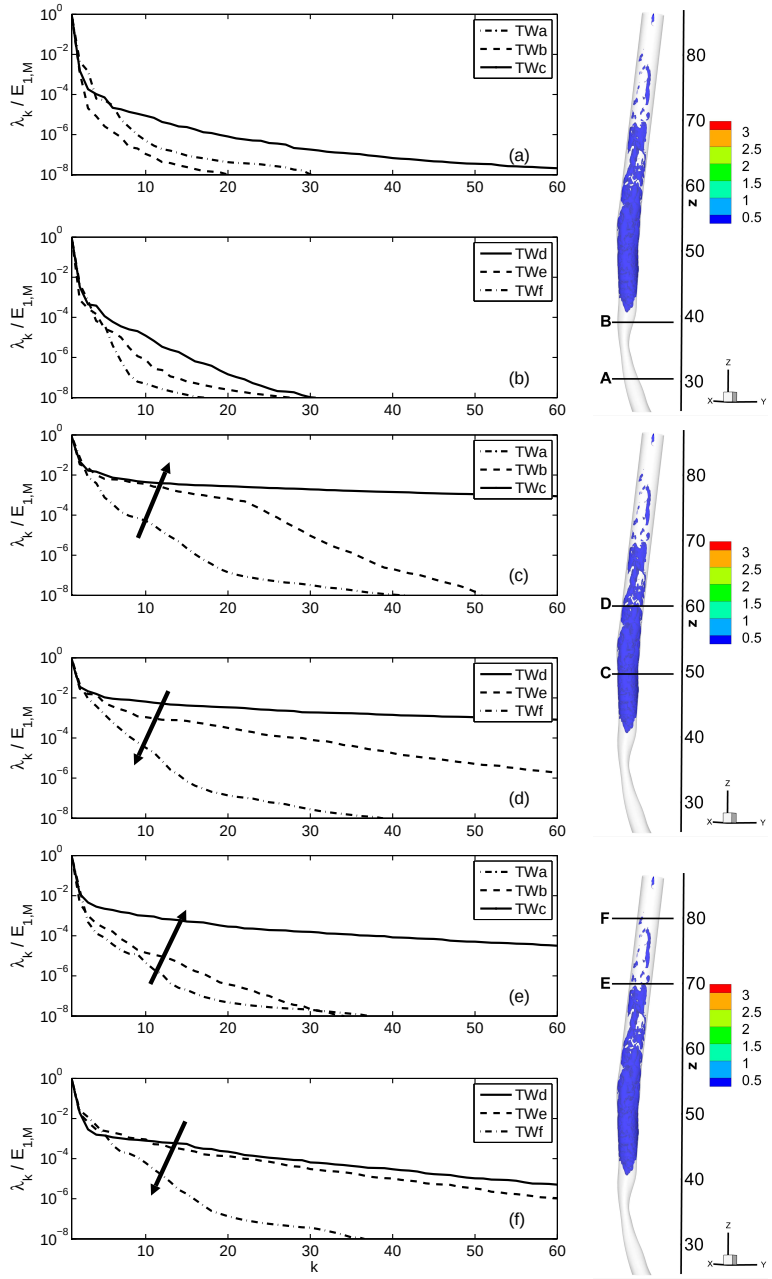


Figure 5.11: POD eigenspectra obtained over different time-windows in *sub-domains* AB, CD and EF (right): (a,c,e) - time-windows TWa, TWb and TWc (flow acceleration and transition to turbulence); (b,d,f) - time-windows TWd, TWe and TWf (flow deceleration and laminarization); (arrows show the time growth, color represents the w-iso-surface reconstructed from POD modes 20 to 50 at $t = 0.13$).

5.4 Utility of POD in clinical setting

Conventional MRI, PC-MRI and CDUS measurements provide two-dimensional images. In this section we synthesize a collection of medical images by extracting them from CFD simulation data. Specifically, the velocity field computed in ICA is extracted from 2D slices (space-windows), and then time-window POD analysis is performed. First we examine data sampled with high temporal resolution and subsequently we compare the results with those obtained with *reduced* sampling rates.

The velocity field computed in ICA is extracted along 2D slices (space-windows), and then time-window POD analysis is performed. In figure 5.12 we show spectra computed over different time-windows using velocity fields obtained from a transverse to the main flow 2D slice at $z = 60$. Similarly to 3D POD analysis (see figures 5.11c and d)), these POD spectra reveal transient flow regimes shown by arrows. POD spectra were also computed along longitudinal cross-sections mimicking an ultrasound image. In figure 5.12(iv-vi) we plot spectra obtained over different time-windows with a longitudinal slice located in the sub-domain CD. Comparison with the spectra obtained by high-resolution 3D POD analysis shows remarkable similarity (see figures 5.11c and 5.11d.)

To quantify the *quasi-instantaneous* decay of the POD eigenvalues we propose the following procedure. We recall that turbulence is associated with existence of the high POD modes that exhibit a power law energy decay, namely

$$\lambda_k \sim k^{-s(t)}. \tag{5.2}$$

We extract flow field data from different planes, as shown in figure 5.12. For each time instant t over the cardiac cycle, the POD analysis is performed over a relatively short time-window $t - \Delta t'/2 < t < t + \Delta t'/2$, where $\Delta t'$ was approximately 0.01 second at the systolic peak and approximately 0.1 second during the diastolic phase. Taking advantage of the high time resolution of our simulations, each time interval $\Delta t'$ was covered by 80 snapshots. The exponent $s(t)$ has been computed by employing the POD procedure used to obtain the eigenspectrum shown in figure 5.7. In figure 5.13(top) we plot the exponent $s(t)$ of the POD eigenvalues. The double hump curves clearly indicate the transient nature of the flow. The low values of the slope ($0.8 < s < 1.1$) correspond to the turbulent regime that occurs during the systolic phase. The secondary turbulence regime at $t \approx 0.55$ mentioned above is

also captured by the low slope values in figure 5.13(top).

Our CFD simulations were carried out with very high time resolution, allowing accurate estimation of the POD eigenspectrum. At the present time, there are serious limitations in the spatial and temporal resolution of MRI and CDUS imaging. MRI is a 3D technique for velocity imaging with high spatial resolution but low temporal resolution. CDUS images of a carotid artery are two-dimensional with time resolution of about 20-50 Hz. However, the sampling rate of the US devices is expected to grow significantly due to the introduction of parallel data processing. For example, owing to massive parallel beam formation, much higher frame rates can be achieved with a new cardiovascular platform SC2000 system launched recently by Siemens [8]. High temporal resolution of 481Hz in US imaging (sample rate) is reported in [102]. We note that several artifacts and noise contamination also occur during medical scanning, which may increase uncertainty in the data processing and lead to erroneous interpretations. As a first step, the artifacts and noise should be, if possible, identified and treated prior to using medical images for acquisition of clinical data.

To estimate if the suggested procedure can be used with relatively low time resolution imaging equipment we performed the following computations. We selected 800 time instants throughout the cycle and assumed that each time instant τ corresponds to the middle of a time-window $\Delta\tau = [\tau_1, \tau_M]$, namely $\tau = (\tau_M + \tau_1)/2$. Here M is the number of snapshots within the time-window $\Delta\tau$ that depends on the imaging equipment time resolution. Specifically, we used $\Delta\tau \approx 0.12$ sec, and, consequently, M=100 and 13 corresponding to 825 Hz and 107 Hz samplings, respectively. For each time-window, we computed the POD eigenspectra with different number of snapshots corresponding to different time resolutions. The decay rate $s(\tau)$ is shown in figure 5.13(bottom). We recall, that the double hump shape of the $s(\tau)$ curve points to the transient turbulent regime. From figure 5.13(bottom), all spectra, even that obtained with 107 Hz, are representative of transitional regime. A certain drawback of the low resolution data is that $s(\tau)$ curve shows late transition and early re-laminarization. For example, the regime at $\tau = 0.11$ is turbulent, as we have seen in the previous sections, however, the corresponding time-window, $\Delta\tau = [0.05, 0.17]$, covers the laminar part of the cycle $0.05 < t < 0.09$. For the 107 Hz sampling, the number of snapshots within an interval $\Delta\tau$ is only 13 which means that the sampling includes only few snapshots of the turbulent state. The same considerations explain the early re-laminarization at $\tau = 0.6$. On the other hand, the decay rate computed at $\tau = 0.11$ with 206 Hz data

shows good agreement with that computed with 825 Hz data although it also predicts a slight delay of transition. Analyzing the $s(\tau)$ curves in figure 5.13(bottom) and taking into account the rapid growth of the medical imaging equipment time resolution, we can be optimistic that the windowed-POD technique can be applied to analyze transient turbulence in stenosed vessels in the not-distant future.

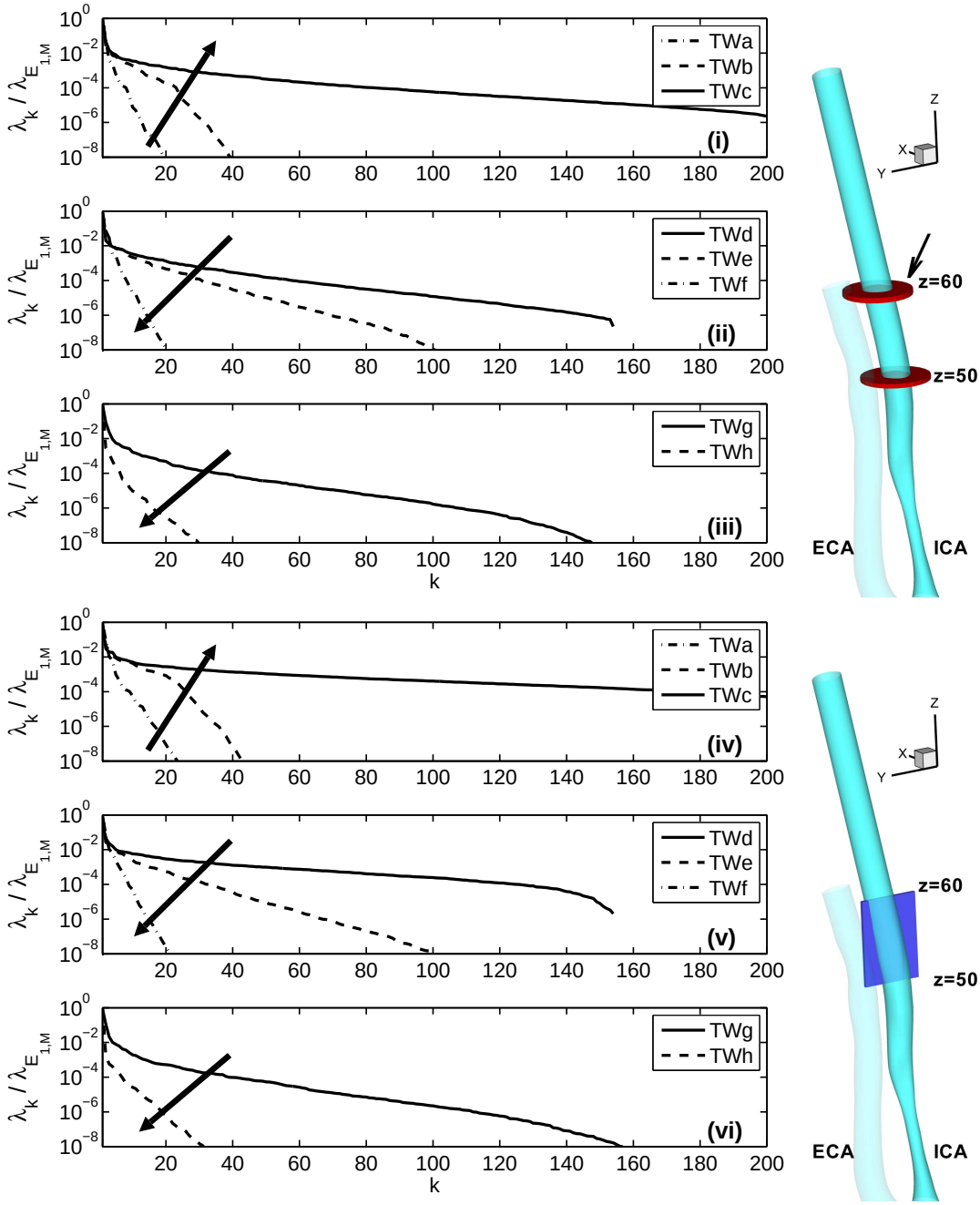


Figure 5.12: 2D POD: eigenspectra obtained over different time intervals (see figure 5.2). (i-iii): velocity field is extracted at $z = 60$; (iv - vi) Velocity field is extracted on a slice with $x = const$, between $z = 50$ and $z = 60$; (i,iv) - time-windows TWa, TWb and TWc (flow acceleration, and transition to turbulence); (ii,v) - time-windows TWd, TWe and TWf (flow deceleration, and laminarization); (iii,vi) - time-windows TWg and TWh (diastole phase); arrows show the time growth.

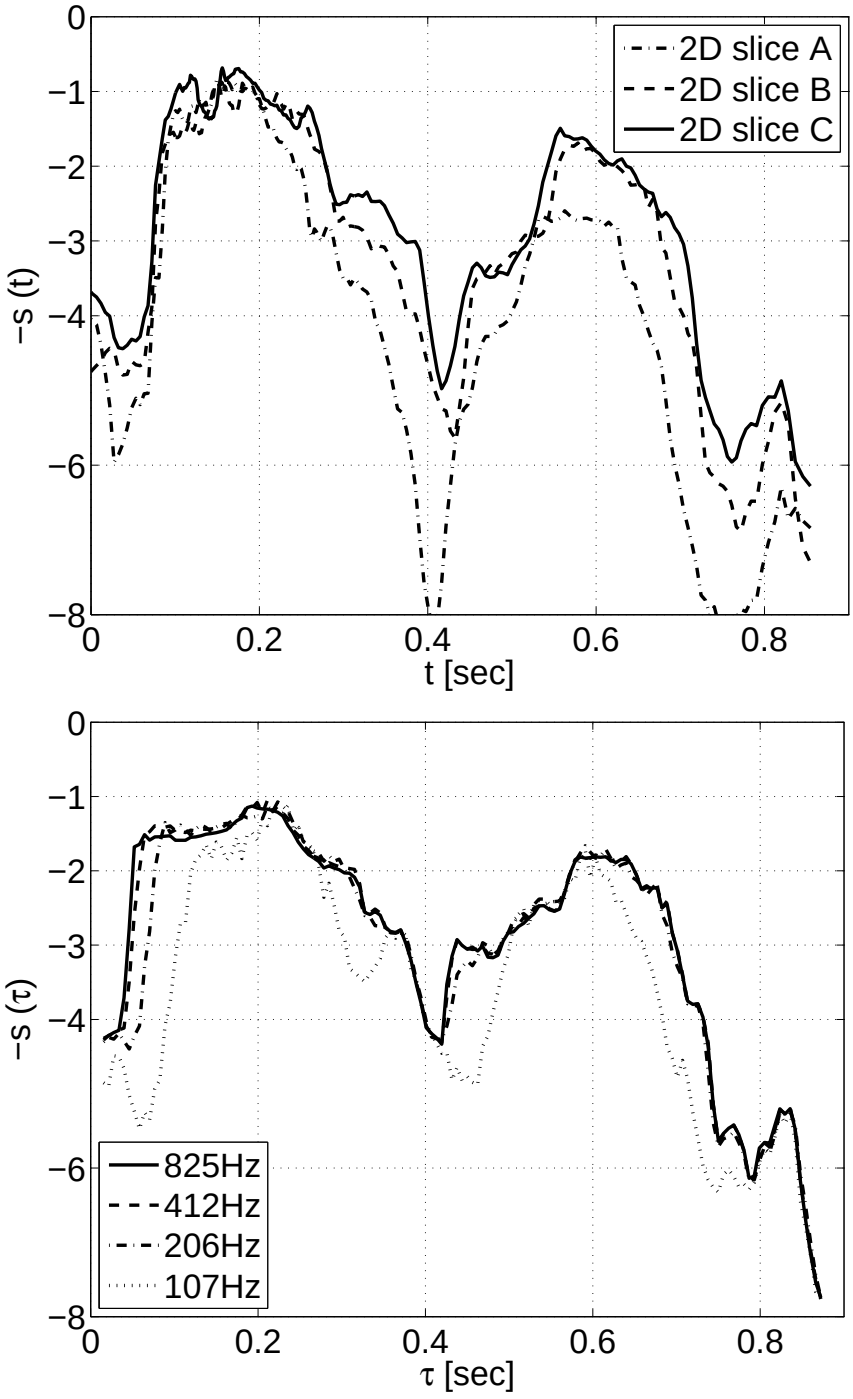


Figure 5.13: 2D POD. Top: decay rate of POD eigenspectra. Data are extracted along: slice $z = 50$ (2D slice A), slice $z = 60$ (2D slice B) and slice with $x = const$ and located between $z = 50$ and $z = 60$ (2D slice C) shown illustrated in figure 5.12. $s(t)$ is computed for the modes $k = 2 \div 10$. Bottom: decay rate of POD eigenspectra, computed with variable temporal resolution. Data are extracted along the slice $z = 60$; $s(\tau)$ is computed for the modes $k = 2 \div 10$.

5.5 Discussion and outlook

High-resolution simulations to study transitional flow in a stenosed carotid artery alternating between laminar and turbulent states has been performed. The high degree of narrowing in the ICA (about 77%) creates a jet flow, which attaches to the wall of the artery and at certain (instantaneous) value of the Reynolds number becomes unstable leading to the onset of turbulence. In particular, a mixed flow state is characterized by regions of unsteady laminar flow and a sub region of turbulence, starting downstream of the stenosis and extending about five to six centimeters farther downstream. As the Reynolds number decreases, flow re-laminarization in this sub-region takes place.

Studies of transitional flow in stenosed vessels with axi-symmetric and eccentric narrowing presented in [77, 9, 95] suggest that the transition to turbulent flow appears at Reynolds number $Re > 500$ and Womersley number $Ws > 7.5$, i.e., at values higher than the ones in our simulations; however, this is due to the geometric complexity. For example, Varghese *et. al.* [95] observed that introducing slight eccentricity to the stenosis led to intermittent laminar-turbulent flow, which was not the case for flow in a pipe with an axi-symmetric constriction. This finding implies that due to arterial *geometric variability* among different patients with a stenosed ICA, transition to a turbulent state *may or may not* take place even if these patients have the same degree of stenosis or similar heart rate. Moreover, geometrical modifications, e.g., arterial deformations due to neck bending or twisting may trigger the onset of turbulence.

The specific focus of our work is the *quantification* of transitional blood flow. To this end, we applied Proper Orthogonal Decomposition to cross-correlate flow fields at different times; laminar flows exhibit high degree of correlation while turbulent flows show lower correlation. The degree of correlation between velocity fields from different times can be measured by the rate of decay of the eigenvalues of the correlation matrix. The *value* of POD analysis of blood flow is in identification and characterization of arterial segments with high turbulent energy and hence of associated pathologies. If the region of turbulent flow is very small compared to the size of the full domain considered, analysis of POD spectra or inspection of temporal modes performed on the entire domain is insufficient. To this end, we developed a time- and space-window POD version that can quantify precisely the kinetic energy associated with different POD modes. In CFD simulations this can be done with

either 3D or 2D sub domains while in the clinical setting 2D images obtained via PC-MRI or CDUS can be used. This approach can be compared to the window-FFT method or the wavelet decomposition, where a frequency spectrum is analyzed over a certain time interval.

The window-POD procedure is computationally favorable and can be carried out very fast using a standard laptop. Indeed, the three-dimensional POD analysis of turbulent flow in a relatively large computational domain is computationally expensive. For example, in our study the data was computed at 24 million quadrature points. The computational work associated with POD (numerical integration) is not very high, however, the memory requirements are quite substantial. In our simulation the estimated memory requirement for POD analysis in the entire domain and over a one cardiac cycle was $400GB$. POD analysis of 1125 snapshots required about 60 minutes on 128 AMD Opteron 2.6MHz CPUs of the CRAY XT3 computer with $4GB$ memory per processor; this time also includes the input of velocity and pressure fields and output of data for subsequent visualization. However, *window-POD* analysis is computationally much more favorable and can be performed on a standard desktop computer in about five minutes.

The POD method for analysis of transitional flow in arteries has some limitations. The *accuracy* of POD analysis as clinical diagnostic tool depends on the spatio-temporal resolution of PC-MRI or CDUS, which at the present time may be sufficient to adequately capture laminar flows but it is not clear that it can capture mixed or fully turbulent regimes. In future work, we are planning to investigate this important issue using clinical data. Also, in the present work we have applied the utility of the time-space window POD method to analyze flow in *rigid* arteries. Although, arterial walls affected by atherosclerosis are less elastic than the healthy ones, POD should be extended to analyze flow in flexible vessels. Of interest would be how the transition scenario described in the present work is modified by the arterial wall movement.

Chapter 6

Concluding Remarks

The work described in this Thesis have been inspired by the Physiological Virtual Human initiative and by recent advances in High Performance Computing (HPC) infrastructure, specifically the transition into the new era of Petaflop computing and the development of the Grid technology. Specifically, we attempted to bridge between the worlds of HPC and medicine, by developing new robust numerical methods and efficient parallel algorithms to simulate blood flow in the human arterial tree. In particular we have achieved the following results:

- We have investigated performance of two tensor-product spectral bases defined on a triangular element. We concluded that the cartesian tensor-product bases are more robust than the barycentric bases, in part due to better sparsity patterns of the corresponding linear operators. However, use of barycentric tensor bases was advantageous in solving diffusion problem, defined on a computational domain discretized with distorted elements (with extremely high aspect ration).

- We have compared performance of three sets of quadrature grids for triangular element, and we have arrived to the following conclusions: The collapsed-cartesian grid is overall the most efficient, particularly for high order polynomial expansions. The stability properties are nearly independent of the grid types; however, the performance of barycentric grids is slightly better when the size of a timestep approaches its critical value. In solution of nonlinear problems with Galerkin projection the loss of accuracy can be a result of underresolution in both numerical integration and differentiation.

- We focused on the numerical accuracy and filtering of high-frequency pressure os-

cillations arising due to singularities and explicit treatment of pressure boundary condition. Robustness of a semi-implicit and implicit numerical schemes have been compared. We have observed that implicit numerical scheme for solution of Navier-Stokes equations does not posses time-splitting error, and effectively removes erroneous pressure oscillations. The use of semi-implicit scheme requires filtering the pressure oscillations by lower-order polynomial approximation for the pressure, that is employing $P/P-1$ or $P/P-2$ - approach. The later approach results in loss of numerical accuracy in the problems where solution is smooth.

- We have developed and implemented new computational and numerical approaches for solving Navier-Stokes equations to simulate a blood flow in very complex (patient-specific) geometry. The *new methods* developed in the course of this Thesis are:
  a) the two-level domain decomposition, featuring the continuous and discontinuous-like Galerkin projection. The computational domain is first sub-divided into several large patches, and then the standard SEM discretization and domain partitioning is performed withing each patch. We considered the non-overlapping and overlapping patches. The use of overlapping sub-domains enhances numerical accuracy and stability.
  b) time-dependent resistance-capacitance outflow boundary condition ($R(t)C$) allowing to control effectively the flow distribution in arterial networks with multiple outlets, and also to integrate in a straightforward way clinically measured flowrates in numerical simulations.
  c) multi-level communicating interface (MCI). The MCI has been tested in simulations on distributed computers and optimized for high latency networks. Later MCI was successfully implemented for simulations on a single cluster.

- We have increased the scalability of high-order spectral/*hp*-element Navier-Stokes solver $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$ by redesigning parallel algorithms for coarse space linear vertex solver. The code scales now on several thousands of processors.

- We have developed a simple but extremely efficient and embarrassingly parallel accelerator for iterative solution of a system of linear equations for dynamical systems. The effectiveness of the method does not depend on spatial discretization (grid independent), and our numerical experiments indicate that the method becomes even

more efficient for more refined spatial discretization.

- We have performed the first high-resolution numerical simulations of unsteady flow in tens of brain arteries, including the Circle of Willis. The simulations revealed high-degree of flow three-dimensionality even in the narrow vessels.

- We have performed comparative study of the 1D and 3D models for simulation of a blood flow dynamics in complex arterial networks.

- We have suggested a new methodology (time- space-window POD analysis) for analysis of intermittent in time and space laminar-turbulent flow. The new method has been applied for analysis of intermittent flow in stenosed carotid artery.

There is a lot more to be done to take the numerical simulations to the state where they can be of truly predictive value in the framework of healthcare. Achieving high-order numerical accuracy and reasonable solution time allows to isolate the discretization errors and concentrate on the modeling. There are various models employed for flow simulations, e.g., 1D models, 3D models with rigid and elastic walls, etc, and it is crucial to understand under what circumstances each model can (or can not) be applied. The multiscale nature of the human organism also suggests multiscale approach for numerical simulations. The model validation and uncertainty quantification have the utmost importance in modeling physiological processes.

There are also specific tasks that should be performed to leverage the new numerical and computational approaches described in this Thesis. For example, developing fully implicit solver for Navier-Stokes equations which will integrate the two-level domain decomposition and the convergence accelerations strategies. Solving Navier-Stokes equations implicitly will relax the restrictions on the size of the time step, and application of POD-based acceleration technique might be advantageous over the simple extrapolation method. Low iteration count required for solution of linear systems enhances scalability of a solver, and at the same time high-light the importance of better load-balancing in tasks such as computing boundary conditions. Another very important issue that has no satisfactory solution so far is how to take advantage of the multi-core architecture. This can be adequately addressed by using new dynamic programming languages, e.g. Unified Parallel C (UPC), or by extensive use of shared memory optimizations.

Appendix A

Construction of the elemental Transformation Matrix R

We recall that

$$\mathbf{S_2} = \mathbf{R}\mathbf{S_1}\mathbf{R}^{\mathbf{T}}$$

and consider a single elemental matrix and the transformation of basis which arises from a matrix $\mathbf{R}$ of the form:

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{R_{ve}} & \mathbf{R_{vf}} \\ 0 & \mathbf{I} & \mathbf{R_{ef}} \\ 0 & 0 & \mathbf{I} \end{bmatrix},$$

where the the vertex modes are listed first followed by the edge and the face modes. The matrices $\mathbf{R_{ve}}, \mathbf{R_{vf}}$ represent the modification of the vertex modes by the edges and face modes. Similarly the matrix $\mathbf{R_{ef}}$ represents the modification of the edge modes by the face modes. The transformation matrix has upper triangular structure and hence it is easily invertible.

Let us define $\mathbf{R}$ as

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{R_v} \\ 0 & \mathbf{A} \end{bmatrix} \qquad \text{where} \qquad \mathbf{R_v} = \begin{bmatrix} \mathbf{R_{ve}} & \mathbf{R_{vf}} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{R_{ef}} \\ 0 & \mathbf{I} \end{bmatrix}$$

and write the original Schur complement of the Helmholtz matrix as

$$\mathbf{S_1} = \begin{bmatrix} \mathbf{S_{vv}} & \mathbf{S_{v,ef}} \\ \mathbf{S_{v,ef}^T} & \mathbf{S_{ef,ef}} \end{bmatrix} = \begin{bmatrix} \mathbf{S_{vv}} & \mathbf{S_{ve}} & \mathbf{S_{vf}} \\ \mathbf{S_{ve}^T} & \mathbf{S_{ee}} & \mathbf{S_{ef}} \\ \mathbf{S_{vf}^T} & \mathbf{S_{ef}^T} & \mathbf{S_{ff}} \end{bmatrix}$$

then applying the transformation matrix, namely $\mathbf{S_2} = \mathbf{R S_1 R^T}$ we obtain

$$\mathbf{S_2} = \begin{bmatrix} \mathbf{S_{vv}} + \mathbf{R_v S_{v,ef}} + \mathbf{S_{v,ef} R_v^T} + \mathbf{R_v S_{ef,ef} R_v^T} & [\mathbf{S_{v,ef}} + \mathbf{R_v^T S_{ef,ef}}]\mathbf{A^T} \\ \mathbf{A}[\mathbf{S_{v,ef}} + \mathbf{S_{ef,ef} R_v^T}] & \mathbf{A S_{ef,ef} A^T} \end{bmatrix} \tag{A.1}$$

where

$$\mathbf{A S_{ef,ef} A^T} = \begin{bmatrix} \mathbf{S_{ee}} + \mathbf{R_{vf} S_{ef}} + \mathbf{S_{ef} R_{vf}^T} + \mathbf{R_{vf} S_{ff} R_{vf}^T} & \mathbf{S_{ef}} + \mathbf{R_{vf}^T S_{ff}} \\ \mathbf{S_{ef}} + \mathbf{S_{ff} R_{vf}^T} & \mathbf{S_{ff}} \end{bmatrix}. \tag{A.2}$$

In order to completely orthogonalise the vertex modes with the edge and face modes we require that

$$\mathbf{R_v^T} = -\mathbf{S_{ef,ef}^{-1} S_{v,ef}^T}. \tag{A.3}$$

To decouple the edge modes from the face modes we see from inspecting (A.2) that

$$\mathbf{R_{ef}^T} = -\mathbf{S_{ff}^{-1} S_{ef}^T}. \tag{A.4}$$

Appendix B

Parallel matrix vector multiplication in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$

```
// z - vector of length Rank(V_SC) to store the forcing term and then
//     the solution vector.
// Js - global index of first column of V_SC stored on this partition.
// Je - global index of  last column of V_SC stored on this partition.
// tmp_buf - temporary buffer


  /* compute x = [A]y  */
  gather_y_local(z,y_local,VERSION);
  do_local_matrix_vector_mult(A_local,y_local,x_local);
  scatter_x_local(x_local,z);


void gather_y_local(double *z, double *y_local, int VERSION){

    switch(VERSION){
    case V1:
      MPI_Allreduce(z,tmp_buf,length(z),MPI_DOUBLE,MPI_SUM,MPI_COMM_WORLD);
      memcpy(y_local,tmp_buf+Js,(Je-Js+1)*sizeof(double));
```

```
break;

case V2:
  MPI_Allreduce(z+Js,tmp_buf,Je-Js+1,
                MPI_DOUBLE,MPI_SUM,MPI_COMM_ROW);
  MPI_Allreduce(tmp_buf,y_local,Je-Js+1,
                MPI_DOUBLE,MPI_SUM,MPI_COMM_COLUMN);
break;

case V3:
  for (np = 0; np < size(MPI_COMM_ROW); ++np){
    for (i = 0, k = 0; i < length(z); ++i){
      if (Js <= global_index_z[i] <= Je){
        if (z[i] != 0)
           sendbuf[np][k++] = z[i];
      }
    }
  }
  MPI_Alltoallv(sendbuf,length_of_sendbufs,sdispls,MPI_DOUBLE,
                recvbuf,length_of_recvbufs,rdispls,MPI_DOUBLE,
                MPI_COMM_ROW);
  for (np = 0; np < size(MPI_COMM_ROW); ++np)
    map_recvbuf_to_y_local(recvbuf[np],tmp_buf);
  MPI_Allreduce(tmp_buf,y_local,(Je-Js+1),
                MPI_DOUBLE,MPI_SUM,MPI_COMM_COLUMN);
break;

case V4:
  for (np = 0; np < size(MPI_COMM_ROW); ++np)
     MPI_Irecv(recvbuf[np],count[np],MPI_DOUBLE,
               np,tag,MPI_COMM_ROW,rqst_rcv[np]);
  for (np = 0; np < size(MPI_COMM_ROW); ++np){
```

```
            for (i = 0, k = 0; i < length(z); ++i){
               if (Js <= global_index_z[i] <= Je){
                  if (z[i] != 0)
                     sendbuf[np][k++] = z[i];
               }
            }
            MPI_Isend(sendbuf[np],length(sendbuf[np]),MPI_DOUBLE,
                      np,tag,MPI_COMM_ROW,rqst_snd[np]);
        }
        for (np = 0; np < size(MPI_COMM_ROW); ++np){
          MPI_Waitany(np,rqst_rcv,&index,status);
          map_recvbuf_to_y_local(recvbuf[index],tmp_buf);
        }
        MPI_Allreduce(tmp_buf,y_local,(Je-Js+1),
                      MPI_DOUBLE,MPI_SUM,MPI_COMM_COLUMN);
    break;
    }//end of switch
}
```

Appendix C

resistance - flow rate relationship in a network of five vessels

In the following we derive the resistance - flow rate relationship in a **network of five vessels**, shown in figure C.1.

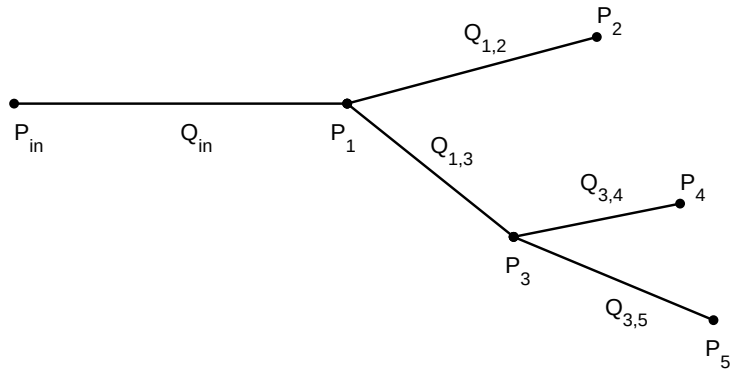


Figure C.1: Sketch of 1D model of five arteries: one inlet and five outlets.

The $Q_{1,2}/Q_{1,3}$ ratio is given by

$$\frac{Q_{1,2}}{Q_{1,3}} = \frac{R_3 - L_{1,3}K_{1,3}}{R_2 - L_{1,2}K_{1,2}}. \tag{C.1}$$

Thus,

$$R_3 = L_{1,3}K_{1,3} + \frac{Q_{1,2}}{Q_{1,3}}(R_2 - L_{1,2}K_{1,2}). \tag{C.2}$$

Now we apply (4.6) to segments $3, 5$ to obtain

$$P_5 - P_3 = L_{3,5}K_{3,5}Q_{3,5} \tag{C.3}$$

and using the resistance boundary condition $P_j = R_jQ_j$, we get

$$R_5Q_{3,5} - R_3Q_{1,3} = L_{3,5}K_{3,5}Q_{3,5}. \tag{C.4}$$

Next, we substitute R_3 in (C.4) by formula (C.2)

$$R_5Q_{3,5} - Q_{1,3}(L_{1,3}K_{1,3} + \frac{Q_{1,2}}{Q_{1,3}}(R_2 - L_{1,2}K_{1,2})) = L_{3,5}K_{3,5}Q_{3,5} \tag{C.5}$$

and, assuming that $|L_jK_j| \ll 1$ and $|L_jK_j| \ll R_j$, we obtain

$$R_5Q_{3,5} - R_2Q_{1,2} = L_{3,5}K_{3,5}Q_{3,5} \approx 0. \tag{C.6}$$

From equation (C.6) we obtain

$$\frac{Q_{3,5}}{Q_{1,2}} \approx \frac{R_2}{R_5}.$$

Similarly, we can obtain

$$\frac{Q_{3,4}}{Q_{1,2}} \approx \frac{R_2}{R_4}.$$

So far we derived the Q_i/Q_j relations for a steady flow but similarly to the previous case, we can derive the $\hat{Q}_{i,k}/\hat{Q}_{j,k}$ relations for unsteady flow and show that

$$\frac{\hat{Q}_{i,k}}{\hat{Q}_{j,k}} \approx \frac{R_j}{R_i}$$

and therefore

$$\frac{Q_i(t)}{Q_j(t)} \approx \frac{R_j}{R_i}.$$

By neglecting friction we can extend the $R-Q$ relation to the network with an arbitrary number of segments. For example, using the model provided in figure C.1 we obtain:

$$R_1Q_{in} \approx R_2Q_{1,2} \approx R_3Q_{1,3} \approx R_5Q_{3,5} \approx R_4Q_{3,4}...$$

Appendix D

class TerminalBoundary

```
class TerminalBoundary{
  private:
  public:
  char type;                //outlet: type = 'O', inlet: type = 'I'
  int NBoundaryFaces;       //number of elements facing inlet/outlet
                            //in this partition
  int *BoundaryFaceList;    //index of matching element,
                            //face and index of matching
                            //face from partner's boundary
                            // dimension [NBoundaryElements*3]

  int *BoundaryElementList_local_uvw,
      *BoundaryElementList_local_p;

  double Area; // area of the boundary

  int ID_local;   // local to patch index of artery
  int ID_global;  // global (unique) index of artery

  MPI_Comm  TerminalBndr_communicator; //L4 communicator
  int  TerminalBndr_communicator_size; //size of L4
```

```
  int  TerminalBndr_communicator_rank; //rank in L4

  //functions
  int  set_up_TerminalBndr(Domain *Omega, char **standard_labels);
  double compute_flow_rate(Domain *Omega);
  double compute_mean_pressure(Domain *Omega);
};
```

Appendix E

class OvrLapBoundary

```
class OvrLapBoundary{
  private:
  public:
  char type;                    // outlet: type = 'O', inlet: type = 'I'
  int NBoundaryElements;        //number of elements facing overlapping
                                //outlet in this partition
  int *BoundaryElementList;     //index of matching element,
                                //face and index of matching
                                //face from partner's boundary
                                // dimension [NBoundaryElements*3]

  int NBoundaryElements_global;  //total number of elements
                                 //facing overlapping outlet.
  int *NBoundaryElements_per_rank;

  int Nqp_total_FrPr;           // total number of quadrature points
                                // required by partner
  int *Nqp_per_face_global_FrPr; // number of quadrature points per face
                                // required by partner
  double **BoundaryFace_normal;  // normal to face  pointing outward
```

```
  int ID_local;     // local to patch index of artery
  int ID_global;    // global (unique) index of artery

  MPI_Comm  OvrLapBndr_communicator;      //L4 communicator
  int       OvrLapBndr_communicator_size; //size of L4
  int       OvrLapBndr_communicator_rank; //rank in L4
  int       Partner_rank_in_comm_world;   //rank in L1

  MPI_Request  request_send, request_send_dUVWdn;
  int   N_Vert_modes2transfer,     /* Number of vertex modes to transfer =1 */
        N_Edge_modes2transfer,     /* Number of edge modes to transfer      */
        N_Face_modes2transfer,     /* Number of face modes to transfer      */
        N_modes2transfer_total;    /* Total number of modes to transfer     */

  //storage for velocity, pressure and flux values:
  double **UVW_OLbndry,**UVW_OLbndry_global;
  double **P_OLbndry, **P_OLbndry_global;
  double **dUVWdn_global_FrPr;

  double ***HA, ***HB;  // interpolation operator

  void allocate_UVWP();
  void constract_interp_oper_for_dUVWdn(Domain *Omega);
  void prepare_BC_uvw2transferALLmodes(Domain *Omega);
  void transfer_uvw_3D_ALL_send();
  void prepare_BC_p2transferALLmodes(Domain *Omega);
  void transfer_p_3D_ALL_send();
  void prepare_dUVWdn(Domain *Omega);
  void transfer_dUVWdn_3D_ALL_send();
};
```

Appendix F

class MergingBoundary

```
class MergingBoundary{
  private:
  public:
  char type;              //outlet: type = 'O', inlet: type = 'I'

  int NBoundaryFaces;     //number of elements facing inlet/outlet
                          //in this partition
  int *BoundaryFaceList;  //index of matching element,
                          //face and index of matching
                          //face from partner's boundary
                          // dimension [NBoundaryElements*3]

  int *BoundaryElementList_local_uvw,  //list of IDs of elements
      *BoundaryElementList_local_p;    //inlet or outlet for
                                       //velocity and pressure systems

  double Area; // area of the boundary
  int Nqp;     // Total number of quadrature points
               // at merging faces in this partition
  int *Nqp_per_face; //Number of quadrature points per face
```

```
/* root of L4 needs global info */
int NBoundaryFaces_global; //total number of elements facing inlet/outlet.
int *NBoundaryFaces_per_rank;
int Nqp_global;     // Total number of quadrature points at
                    // merging faces in all partition
int *Nqp_per_rank;  // number of quadrature points per partition
int *Nqp_per_face_global;
int ID_local;      // local to patch index of artery
int ID_global;     // global (unique) index of artery
MPI_Comm  MergingBndr_communicator;            //L4 communicator
MPI_Request request_recv, request_recv_dUVWdn;
int  MergingBndr_communicator_size; //size of L4
int  MergingBndr_communicator_rank; //rank in L4
int  Partner_rank_in_comm_world;   //rank in L1
int   N_Vert_modes2transfer,     /* Number of vertex modes to transfer =1 */
      N_Edge_modes2transfer,     /* Number of edge modes to transfer      */
      N_Face_modes2transfer,     /* Number of face modes to transfer      */
      N_modes2transfer_total;    /* Total number of modes to transfer     */

//storage for velocity, pressure and flux values:
double **UVW_MRbndry,**UVW_MRbndry_global;
double **P_MRbndry, **P_MRbndry_restart,
       **P_MRbndry_global, **P_MRbndry_global_n1;
double **dUVWdn,**dUVWdn_n1;
double *dUVWdn_global;

//storage for mapping vertex and edge DOF with respect
//to adjacent patch
int **mapping_vert, **mapping_edge, **signchange_edge;

//functions:
int  set_up_MergingBndr(Domain *Omega, char **standard_labels);
```

```
  void map_vertices(double *XYZ_FrPr, double *my_XYZ);
  void collect_XYZ_face1(Domain *Omega);
  void set_P_restart(Domain *Omega);
  void transfer_uvw_3D_ALL_recv(int ACTION);
  void update_bndry_uvw(Domain *Omega);
  void transfer_p_3D_ALL_recv(int ACTION);
  void update_bndry_p(Domain *Omega);
  void transfer_dUVWdn_3D_ALL_recv(int ACTION);
  void update_FLUX(Domain *Omega);
  double compute_flow_rate(Domain *Omega);
  double compute_mean_pressure(Domain *Omega);
};
```

Appendix G

Input files for $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}$

The first step in preparing input data for $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}$ is to create a mesh and subdivide it into overlapping or non-overlapping regions. The inter-patch interface must be a flat surface. The 2D mesh at the inter-patch interface must be confirming. Once a volume mesh for each 3D patch is created, the boundaries corresponding to different inlets and outlets must be labeled. It is important to create unique labels for different inlets and outlets. The second step in preparing the input file is very similar to preparing the file for $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$. Here we follow the methodology corresponding to converting an output file from Gridgen (volume mesh, with unique labeling for boundaries of the domain), and as an example we use mesh corresponding to CoW (the four patches of the domain as presented in figure 4.59(a). We start from the domain colored in orange, the patch has 3 inlets and 11 outlets and perform the following steps:

1. verify that the Gridgen (Version 15) output file (part1.inp) contains correct labeling for *all* inlets, outlets and walls. The header of the file looks like:

```
FIELDVIEW_Grids 3 0
!FVUNS V3.0 File exported by Gridgen 15.10R1 Fri Oct 19 19:42:09 2007
!FVREG file: E:\PROJECTS_ATREE\NECK_CRANIAL_ARTERY\PART_1j.inp.fvreg
Grids 1
Boundary Table 31
1 0 1 ICA_right
1 0 1 ICA_left
1 0 1 M1_L
```

```
1 0 1 out_M1_L
1 0 1 inl_ICA_R
1 0 1 inl_ICA_L
1 0 1 cerebral_R
1 0 1 out_cerebral_R
1 0 1 out_cerebral_L
1 0 1 cerebral_L
1 0 1 anter_cerebral_L
1 0 1 out_anter_cerebral_left
1 0 1 anter_cerebral_R
1 0 1 out_anter_cerebral_R
1 0 1 poster_cerebral_L
1 0 1 out1_poster_cerevb_L
1 0 1 tip_poster_cerebral2_L
1 0 1 out2_poster_cereb_L
1 0 1 M1_R
1 0 1 out_M1_R
1 0 1 ophtalmic_right
1 0 1 R_opth_tip
1 0 1 out_opthalmic_R
1 0 1 post_crbr2_R
1 0 1 out2_post_cerebral_R
1 0 1 poster_cereb_R
1 0 1 out1_poster_cerebral_R
1 0 1 basil_CoW
1 0 1 basilar
1 0 1 inl_basilar
1 0 1 CoW
Nodes 37638
```

the first three integers and simply part of the string of characters labeling the boundaries, they do not posses any additional information.

2. run converter utility that reformat the part1.inp file into “part1.rea” and “part1.info” files. Next we will work with the “part1.info” file. The default “part1.info” contains information that helps to identify all the boundaries of the domain. For example the part concerning with the outlet marked as “1 0 1 out_M1_L” appears as

```
1 0 1 out_M1_L
0 CURVED
a CURVE_TYPE
W TYPE
0 LINES
```

and with inlet marked as “1 0 1 inl_ICA_R” appears as

```
1 0 1 inl_ICA_R
0 CURVED
a CURVE_TYPE
W TYPE
0 LINES
```

It is user responsibility now to modify the context of the “part1.info” to comply with standards implemented in $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}}$. All inlets and outlets must be labeled with standard labels $bcA(x, y, z)$, $bcB(x, y, z)...bcZ(x, y, z)$, currently $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}}$ supports up to 26 inlets and 26 outlets, the numbering bust be consecutive, i.g., user must not skip $bcB(x, y, z)$ and label the second outlet as $bcC(x, y, z)$. The label $bcA(x, y, z)$ corresponds to inlet No. 1 and also for outlet No. 1, the label $bcB(x, y, z)$ corresponds to inlet No. 2 and also for outlet No. 2, and so on. In our example the modified part of the “part1.info” file will be

```
1 0 1 out_M1_L
0 CURVED
a CURVE_TYPE
O TYPE
0 LINES
INLINE
0. 0. 0. bcA(x,y,z)
```

and

```
1 0 1 inl_ICA_R
0 CURVED
a CURVE_TYPE
v TYPE
3 LINES
u = 1.0
v = 2.0
w = 3.0
INLINE
0. 0. 0. bcA(x,y,z)
```

Here we used some default boundary conditions for the velocity at the inlet "1 0 1 inl_ICA_R", user should set up boundary conditions according to rules of $\mathcal{NEKTAR}$.

3. When modification of the "part1.info" file is completed (**check it three times at least !!!**), the converter utility should be employed again, but this time using the modified "part1.info" file. By completion a new "part1.rea" file is created.

4. Next step is to make sure that the number of inlets and outlets is specified in the "part1.rea" file, which header is as follows:

```
****** PARAMETERS *****
GRIDGEN 3D -> NEKTAR
3 DIMENSIONAL RUN
15 PARAMETERS FOLLOW
0.01     KINVIS
4        MODES
0.001    DT
10000    NSTEPS
1        EQTYPE
2        INTYPE
1000     IOSTEP
```

```
10        HISSTEP
3         PRECON
11        NOUTLETS
3         NINLETS
1.0       DPSCAL
1.0       DVSCAL
4         NPODORDER
2         NPPODORDER
0  Lines of passive scalar data follows
0  LOGICAL SWITCHES FOLLOW
```

Here we focus only on some parameters: a) $NINLETS$ and $NOUTLETS$ - integers specifying the number of inlets and outlets, respectively. Parameters $NPODORDER$ and $NPPODORDER$ specify the number of snapshots (fields) that will be used to compute initial state for the iterative solver, the first parameter corresponds to the Helmhotz solver for the velocity while second to the Poisson solver for the pressure (POD-based extrapolation is not implemented for the pressure), the default values for the parameters are zero.

5. After setting up (correctly) the four "*.rea" files required for our simulation, we set the so called configuration file. The configuration guides $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}$ on how to process the computation, that is how to split the global communicator and how to connect between patches. The configuration file in our case has the following context:

```
5  number of runs
-N16   IO1D/1Dart.in
-N1024 -i -n5 -chk  -V -S -vtk  50 IO_1/part_1.rea
-N128  -i -n6 -chk  -V -S -vtk 150 IO_2/part_2.rea
-N256  -i -n5 -chk  -V -S -vtk 250 IO_3/part_3.rea
-N256  -i -n8 -chk  -V -S -vtk 350 IO_4/part_4.rea
4  number of 1D/3D BCs
3 11  1 0 6  2 0 6  16 0 6  5 0 6  6 0 6  7 0 6  8 0 6  9 0 6  ...
      10 0 6 11 0 6  12 0 6  13 0 6  14 0 6  15 0 6
2 3   3 0 6  4 0 6   17 0 6  18 0 6  16 0 6
```

```
1 10  5 0 6  21 0 6  22 0 6  23 0 6  24 0 6  25 0 6  26 0 6  ...
      27 0 6 28 0 6  29 0 6  30 0 6
1 10  6 0 6  31 0 6  32 0 6  33 0 6  34 0 6  35 0 6  36 0 6  ...
      37 0 6  38 0 6  39 0 6  40 0 6
1  number of TG sites
0 1 2 3 4
```

The "number of runs" indicates how many tasks $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}}$ will execute (for each task L3 sub-communicator is created). In our case five tasks are executed, the first is 1D simulation with input file " 1Dart.in" located in directory named "IO1D". Although, 1D simulation is executed on a single core only, it is recommended to allocate a node to this task, in our example we provide 16 cores (-N16), assuming that the simulation is performed on Ranger, which has 16 cores per node. Next four tasks are solution of 3D Navier-Stokes equations, defined in four patches. For each patch the user may request different number of cores. It is highly recommended that no computational node is split between different tasks. As a rule of thumb the number of cores for each task should be proportional to the number of spectral elements in each patch, if the same order of polynomial expansion is employed in each patch. Incrementing polynomial order by one typically slowes the computation by a factor of ≈ 1.5 if the low-energy preconditioner is employed and by factor ≈ 2 in simulations with the diagonal preconditioner. Different preconditioners may be specified for each patch, as well as different orders of polynomial expansion (also keep $P \geq 4$). At each of the four lines corresponding to 3D patches we also specify patch-specific arguments (similar to simulations with $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r}$). Argument $-n5$ sets polynomial expansion order to $P = 4$, argument $-chk$ instructs to output field files with a frequency specified by $IOSTEP$ parameter in the "*.rea" file, Argument $-V$ is to indicate that we use time-dependent boundary conditions, argument $-S$ indicates that the field files created at each $IOSTEP$ will not overwrite the previous once, but instead will be saved in separate directories, for example for the first task the directories will be named "IO_1/PART_1/CHK_DATA_x", where x is an index. Arguments $-vtki$, instruct $\boldsymbol{\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r\mathcal{G}}$ to output at every i steps visualization data "*.vtu" file format - appropriate for visualization with ParaView, the data corresponds

to solution computed at vertices only. The last argument is the location of the "*.rea" file. For example for the first path the input file "part_4.rea" is located in directory "IO_1"; the output files corresponding to this patch will be saved in "IO_1". Next we specify the number of 3D patches and then information regarding the arterial network described by each patch. First and the second integers specify the number of inlets and outlets, correspondingly; then we provide *global* indices for arteries which have inlets, outlets and also for overlapping segments. For each inlet/outlet the data is specified by three integers, but currently only the first out of each triplet is an important number, the rest are to comply with older versions of $\mathcal{NEKTARG}$. IDs of inlets are provided first and followed by IDs for outlets. in our example the inlets of the first patch have IDs 1 2 and 16, and their corresponding *local* numerators are 1 2 and 3, the corresponding labels for these inlets are "bcA(x,y,z)", "bcB(x,y,z)" and "bcC(x,y,z)". The indices of outlets in the first patch are 5,6,7,...14,15, and the corresponding *local* numerators are 1,2,3,....,10,11, the corresponding labels for the outlets are "bcA(x,y,z)", "bcB(x,y,z)", ... "bcK(x,y,z)". The mapping for patches 2,3 and 4 is defined analogously. The arterial segments having the same *global* ID correspond to the same artery and will be merged by the inter-patch conditions. In our example the inlet No. 3 (global ID=16) of the first patch is connected with the outlet No. 3 of the second patch, which has also global ID=16. The last section of configuration file is to specify the mapping between the L3 and L1 communicators. in our example the first 16 ranks of MPI_COMM_WORLD will go to the forst task (1D model) the next 1024 ranks of MPI_COMM_WORLD will form L3 sub-communicator of assigned for the first 3D patch, and so on.

6. To run $\mathcal{NEKTARG}$ we calculate the total number of cores and then submit the job using $mpirun\ -n1552\ a.out\ conf.file$.

Bibliography

[1] http://ipm-hpc.sourceforge.net.

[2] project descriptions have been published on-line (http://www.physiome.org and http://www.europhysiome.org).

[3] http://glaros.dtc.umn.edu/gkhome/views/metis.

[4] http://www.netlib.org/scalapack.

[5] http://www.cs.berkeley.edu/ demmel/SuperLU.htm.

[6] www.pointwise.com.

[7] http://www.teragrid.org/userinfo/jobs/mpich_g2.php.

[8] Private communication with Kutay Ustuner, Scientist, Principal Siemens, Health Care Ultrasound Division.

[9] Disorder distal to modified stenoses in steady and pulsatile flow. *Journal of Biomechanics*, 11:441–453, 1978.

[10] An analysis of three different formulations of the discontinuous galerkin method for diffusion equations. *Mathematical Models and Methods in Applied Sciences*, 13(3):395–413, 2003.

[11] Selecting the numerical flux in discontinuous galerkin methods for diffusion problems. *Journal of Scientific Computing*, 22-23(3):385–411, 2005.

[12] Y. M. Akay, M. Akay, W. Welkowitz, S. Lewkowicz, and Y. Palti. Dynamics of the sounds caused by partially occluded femoral arteries in dogs. *Annals of Biomedical Engineering*, 22:493–500, 1994.

[13] J. Alastruey, K.H. Parker, J. Peiro, S.M. Byrd, and S.J. Sherwin. Modelling of circle of Willis to assess the effects of anatomical variations and inclusions on cerebral flows. *J. Biomechanics*, 40:1794–1805, 2007.

[14] A. D. August, S. A. G. McG. Thom B. Ariff, X.Y. Xu, and A.D. Hughes. Analysis of complex flow and relationship between blood pressure, wall shear stress, and intima-media thickness in the human carotid artery. *Am. J. Physiol. Heart. Circ. Physiol.*

[15] H. Baek, M.V. Jayaraman, and G.E. Karniadakis. Distribution of WSS on the internal carotid artery with an aneurysm: A CFD sensitivity study. In *Proceedings of IMECE2007*, November 11-15, Seattle, USA 2007.

[16] R. .A. Bajura and M.R. Catalano. Transition in a two-dimensional plane wall jet. *Journal of Fluid Mechanics*, 70:773–799, 1975.

[17] J. Bale-Glickman, K. Selby, D. Saloner, and O. Savas. Experimental flow studies in exact-replica phantoms of atherosclerotic carotid bifurcations under steady input conditions. *Journal of Biomechanical Engineering*, 125(1):38–48, 2003.

[18] B. Bergen, F. Hulsemann, and U. Rude. Is 1.7 1010 unknowns the largest finite element system that can be solved today? In *SC05*, 2005. No page numbers available.

[19] G. Berkooz, P. Holmes, and J.L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993.

[20] C. Bernardi and I. Maday. *Spectrales de problèmes aux limites elliptiques*. Springer, Paris, 1992.

[21] I. Bica. Iterative substructiring algorithm for p-version finite element method for elliptic problems, 1997. PhD thesis, Courant Institute, NYU.

[22] M. L. Bittencourt. Fully tensorial nodal and modal shape functions for triangles and tetrahedra. *International Journal for Numerical Methods in Engineering*, 63(11):1530–1558, 2005.

[23] M. G. Blyth and C. Pozrikidis. A lobatto interpolation grid over the triangle. *IMA Journal of Applied Mathematics*, 71(1):153–169, 2006.

[24] B. Boghosian, P. Coveney, S. Dong, L. Finn, S. Jha, G. Karniadakis, and N. Karonis. NEKTAR, SPICE, and Vortonics: Using federated grids for large scale scientific applications. pages 34–43, June 2006.

[25] I. Borazjani and F. Sotiropoulos L. Ge. Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3d rigid bodies. *Journal of Computational Physics*, 16:7587–7620, 2008.

[26] F. Cassot, F. Lauwers, C. Fuard, S. Prohaska, and V. Lauwers-Cances. A novel three-dimensional computer-assisted method for a quantitative study of microvascular networks of the human cerebral cortex. *Microcirculation*, 13:1–18, 2006.

[27] M. A. Castro, C.M Putman, and J. R. Cebral. Patient-specific computational fluid dynamics modeling of anterior communicating artery aneurysms: A study of the sensitivity of intra-aneurysmal flow patterns to flow conditions in the carotid arteries. *American Journal of Neuroradiolgy*, 27:2061–2068, 2006.

[28] M. A. Castro, C.M Putman, and J. R. Cebral. Patient-specific computational modeling of cerebral aneurysms with multiple avenues of flow from 3d rotational angiography images. *Academic Radiology*, 13(7):811–821, 2006.

[29] Q. Chen and I. Babuška. The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. *Computational Methods in Applied Mechanics and Engineering*, 137(1):89–94, 1996.

[30] D. Chungcharoen. Genesis of korotkoff sounds. *American Journal of Physiology*, 207:190–194, 1964.

[31] C. Clark. The propagation of turbulence produced by a stenosis. *Journal of Biomechanics*, 13:591–604, 1980.

[32] B. Cockburn, G. E. Karniadakis, and C. W. Shu. *Discontinuous Galerkin Methods: Theory, Compuration and Applications*. Springer, 2000.

[33] S. Dong, J. Insley, N.T. Karonis, M. Papka, J. Binns, and G.E.. Karniadakis. Simulating and visualizing the human arterial system on the TeraGrid. 22(8):1011–1017, 2006.

[34] S. Dong and G. E. Karniadakis. Dual-level parallelism for high-order cfd methods", journal = Parallel Computing, volume = 30, pages = 1-20, year = 2004.

[35] S. Dong, G.E. Karniadakis, and N.T. Karonis. Cross-site computations on the TeraGrid. *Computing in Science & Engineering*, 7:14–23, 2005.

[36] P.F. Fischer, F. Loth, S.E. Lee, S.W. Lee, D. Smith, and H. Bassiouny. Simulation of high reynolds number vascular flows. *CMAME*, 196:3049–3060, 2007.

[37] L. Formaggia, J. F. Gerbeau, F. Nobile, and A. Quarteroni. Numerical treatment of defective boundary conditions for the navier-stokes equations. *SIAM Journal on Numerical Analysis*, 40(1):376 – 401, 2002.

[38] L. Formaggia, J. F. Gerbeaum, F. Nobile, and A. Quarteroni. On the coupling of 3d and 1d navier-stokes equations for flow problems in compliant vessels. *Computer Methods in Applied Mechanics and Engineering*, 191:561–582.

[39] L. Formaggia, D. Lamponi, M. Tuveri, and A. Veneziani. Numerical modeling of 1d arterial networks coupled with a lumped parameters description of the heart. *Computer Methods in Biomechanics and Biomedical Engineering*, 9(5):273–288, 2006.

[40] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani. Multiscale modelling of the circulatory system: a preliminary analysis. *Computing and Visualization in Science*, 2(2-3):75–83, 2004.

[41] M. Gander, C. Japhet, Y. Maday, and 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):259–266, 2006.

[42] C.A. Gibbons and R.E. Shadwick. Circulatory mechanics in the toad bufo marinus: Ii. haemodynamics of the arterial windkessel. *Journal of Experimental Biology*, 158:291–306, 1991.

[43] L. Grinberg and G. E. Karniadakis. Outflow boundary conditions for arterial networks with multiple outlets. *Annals of Biomedical Engineering*, 36(9):1496–1514, 2008.

[44] L. Grinberg, B. Toonen, N. Karonis, and G.E. Karniadakis. A new domain decomposition technique for TeraGrid simulations. In *TG07*, June 2007.

[45] J. Hesthaven. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM Journal on Numerical Analysis*, 35(2):655–676, 1998.

[46] J. Heywood, R. Rannacher, , and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *Int. J. Num. Meth. Fluids*, 22(5):325–352, 1996.

[47] T. J. R. Hughes, G. Scovazzi, P. B. Bochev, and A. Buffa. A multiscale discontinuous galerkin method with the computational structure of a continuous galerkin method. *Computer Methods in Applied Mechanics and Engineering*, 195.

[48] G. E. Karniadakis, M. Israeli, and S.A. Orszag. High-order splitting methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 97(1):414–443, 1991.

[49] G. E. Karniadakis and S. J. Sherwin. A triangular spectral element method; applications to the incompressible navier-stokes equations. *Computational Methods in Applied Mechanics and Engineering*, 123:189–229, 1995.

[50] U. Köhler, I. Marshall, M. B. Robertson, Q. Long, and X.Y. Xu. Mri measurement of wall shear stress vectors in bifurcation models and comparison with cfd predictions. *Journal of Magnetic Resonance Imaging*, 14(5), pages = 563-73, year = 2001).

[51] R. M. Kirby and G.E. Karniadakis. De-aliasing on non-uniform grids: algorithms and applications. *Journal of Computational Physics*, 191:249–264, 2003.

[52] R.M Kirby and G.E. Karniadakis. *Spectral element and hp methods, Encyclopedia of Computational Mechanics*. John Wiley & Sons, 2004.

[53] D. N. Ku. Blood flow in arteries. *Annual Review of Fluid Mechanics*, 29:399–434, 1997.

[54] D. N. Ku, D. P. Giddens, C. K. Zarins, and S. Glagov. Pulsatile flow and atherosclerosis in the human carotid bifurcation. positive correlation between plaque location and low oscillating shear stress. *Arteriosclerosis*, 5:293–302, 1985.

[55] M. Lell, K. Anders, E. Klotz, H. Ditt, W. Bautz, and B. F. Tomandl. Clinical evaluation of bone-subtraction ct angiography (bscta) in head and neck imaging. *European Radiology*, 16(4):889–897, 2006.

[56] D. Loeckx, W. Coudyzer, F. Maes, D. Vandermeulen, G. Wilms, G. Marchal, and P. Suetens. Nonrigid registration for subtraction ct angiography applied to the carotids and cranial arteries. *Academic Radiology*, 14(12):1562–1576, 2007.

[57] J.W. Lottes and P.F. Fischer. Hybrid multigrid/Schwarz algorithms for the spectral element method. *J. Sci. Comput.*, 24(1):613–646, 2005.

[58] J. L. Lumley. The structure of inhomogeneous turbulent flow. *In atmospheric turbulence and radio wave propagation, (ed. A.M. Yaglom and V.I. Tatarski), Nauka, Moscow*, pages 160–178, 1967.

[59] J. Mandel. Two-level domain decomposition preconditioning for the p-version finite element method in three-dimensions. *Int. J. Numer. Meth. Eng.*, 29:1095 – 1108, 1990.

[60] M. Manhart. Vortex shedding from a hemisphere in turbulent boundary layer. *Theorectical and Computational Fluid Dynamics*, 12(1):1–28, 1998.

[61] C. J. Mills, I. T. Gabe, J. H. Gault, D. T. Mason, J. Ross Jr, E. Braunwald, and J. P. Shillingford. Pressure-flow relations and vascular impedance in man. *Cardiovascular Research*, 4:405417, 1970.

[62] J. A. Moore, D. A. Steinman, D. W. Holdsworth, and C. R. Ethier. Accuracy of computational hemodynamics in complex arterial geometries reconstructed from magnetic resonance imaging. *Annals of Biomedical Engineering*, 27(1):32–41, 2004.

[63] C.D. Murray. The physiological principle of minimum work. I The vascular system and the cost of blood volume. *Proc. Natl. Acad. Sci. USA*, 12:207–214, 1926.

[64] W.N. Nichols, M.F. ORourke, and C. Hartle. *McDonalds Blood Flow in Arteries; Theoretical, Experimental and Clinical Principles.* A Hodder Arnold Publication, 1998.

[65] M. S. Olufsen, C. S. Peskin, W. Y. Kim, E. M. Pedersen, A. Nadim, , and J. Larsen. Numerical simulation and experimental validation of blood flow in arteries with structured-tree outflow conditions. *Annals of Biomedical Engineering*, 28:1281–1299, 2000.

[66] M.S. Olufsen. Structured tree outflow condition for blood flow in larger systemic arteries. *American Journal of Physiology*, 276:H257–H268, 1999.

[67] Y. Papaharilaou, D.J. Doorly, S.J. Sherwin, J. Peiro, C. Griffith, N. Cheshire, V. Zervas, J. Anderson, B. Sanghera, N. Watkins, and C.G. Caro. Combined mr imaging and numerical simulation of flow in realistic arterial bypass graft models. *Biorheology*, 39:525–531, 2002.

[68] L. F. Pavarino and O. B. Widlund. A polylogarithmic bound for an iterative substructuring method for spectral elements in three dimensions. *SIAM Journal on Numerical Analysis*, 33(4):1303–1335, 1996.

[69] L. F. Pavarino, E. Zampieri, R. Pasquetti, and F. Rapetti. Overlapping schwartz method for fekete and gauss-lobatto spectral elements. *SIAM Jornal of Scientific Computing*, 29(3):1073–1092, 2007.

[70] J. Peiró, S. Giordana, C. Griffith, and S. J. Sherwin. High-order algorithms for vascular flow modelling. *International Journal for Numerical Methods in Fluids*, 40(1-2):137–151, 2002.

[71] R. Ponzini, C. Vergara, A. Redaelli, and A. Veneziani. Reliable cfd-based estimation of flow rate in haemodynamics measures. *Ultrasound in Medicine and Biology*, 32(10).

[72] Y. Saad. *Iterative methods for Sparse Linear Systems, second edition.* Society for Industrial and Applied Mathematics, 2003.

[73] K.E. Thorpe D.L. Sackett W. Taylor H.J.M. Barnett R. B. Haynes Sauve, J.S. and A. J. Fox. title =.

[74] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science (2nd ed).* Cambridge University Press, Cambridge, 1998.

[75] T.F. Sherman. The meaning of Murray's law. *J. Gen. Physiol.*, 78:431–453, 1981.

[76] S. Sherwin and M. Casarin. Low-energy basis preconditioning for elliptic substructured solvers based on unstructured spectral/hp element discretization. *J. Comp. Phys.*, 171(1):394–417, 2001.

[77] S. J. Sherwin and H.M. Blackburn. Three-dimensional instabilities of steady and pulsatile axisymmetric stenotic flows. *Journal of Fluid Mechanics*, 533(297-327).

[78] S. J. Sherwin and G. E. Karniadakis. A new triangular and tetrahedral basis for high-order finite element methods. *International Journal for Numerical Methods in Engineering*, 38:3775–3802.

[79] S. J. Sherwin and G. E. Karniadakis. *Spectral/hp element methods for CFD, second ed.* Oxford University Press, Oxford, 2005.

[80] S. J. Sherwin and J. Peiró. Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering.*

[81] S.J. Sherwin, V. Franke, J. Peiro, and K. Parker. 1D modelling of a vascular network in space-time variable. *J. Eng. Math.*, 47:217–250, 2003.

[82] L. Sirovich. Turbulence and dynamics of coherent structures: I-iii. *Quarterly of Applied Mathematics*, 45:561–590, 1987.

[83] M. Snir, S. W. Otto, D. W. Walker, J. Dongarra, and S. Huss-Lederman. *MPI: The Complete Reference.* Cambridge: MIT Press, 1995.

[84] R.L. Spilker, J.A. Feinstein, D.W. Parker, V.M. Reddy, and C.A. Taylor. Morphometry-based impedance boundary conditions for patient-specific modeling of blood flow in pulmonary arteries. *Ann. of Biomed. Eng.*, 35(4):546–559, 2007.

[85] B.N. Steele, M.S. Olufsen, and C.A. Taylor. Fractal network model for simulating abdominal and lower extremity blood flow during resting and exercise conditions. *Computer Methods in Applied Mechanics and Engineering*, 10:39–51, 2007.

[86] D.A. Steinman, T.L. Poepping, M. Tambasco, R.N. Rankin, and D.W. Holdsworth. Flow patterns at the stenosed carotid bifurcation: Effects of concentric versus eccentric stenosis. *Annals of Biomedical Engineering*, 28:415, 2000.

[87] N. Stergiopulos, D.F. Young, and T.R. Rogge. Computer simulation of arterial flow with applications to arterial and aortic stenoses. *J. Biomech*, 25:1477–1488, 1992.

[88] G. Strang and G.J. Fix. *An analysis of the finite element method*. Prentice-Hall, 1973.

[89] J. S. Stroud, S. A. Berger, and D. Saloner. Numerical analysis of flow through a severely stenotic carotid artery bifurcation. *Journal of Biomechanical Engineering*, 124:9–20, 2002.

[90] C. A. Taylor, T. J. R. Hughes, and C. K. Zarins. Effect of exercise on hemodynamic conditions in the abdominal aorta. *Journal of Vascular Surgery*, 29(6):10771089, 1999.

[91] M. A. Taylor, B. A. Wingate, and L. P. Bos. Cardinal function algorithm for computing multivariate quadrature points. *SIAM Journal on Numerical Analysis*, 45(1):193–205, 2007.

[92] L. N. Trefethen and M. Embree. *Spectra and pseudospectra: The behavior of Nonnormal Matrices and Operators*. 2005.

[93] H.M. Tufo and P.F. Fischer. Fast parallel direct solvers for coarse grid problems. *J. Parallel & Distr. Comput.*, 61(2):151–177, 2001.

[94] S. A. Urquiza, P. J. Blanco, M. J. Vénere, and R. A. Feijóo. Multidimensional modelling for the carotid artery blood flow. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4002–4017, 2006.

[95] S. S. Varghese, S H. Frankel, and P.F. Fischer. Direct numerical simulation of stenotic flows. part 2. pulsatile flow. *Journal of Fluid Mechanics*, 582:281–318, 2007.

[96] A. Veneziani and C. Vergara. Flow rate defective boundary conditions in haemodynamics simulations. *International Journal for Numerical Methods in Fluids*, 47.

[97] I.E. Vignon-Clementel, C.A. Figueroa, K.E. Jansen, and C.A. Taylor. Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries. *Computer Methods in Applied Mechanics and Engineering*, 195:3776–3796, 2006.

[98] P. Volino and N. Magnenat-Thalmann. The spherigon: A simple polygon patch for smoothing quickly your polygonal meshes. *Proceedings of the Computer Animation*, pages 72–78, 1998.

[99] S. Wandzura and H. Xiao. Symmetric quadrature rules on triangle. *Computers and Mathematics with Applications*, 45:1829–1840, 2003.

[100] J.J. Wang and K.H. Parker. Wave Propagation in a model of the arterial circulation. *J. Biomech.*, 37:457–470, 2004.

[101] K. C. Wang, C. A. Taylor, Z. Hsiau, D. Parker, and R. W. Dutton. Level set methods and mr image segmentation for geometric modelling in computational hemodynamics. *Engineering in Medicine and Biology Society, Proceedings of the 20th Annual International Conference of the IEEE 1998*, 6(29):3079 – 3082, 1998.

[102] S. Wang, W. Lee, J. Provost, J. Luo, and E. Konofagou. A composite high-frame-rate system for clinical cardiovascular imaging. *IEEE Trans Ultrason Ferroelectr Freq Control*, 55(10):2221–2233, 2008.

[103] B. Weir. Unruptured intracranial aneurysms: a review. *Journal of Neurosurgery*, 96:3–42, 2002.

[104] D.B. Xiu and S.J. Sherwin. Parametric Uncertainty Analysis of Pulse Wave Propagation in a Model of a Human Arterial Network. *J. of Comp. Phys.*, 226:1385–1407, 2007.

[105] M. Zamir. On fractal properties of arterial trees. *J. Theor. Biol.*, 197:517–526, 1999.