JOURNAL OF THEORETICAL
AND APPLIED MECHANICS
42, 4, pp. 869-903, Warsaw 2004

COMPUTATIONS OF AN UNSTEADY VISCOUS FLOW IN A THREE-DIMENSIONAL SYSTEM OF DUCTS. PART II: IMPLEMENTATION OF THE SPECTRAL ELEMENT METHOD AND SAMPLE RESULTS

Jacek Szumbarski
Piotr Olszewski
Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology
e-mail: jasz@meil.pw.edu.pl; polsz@meil.pw.edu.pl

Konrad Wawruch
Interdisciplinary Centre for Mathematical and Computational Modeling, Warsaw University
e-mail: kwaw@icm.edu.pl

Zbigniew Małota
Foundation of Cardiac Surgery Development, Zabrze
e0mail: zmalota@o2.pl

The second part the paper focuses on implementation issues of the spectral element technique and presents sample results of numerical simulations. Particular algorithms like preconditioning of the presser solver, fast diagonalization method and the projection method for solving sequences of large linear systems are described in some details. The computational results, obtained for both test domains and geometries of medical origin, are presented. The comparison with the results of computations with the FIDAP package is provided. The difficulties and limitations of the spectral method and current implementation are discussed. Attempts of the parallelization of the spectral code and the obtained efficiency are briefly described.

Key words: Navier-Stokes equations, spectral element methods, pressure preconditioners, fast diagonalization, parallel computations

1. Introduction

In the first part of the paper, the problem of a nonstationary flow of a viscous incompressible fluid in a three dimensional pipe system has been formulated. The most remarkable element of this formulation is, following the terminology introduced by Formaggia *et al.* (2000), the enforcement of "defective" boundary conditions at inlet/outlet sections of a computational domain. Such conditions prescribe temporal variation of the flow rate (the volume flux) or area-averaged static pressure. It means that distributions of velocity and pressure fields across the inlet/outlet section are not *a priori* assumed – they are obtained during a solution process. Moreover, the flow rate inlet/outlet conditions (referred to as VF-type conditions) can coexist with the area-averaged pressure conditions (referred to as AP-type conditions), meaning that each of the inlet/outlet sections can be of either of these types (but not of both at the same time). While the boundary conditions of the AP-type can be incorporated directly in the weak formulation, the VF-type conditions are, in fact, additional constrains imposed on the problem, and they are enforced using Lagrange's multipliers. If mixed explicit-implicit time integration schemes are used, the algebraic problem to be solved during each time step becomes linear. It is convenient to seek for a solution to such a system in the form of a linear combination of particular solutions, which all but one can be computed in advance. This procedure allows also for simple evaluation of time-dependent Lagrange's multipliers, which are interpreted as instantaneous values of the area-averaged pressure at VF-type inlets/outlets.

In this part of the paper, we discuss the most important elements of the numerical implementation based on the spectral elements and present sample results of numerical calculations. For a convenient reference, the summary of the numerical algorithm, formulated in Part I, is provided in Section 2. A brief account of the spectral element discretization used in this study is presented in Section 3. For a detailed exposition of the spectral methods, the reader may refer to recent monographs (Deville *et al.*, 2002; Karniadakis and Sherwin, 1999).

Some issues, particularly important for the overall performance of the method, also are addressed: the construction of the preconditioner of the pressure solver (Section 4), the fast diagonalization method (Section 5) and the projection method for solving a sequence of large linear problems (Section 6). Some longer mathematical derivations and formulas have been shifted to Appendix.

Sample numerical results are described in Section 7. The focus is on the validation of the algorithm in the case of mixed inlet/outlet conditions rather than on solving a particular (technically interested) problem. Therefore, most of the calculations are carried out for a simple T-shaped pipe system. Some preliminary results, obtained for a more complicated model of the Blallock-Taussig shunt, are also presented. Finally, we shortly discuss an important issue of the parallelization and present some preliminary results concerning the parallel efficiency of the spectral solver.

2. Description of the numerical method

In this section, the numerical method, described in Part I of the paper, is summarized for convenient reference.

Let $\boldsymbol{u}_\alpha^{(m)}$ $(\alpha = 1,2,3)$ and $\boldsymbol{\pi}^{(m)}$ denote vectors of a spectral representation of velocity components and pressure computed in the time step $t^{(m)}$. The following procedure, based on the OIFS scheme, yields the solution in the next time step $t^{(m+1)}$:

Computation of the convection substep

The following initial value problems $(k = 1, ..., K$, where K is the order of OIFS scheme)

$$\begin{aligned}&\frac{d}{dt}(\widehat{\boldsymbol{u}}_\alpha)_k = -C_\alpha[(\widehat{\boldsymbol{u}}_1)_k, (\widehat{\boldsymbol{u}}_2)_k, (\widehat{\boldsymbol{u}}_3)_k](\widehat{\boldsymbol{u}}_\alpha)_k \qquad \alpha = 1,2,3\\ &(\widehat{\boldsymbol{u}}_\alpha)_k(t = t^{(m-k+1)}) = \boldsymbol{u}_\alpha^{(m-k+1)}\end{aligned} \tag{2.1}$$

are integrated numerically using the 4th order Runge-Kutta explicit scheme. The integration is carried out with the uniform sub-step $h = \Delta t/M_S$ (M_S is typically not larger that 5) until the time $t^{(m+1)} = t^{(m)} + \Delta t$ is reached. The system of differential equations (2.1) is a space-discretized counterpart of equations (3.13) from Part I. As a result, one obtains vectors $(\widehat{\boldsymbol{u}}_\alpha)_k^{(m+1)} = (\widehat{\boldsymbol{u}}_\alpha)_k(t = t^{(m+1)})$, $\alpha = 1,2,3$, $k = 1, ..., K$.

Determination of the solution to the special Stokes problem

Next, the following (unsteady) Stokes problem is solved

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & (\mathbf{D}_1)^\top \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & (\mathbf{D}_2)^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & (\mathbf{D}_3)^\top \\ \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^{\{0\}} \\ \boldsymbol{u}_2^{\{0\}} \\ \boldsymbol{u}_3^{\{0\}} \\ \boldsymbol{\pi}^{\{0\}} \end{bmatrix}^{(m+1)} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \\ \mathbf{0} \end{bmatrix}^{(m+1)} \tag{2.2}$$

where the right-hand side vectors are defined as follows

$$\boldsymbol{r}_\alpha^{(m+1)} = -\frac{1}{\Delta t}\sum_{k=1}^{K} \beta_k \mathbf{M}_V (\widehat{\boldsymbol{u}}_\alpha)_k^{(m+1)} - (\boldsymbol{\Lambda}_\alpha^P)^\top \boldsymbol{P}^{(m+1)} \qquad \alpha = 1,2,3 \tag{2.3}$$

In the above, the symbol $\boldsymbol{P}^{(m+1)}$ denotes the vector of the section-averaged pressure prescribed for all inlets/outlets of the AP-type at the time $t = t^{(m+1)}$.

The solution to linear system (2.2) is sought using the Schur-compliment method in an incremental version, which can be summarized as follows:

- Solution of three (independent) linear systems

$$\mathbf{A}\widetilde{\boldsymbol{u}}_\alpha = \boldsymbol{r}_\alpha^{(m+1)} - (\mathbf{D}_\alpha)^\top \boldsymbol{\pi}^{\{0\}(m)} \qquad \alpha = 1,2,3 \tag{2.4}$$

  The Helmholtz matrix $\mathbf{A}$ is symmetric, positive definite and, typically, strongly diagonally dominant. Thus, the above systems can be efficiently solved with the Conjugate Gradients Method (CGM) with simple diagonal preconditioning.

- Solution of the linear system for the pressure correction

$$\mathbf{S}\boldsymbol{\pi}' = \mathbf{D}_1\widetilde{\boldsymbol{u}}_1 + \mathbf{D}_2\widetilde{\boldsymbol{u}}_2 + \mathbf{D}_3\widetilde{\boldsymbol{u}}_3 \tag{2.5}$$

  where

$$\mathbf{S} = \mathbf{D}_1\mathbf{A}^{-1}(\mathbf{D}_1)^\top + \mathbf{D}_2\mathbf{A}^{-1}(\mathbf{D}_2)^\top + \mathbf{D}_3\mathbf{A}^{-1}(\mathbf{D}_3)^\top \tag{2.6}$$

  The Uzawa matrix $\mathbf{S}$ is symmetric and positive definite. For an unsteady Stokes problem, this matrix is very poorly conditioned. Hence, an effective preconditioning technique is compulsory. We will discuss this problem in details later. Note also that computing product, like $\mathbf{S} \cdot \boldsymbol{p}$, involves solving of three linear systems with the Helmholtz matrix $\mathbf{A}$. In Part I, we shortly described the method of approximation of the matrix $\mathbf{A}^{-1}$ based on the Neumann expansion. This method, however, is not used in the current implementation of the algorithm.

  In effect of this part of calculations, we obtain the total pressure of the Stokes solution

$$\boldsymbol{\pi}^{\{0\}(m+1)} = \boldsymbol{\pi}^{\{0\}(m)} + \boldsymbol{\pi}' \tag{2.7}$$

- Calculation of the velocity components

  The new velocity field can be calculated as follows. First, the Helmholtz linear systems for velocity corrections

$$\mathbf{A}\boldsymbol{u}'_\alpha = -(\mathbf{D}_\alpha)^\top \boldsymbol{\pi}' \qquad \alpha = 1,2,3 \tag{2.8}$$

  are solved by PCGM iterations with the diagonal preconditioner. Afterwards, the final velocity components are obtained

$$\boldsymbol{u}_\alpha^{\{0\}(m+1)} = \boldsymbol{u}_\alpha^{\{0\}(m)} + \boldsymbol{u}'_\alpha \qquad \alpha = 1,2,3 \tag{2.9}$$

Finally, the vector of corresponding volume fluxes through all inlets/outlets of the VF type is calculated

$$\boldsymbol{f}_{VF}^{\{0\}(m+1)} = \sum_{\alpha=1}^{3} \boldsymbol{\Lambda}_{\alpha}^{F} \boldsymbol{u}_{\alpha}^{\{0\}(m+1)} \tag{2.10}$$

Determination of the full solution at the time $t = t(m+1)$

The full solution at $t = t^{(m+1)}$ is obtained in the form of the following linear combination

$$\begin{Bmatrix} \boldsymbol{u}_1^{(m+1)} \\ \boldsymbol{u}_2^{(m+1)} \\ \boldsymbol{u}_3^{(m+1)} \\ \boldsymbol{\pi}^{(m+1)} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{u}_1^{\{0\}(m+1)} \\ \boldsymbol{u}_2^{\{0\}(m+1)} \\ \boldsymbol{u}_3^{\{0\}(m+1)} \\ \boldsymbol{\pi}^{\{0\}(m+1)} \end{Bmatrix} + \sum_{k=1}^{N_{VF}} \lambda_k^{(m+1)} \begin{Bmatrix} \boldsymbol{u}_1^{\{k\}} \\ \boldsymbol{u}_2^{\{k\}} \\ \boldsymbol{u}_3^{\{k\}} \\ \boldsymbol{\pi}^{\{k\}} \end{Bmatrix} \tag{2.11}$$

Let us remind that the symbols $\boldsymbol{u}_{\alpha}^{\{k\}}$, $\alpha = 1, 2, 3$, $k = 1, ..., N_{VF}$, refer to the solutions to special Stokes problems

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & (\mathbf{D}_1)^{\top} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & (\mathbf{D}_2)^{\top} \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & (\mathbf{D}_3)^{\top} \\ \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^{\{k\}} \\ \boldsymbol{u}_2^{\{k\}} \\ \boldsymbol{u}_3^{\{k\}} \\ \boldsymbol{\pi}^{\{k\}} \end{bmatrix} = \begin{bmatrix} -(\boldsymbol{\Lambda}_1^{F})^{\top} \boldsymbol{\lambda}^{\{k\}} \\ -(\boldsymbol{\Lambda}_2^{F})^{\top} \boldsymbol{\lambda}^{\{k\}} \\ -(\boldsymbol{\Lambda}_3^{F})^{\top} \boldsymbol{\lambda}^{\{k\}} \\ \mathbf{0} \end{bmatrix} \tag{2.12}$$

where

$$\lambda_j^{\{k\}} = \begin{cases} 0 & \text{if} \ \ j \neq k \\ 1 & \text{if} \ \ j = k \end{cases}$$

For a fixed geometry and time step, these solutions can be calculated once and forever. The vector of the Lagrange multipliers $\boldsymbol{\lambda}^{(m+1)}$ is found from the linear system

$$\mathbf{T}_F \boldsymbol{\lambda}^{(m+1)} = \boldsymbol{F}^{(m+1)} - \boldsymbol{f}_{VF}^{\{0\}(m+1)} \tag{2.13}$$

The matrix $\mathbf{T}_F$ of system (2.13) is defined as follows

$$\mathbf{T}_F = \begin{bmatrix} f_1^{\{1\}} & \cdots & f_1^{\{N_{VF}\}} \\ \vdots & \vdots & \vdots \\ f_{N_{VF}}^{\{1\}} & \cdots & f_{N_{VF}}^{\{N_{VF}\}} \end{bmatrix} \tag{2.14}$$

where

$$f_j^{\{k\}} = \Big(\sum_{\alpha=1}^{3} \boldsymbol{\Lambda}_{\alpha}^{F} \boldsymbol{u}_{\alpha}^{\{k\}} \Big)_j \qquad\qquad j = 1, ..., N_{VF} \tag{2.15}$$

Note that parts (2) and (3) of the above procedure provide a solution to the following linear system

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & (\mathbf{D}_1)^\top & (\boldsymbol{\Lambda}_1^F)^\top \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & (\mathbf{D}_2)^\top & (\boldsymbol{\Lambda}_3^F)^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & (\mathbf{D}_3)^\top & (\boldsymbol{\Lambda}_3^F)^\top \\ \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Lambda}_1^F & \boldsymbol{\Lambda}_2^F & \boldsymbol{\Lambda}_3^F & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \boldsymbol{\pi} \\ \boldsymbol{\lambda} \end{bmatrix}^{(m+1)} = \begin{bmatrix} \boldsymbol{r}_1 - (\boldsymbol{\Lambda}_1^P)^\top \boldsymbol{P} \\ \boldsymbol{r}_2 - (\boldsymbol{\Lambda}_2^P)^\top \boldsymbol{P} \\ \boldsymbol{r}_3 - (\boldsymbol{\Lambda}_3^P)^\top \boldsymbol{P} \\ \mathbf{0} \\ \boldsymbol{F} \end{bmatrix}^{(m+1)} \tag{2.16}$$

Typically, the above system contains only a few equations more that the special Stokes systems described above. However, the matrix of (2.16) is not positive definite, and a different iterative technique from PCGM has to be applied.

3. Spectral element discretization

3.1. Physical and standard spectral elements

In the present work, a block-structured grid of hexahedral elements is used. In order to perform local operations, each element in the physical space is mapped to the standard spectral element: the cube $[-1,1]^3$ (see Fig. 1).

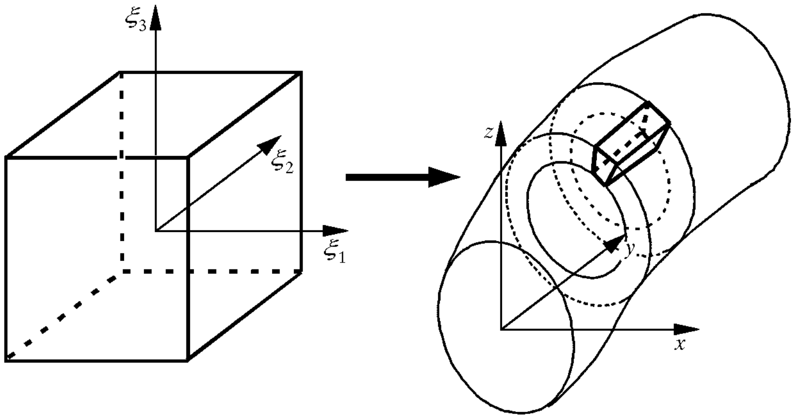

Fig. 1. Standard cube $[-1,1]^3$ and its transformation to a spectral element in the physical space

The mapping is based only on vertex information and takes the following triple-linear form

$$\begin{aligned}
\boldsymbol{x}(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(1-\xi_3)\boldsymbol{x}_1 + \frac{1}{8}(1+\xi_1)(1-\xi_2)(1-\xi_3)\boldsymbol{x}_2 + \\
&+ \frac{1}{8}(1+\xi_1)(1+\xi_2)(1-\xi_3)\boldsymbol{x}_3 + \frac{1}{8}(1-\xi_1)(1+\xi_2)(1-\xi_3)\boldsymbol{x}_4 + \\
&+ \frac{1}{8}(1-\xi_1)(1-\xi_2)(1+\xi_3)\boldsymbol{x}_5 + \frac{1}{8}(1+\xi_1)(1-\xi_2)(1+\xi_3)\boldsymbol{x}_6 + \\
&+ \frac{1}{8}(1+\xi_1)(1+\xi_2)(1+\xi_3)\boldsymbol{x}_7 + \frac{1}{8}(1-\xi_1)(1+\xi_2)(1+\xi_3)\boldsymbol{x}_8
\end{aligned} \tag{3.1}$$

More sophisticated mappings including detailed description of curvilinear interfaces of the element can also be used. An efficient algorithm for dealing with such mappings was proposed by Gordon and Hall (see, for instance, Chapter 4 of the work by Deville *et al.* (2002).

In order to carry out local differentiation and integration, the Jacobi matrix of transformation (3.1) should be calculated. The rows of this matrix can be expressed as follows

$$\begin{aligned}
\frac{\partial}{\partial \xi_1}\boldsymbol{x} &= \frac{1}{8}(1-\xi_2)(1-\xi_3)(\boldsymbol{x}_2-\boldsymbol{x}_1) + \frac{1}{8}(1+\xi_2)(1-\xi_3)(\boldsymbol{x}_3-\boldsymbol{x}_4) + \\
&+ \frac{1}{8}(1-\xi_2)(1+\xi_3)(\boldsymbol{x}_6-\boldsymbol{x}_5) + \frac{1}{8}(1+\xi_2)(1+\xi_3)(\boldsymbol{x}_7-\boldsymbol{x}_8) \\
\frac{\partial}{\partial \xi_2}\boldsymbol{x} &= \frac{1}{8}(1-\xi_1)(1-\xi_3)(\boldsymbol{x}_4-\boldsymbol{x}_1) + \frac{1}{8}(1+\xi_1)(1-\xi_3)(\boldsymbol{x}_3-\boldsymbol{x}_2) + \\
&+ \frac{1}{8}(1-\xi_1)(1+\xi_3)(\boldsymbol{x}_8-\boldsymbol{x}_5) + \frac{1}{8}(1+\xi_1)(1+\xi_3)(\boldsymbol{x}_7-\boldsymbol{x}_6) \\
\frac{\partial}{\partial \xi_3}\boldsymbol{x} &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(\boldsymbol{x}_5-\boldsymbol{x}_1) + \frac{1}{8}(1+\xi_1)(1-\xi_2)(\boldsymbol{x}_6-\boldsymbol{x}_2) + \\
&+ \frac{1}{8}(1-\xi_1)(1+\xi_2)(\boldsymbol{x}_7-\boldsymbol{x}_3) + \frac{1}{8}(1+\xi_1)(1+\xi_2)(\boldsymbol{x}_8-\boldsymbol{x}_4)
\end{aligned} \tag{3.2}$$

3.2. Local basic functions

The local representation of any scalar field defined in the element is based on the usage of the Lagrange interpolation. The interpolation grids in the standard cube $[-1,1]^3$ are defined as triple tensor products of one-dimensional sets of nodes in the interval $[-1,1]$. The grid for velocity components is generated over the Gauss-Jacobi-Lobatto (GJL) nodes

$$\begin{aligned} &\xi_V^{(0)} = -1 \qquad\qquad \xi_V^{(M_V-1)} = 1 \\ &\{\xi_V^{(j)}, \quad j = 1,...,M_V-2\} - \text{roots of the Jacobi polynomial } P^{1,1}_{M_V-2} \end{aligned} \tag{3.3}$$

The pressure field is interpolated using a grid generated using the Gauss-Legendre (GL) nodes

$$\{\xi_P^{(j)}, \quad j = 0,...,M_P-1\} - \text{roots of the Legendre polynomial } P_{M_P-1} \tag{3.4}$$

Note that the pressure grid contains no interface points.

The reason for using two separate (and staggered) grids for the velocity and pressure is to satisfy the div-stability (or inf-sup) condition (see Chapter 8 in Karnidakis and Sherwin (1999) and further references therein). We also assume that $M_V = M_P + 2$. Thus, the numbers of internal nodes of both grids are equal.

The components of the velocity field in the spectral element are represented by local interpolants, which are linear combinations of the following basic functions

$$b^V_{ijk}(\xi_1,\xi_2,\xi_3) = L^V_i(\xi_1)L^V_j(\xi_2)L^V_k(\xi_3) \qquad\qquad i,j,k = 0,...,M_V-1 \tag{3.5}$$

The Lagrange polynomials $L^V_i(\xi)$ $(i = 0,...,M_V-1)$ satisfy the conditions $L^V_i(\xi_V^{(j)}) = \delta^j_i$ $(j = 0,...,M_V-1)$. Analogously, the local basis for the pressure is given as

$$b^P_{ijk}(\xi_1,\xi_2,\xi_3) = L^P_i(\xi_1)L^P_j(\xi_2)L^P_k(\xi_3) \qquad\qquad i,j,k = 0,...,M_P-1 \tag{3.6}$$

where the Lagrange polynomials $L^P_i(\xi)$ satisfy the conditions $L^P_i(\xi_P^{(j)}) = \delta^j_i$ $(j = 0,...,M_P-1)$.

3.3. Local differentiation and integration

Consider a function $f(\xi)$ defined in the interval $[-1,1]$ and its interpolant $\widehat{f}(\xi)$ on the GJL nodes, i.e. the following polynomial

$$\widehat{f}(\xi) = \sum_{\alpha=0}^{M_V-1} f(\xi^V_\alpha)L^V_\alpha(\xi) \tag{3.7}$$

The collocation approximation of the derivative $f'(\xi)$ (called also pseudo-spectral derivative) is defined as follows: it is the GJL interpolant of the derivative of the interpolating polynomial (3.7). Thus, we have

$$f'(\xi) \approx \sum_{\alpha=0}^{M_V-1} \phi_\alpha L_\alpha^V(\xi) \Rightarrow \phi_\alpha \equiv \widehat{f}'(\xi_\alpha^V) = \sum_{\beta=0}^{M_V-1} L_\beta^{\prime V}(\xi_\alpha^V) f(\xi_\alpha^V) \tag{3.8}$$

This way, the vector of nodal values of the derivative $\boldsymbol{\phi}$ is directly related to the vector $\boldsymbol{f}$ containing the nodal values of the differentiated function f, namely

$$\boldsymbol{\phi} = \underline{\mathbf{d}}\boldsymbol{f} \tag{3.9}$$

where the symbol $\underline{\mathbf{d}}$ denotes the matrix of pseudo-spectral differentiation on the GJL grid

$$(\underline{\mathbf{d}})_{\alpha\beta} = L_\beta^{\prime V}(\xi_\alpha^V) \qquad\qquad \alpha, \beta = 0, M_V - 1 \tag{3.10}$$

Another one-dimensional operator used in the spectral element method is the interpolation from GJL grid to GL grid. Given a function $f(\xi)$ by its GJL interpolant $\widehat{f}(\xi)$, this operator yields the GL interpolant defined as

$$\widetilde{f}(\xi) = \sum_{\alpha=1}^{M_P-1} \psi_\alpha L_\alpha^P(\xi) \tag{3.11}$$

where

$$\psi_\alpha \equiv \widehat{f}(\xi_\alpha^P) = \sum_{\beta=0}^{M_V-1} L_\beta^V(\xi_\alpha^P) f(\xi_\beta^V) \tag{3.12}$$

In terms of the vectors of nodal values, the above interpolation can be written in the form of

$$\boldsymbol{\psi} = \underline{\mathbf{t}}\boldsymbol{f} \tag{3.13}$$

where $\underline{\mathbf{t}}$ is arectangular matrix defined as follows

$$(\underline{\mathbf{t}})_{\alpha\beta} = b_\beta^V(\xi_\alpha^P) \qquad\qquad \begin{array}{l} \alpha = 0, ..., M_P - 1 \\ \beta = 0, ..., M_V - 1 \end{array} \tag{3.14}$$

In the calculations involving the gradient and divergence operators, the components of the velocity field are differentiated and then interpolated to the GL grid. Therefore, it is convenient to introduce a composite operator. Its matrix representation is obtained as the product of the matrices defined by (3.10) and (3.14)

$$\underline{\mathbf{z}} = \underline{\mathbf{t}}\,\underline{\mathbf{d}} \qquad\qquad (\underline{\mathbf{z}})_{\alpha\beta} = \sum_{\gamma=0}^{M_V-1} (\underline{\mathbf{t}})_{\alpha\gamma}(\underline{\mathbf{d}})_{\gamma\beta} \tag{3.15}$$

All volume integrals are computed element-by-element using local transformations (3.1) to the standard cube $[-1,1]^3$. The integrals in the cube are computed using Gauss-Jacobi-Lobatto or Gauss-Legendre integration formulas (see Appendix B in Karniadakis and Sherwin, 1999).

4. Preconditioning of the pressure solver

During each time step, algebraic Stokes problem (2.2) has to be solved. The solution is calculated in three major steps explained in Section 2. The most difficult element of the described procedure is the evaluation of the pressure correction. It consists in solving of linear system (2.5) with Uzawa matrix (2.6). The solution is computed approximately using the iterative method of preconditioned conjugate gradients (PCGM). Since system (2.5) is ill conditioned (Maday *et al.*, 1993), the problem of selection of an appropriate preconditioner becomes crucial.

According to Maday *et al.* (1993), Cahouet and Chabard (1986), convergence of the CGM for linear system (2.5) can be efficiently accelerated by means of the following preconditioner

$$\mathbf{R}^{-1} = \nu \mathbf{M}_P^{-1} + \frac{\beta_0}{\Delta t}\mathbf{E}^{-1} \tag{4.1}$$

where $\mathbf{E}$ denotes the pseudo-Laplace matrix

$$\mathbf{E} = \mathbf{D}_1\mathbf{M}_V^{-1}\mathbf{D}_1^{\top} + \mathbf{D}_2\mathbf{M}_V^{-1}\mathbf{D}_2^{\top} + \mathbf{D}_3\mathbf{M}_V^{-1}\mathbf{D}_3^{\top} \tag{4.2}$$

and $\mathbf{M}_P$ is the pressure mass matrix. It should be noted that, in the nodal variant of the spectral element method, all mass matrices are diagonal and thus trivial to invert. The reason for this convenient structure stems from the fact that the volume integrals of products of the basic functions are computed with the Gauss or Gauss-Lobatto formulas based on the same local meshes that are used for the pressure or velocity interpolation. Thus, the basic functions form (discretely) orthogonal systems.

The major difficulty, however, presents efficient calculation of the matrix-vector product with the matrix $\mathbf{E}^{-1}$. In each iteration of the PCGM, one has to solve a linear system with the matrix $\mathbf{E}$, which can be formally written as

$$\mathbf{E}\boldsymbol{q} = \boldsymbol{r} \tag{4.3}$$

The solution should be determined with reasonable accuracy, perhaps not very high, but sufficient for the preconditioner to maintain a satisfactory convergence rate. The matrix $\mathbf{E}$ is symmetric and positive definite, thus the method of conjugate gradients is a natural choice. The point is that the matrix $\mathbf{E}$ is itself ill-conditioned and thus the convergence of the internal iterations will be very slow, unless another lower-level preconditioner is used.

The complete preconditioning strategy used in this study is based on the approach proposed by Ronquist (1991) and further developed by Couzy and Deville (1994). In the latter work, several variants of preconditioning of the Uzawa matrix were discussed. Numerical tests showed that the two-stage preconditioners, referred to as P5 and P7, are the most effective. In this study, the implementation based on the variant P5, described below, proved to be superior.

Assume we need to solve linear system (4.3) where $\dim \mathbf{E} = N$. Assume next that a K-dimensional subspace of R^N is defined. In the context of the Stokes problem, K is equal to the number of the spectral elements in the flow domain, and the subspace will correspond to the pressure which is constant inside each individual element, i.e. piecewise constant in the computational domain.

Assume that K orthonormal vectors have been chosen in the subspace. Using these vectors as columns, one can construct a matrix $\mathbf{J}$ with N rows and K columns. Clearly, the matrix $\mathbf{J}$ is orthogonal, i.e. $\mathbf{J}^\top\mathbf{J} = \mathbf{I}$. Now, the K-dimensional subspace can be formally defined formally as

$$\Pi = \{\boldsymbol{\pi} \in R^N : \quad \boldsymbol{\pi} = \mathbf{J}\boldsymbol{u}, \quad \boldsymbol{u} \in R^K\} \tag{4.4}$$

We will seek a solution to (4.3) in the following form

$$\boldsymbol{q} = \mathbf{J}\boldsymbol{q}_0 + \boldsymbol{q}_1 \tag{4.5}$$

where $\boldsymbol{q}_0$ is a vector from R^K. Thus, the vector $\boldsymbol{q}$ is expressed as a sum of two parts. The first part corresponds to the "background" distribution (uniform in each element), while the second one describes local variations with respect to the locally uniform "background".

The above representation, (4.5), is clearly not unique. We will find a particular pair $\{\boldsymbol{q}_0, \boldsymbol{q}_1\}$ in the following way. First, we define the operator

$$\mathbf{P}_{\Pi^\perp} = \mathbf{I} - \mathbf{E}\mathbf{J}\mathbf{E}_0^{-1}\mathbf{J}^\top \tag{4.6}$$

where

$$\mathbf{E}_0 = \mathbf{J}^\top\mathbf{E}\mathbf{J} \tag{4.7}$$

Note that $\mathbf{P}_{\Pi^\perp}$ is a projection operator, as the following calculation proves

$$\begin{aligned}\mathbf{P}_{\Pi^\perp}\mathbf{P}_{\Pi^\perp} &= (\mathbf{I}-\mathbf{EJE}_0^{-1}\mathbf{J}^\top)^2 = \\ &= \mathbf{I}-2\mathbf{EJE}_0^{-1}\mathbf{J}^\top + \mathbf{EJE}_0^{-1}\mathbf{J}^\top\mathbf{EJE}_0^{-1}\mathbf{J}^\top = \mathbf{I}-\mathbf{EJE}_0^{-1}\mathbf{J} = \mathbf{P}_{\Pi^\perp}\end{aligned} \tag{4.8}$$

The operator $\mathbf{P}_{\Pi^\perp}$ projects all vectors onto a subspace orthogonal to Π, denoted here as $\Pi^\perp$. Indeed, it is easy to verify that $\mathbf{J}^\top\mathbf{P}_{\Pi^\perp}\boldsymbol{w} = \mathbf{0}$. On the other hand, this projection is not orthogonal. The following equality, valid for each vector $\boldsymbol{\pi} \in \Pi$

$$\mathbf{P}_{\Pi^\perp}\mathbf{E}\boldsymbol{\pi} \equiv \mathbf{P}_{\Pi^\perp}\mathbf{EJ}\boldsymbol{u} = \mathbf{0} \tag{4.9}$$

shows that the operator $\mathbf{P}_{\Pi^\perp}$ is the projection onto $\Pi^\perp$ along the subspace $\mathbf{E}(\Pi)$.

Since $\mathbf{P}_{\Pi^\perp}\mathbf{EJ}\boldsymbol{q}_0 = \mathbf{0}$, the application of the projection operator to the equation

$$\mathbf{EJ}\boldsymbol{q}_0 + \mathbf{E}\boldsymbol{q}_1 = \boldsymbol{r} \tag{4.10}$$

yields

$$\mathbf{P}_{\Pi^\perp}\mathbf{E}\boldsymbol{q}_1 = \mathbf{P}_{\Pi^\perp}\boldsymbol{r}$$

or, equivalently

$$(\mathbf{I}-\mathbf{EJE}_0^{-1}\mathbf{J}^\top)\mathbf{E}\boldsymbol{q}_1 = (\mathbf{I}-\mathbf{EJE}_0^{-1}\mathbf{J}^\top)\boldsymbol{r} \tag{4.11}$$

The matrix of the linear system $\mathbf{E}_1 = (\mathbf{I}-\mathbf{EJE}_0^{-1}\mathbf{J}^\top)\mathbf{E}$ is singular. Nevertheless, the system is uniquely solvable in the subspace $\Pi^\perp$. The solution can be computed approximately using the Conjugate Gradient Method with an appropriate preconditioner described later. Once the vector $\boldsymbol{q}_1 \in \Pi^\perp$ is found, the K-dimensional vector $\boldsymbol{q}_0$ is evaluated as a solution to the following linear system

$$\mathbf{E}_0\boldsymbol{q}_0 = \mathbf{J}^\top(\boldsymbol{r} - \mathbf{E}\boldsymbol{q}_1) \tag{4.12}$$

In the context of the spectral element method, linear system (4.12) has as many unknowns as the number of elements in the computational domain. In most situations, the number of the elements does not exceed several thousands, and system (4.12) can be solved by an exact solver. Since the matrix $\mathbf{E}_0$ is symmetric and positive definite, the natural method is the Choleski Decomposition. For a fixed grid geometry, this decomposition can be evaluated once and forever and the forward/backward substitution part of the solution can be implemented efficiently using one of the sparse storage schemes (see Saad, 2003).

It should be noted that the solution to system (4.11) in the subspace $\Pi^\perp$ corresponds to the pressure distribution orthogonal to all piecewise-constant "background" distributions (generated by vectors from Π). The main advantage of the orthogonal decomposition is that linear system (4.11) can be efficiently preconditioned by the block matrix $\{\mathbf{E}\}$.

5. Block-diagonal preconditioning of inner iterations using the Fast Diagonalization Method

In this Section, we will briefly describe the preconditioning of conjugate gradients for linear system (4.11), based on the usage of the block matrix $\{\mathbf{E}\}$ (Couzy and Deville, 1994).

Consider a preconditioning step in PCGM iterations (see Appendix A). In order to find the vector $\widetilde{\boldsymbol{r}}$ out of the current residuum $\boldsymbol{r}$, we follow the following procedure:

(1) The global residual vector $\boldsymbol{r}$ is disassembled into local vectors $\boldsymbol{r}_k^{loc}$, $k = 1, ..., K$ corresponding to individual spectral elements.

(2) Local linear systems $\mathbf{E}_k^{loc}\widetilde{\boldsymbol{r}}_k^{loc} = \boldsymbol{r}_k^{loc}$, $k = 1, ..., K$ are solved independently for each spectral element. The matrices of these systems

$$\mathbf{E}_k^{loc} = \sum_{j=1}^{3}(\mathbf{D}_j)_k^{loc}(\mathbf{M}_V^{-1})_k^{loc}(\mathbf{D}_j^\top)_k^{loc} \tag{5.1}$$

are constructed using local velocity and pressure basic functions. The dimension of the local systems is equal to M_P^3. The dimension of the velocity local base depends on the conditions imposed at interfaces of the element. For internal elements, all components of the matrix $\mathbf{E}_k^{loc}$ are constructed using the full velocity base of M_V^3 functions. If some collocation nodes are located on the material surface of the computational domain, the local velocity base is accordingly reduced.

(3) The resulting local vectors $\widetilde{\boldsymbol{r}}_k^{loc}$, $k = 1, ..., K$ are assembled into the global vector $\widetilde{\boldsymbol{r}}$.

The local systems can be solved in many ways. Note that their dimension is small (typically they contain from several tens to several hundreds equations), so it is natural to use exact solvers. One can simply construct all local matrices explicitly and compute local Choleski factorizations. The numerical cost of such a preconditioner will be roughly as $K \cdot M_P^6$. With a greater number of spectral elements with different shapes, the memory required to store all decomposed local matrices becomes large. Couzy and Deville (1994) proposed an alternative approach based on the Fast Diagonalization Method (FDM), see Appendix B.

In order to use the FDM one has to construct the local matrix as a sum of appropriate tensor products of matrices corresponding to one-dimensional operations. Consider the standard element $[-1,1]^3$. Note that we can use index notations within each element: a "natural" triple-index notation and a single-index numeration. The latter is usually more convenient because it leads to a simpler formalism. The local velocity and pressure basic functions in the standard element, introduced in Section 2, can be conveniently numerated as follows

$$\begin{aligned} B_i^V(\xi_1,\xi_2,\xi_3) &= b^V_{I_1(i)}(\xi_1) b^V_{I_2(i)}(\xi_2) b^V_{I_3(i)}(\xi_3) \\ B_j^P(\xi_1,\xi_2,\xi_3) &= b^P_{J_1(j)}(\xi_1) b^P_{J_2(j)}(\xi_2) b^P_{J_3(j)}(\xi_3) \end{aligned} \tag{5.2}$$

In the above, the integer functions $I_\alpha(i)$ and $J_\alpha(j)$ $(\alpha = 1,2,3)$ define the mappings between triple-index and single-index numeration of the basic functions for the velocity and pressure, respectively.

We will now consider systematic calculations of the block matrix $\{\mathbf{E}\}$. It is sufficient to consider an arbitrary element in the physical space. However, in the case of general hexahedral geometry, the volume integrals will not factorize into three one-dimensional ones, and direct application of the FDM is not possible. Thus, let us assume that all physical elements are parallelepipeds.

In order to compute matrix (5.1), the following local matrices have to be computed

$$(\mathbf{D}_\alpha^\top)_{ij} = -\int\limits_{\Omega_K} \frac{\partial v_i}{\partial X_\alpha} q_j \, d\boldsymbol{x} = -\int\limits_{[-1,1]^3} \Big(\sum_{\beta=1}^{3} \frac{\partial B_i^V}{\partial \xi_\beta}\frac{\partial \xi_\beta}{\partial X_\alpha}\Big) B_j^P \Big|\frac{\partial(\boldsymbol{x})}{\partial(\boldsymbol{\xi})}\Big| \, d\boldsymbol{\xi} \tag{5.3}$$

The elements of the Jacobi matrix and its inverse in a parallelepiped are the same at each point of the integration domain. Introducing edge vectors

$$\begin{aligned} \boldsymbol{x}_2 - \boldsymbol{x}_1 = \boldsymbol{x}_3 - \boldsymbol{x}_4 = \boldsymbol{x}_6 - \boldsymbol{x}_5 = \boldsymbol{x}_7 - \boldsymbol{x}_8 \equiv \boldsymbol{k}_1 \\ \boldsymbol{x}_4 - \boldsymbol{x}_1 = \boldsymbol{x}_3 - \boldsymbol{x}_2 = \boldsymbol{x}_8 - \boldsymbol{x}_5 = \boldsymbol{x}_7 - \boldsymbol{x}_6 \equiv \boldsymbol{k}_2 \\ \boldsymbol{x}_5 - \boldsymbol{x}_1 = \boldsymbol{x}_6 - \boldsymbol{x}_2 = \boldsymbol{x}_7 - \boldsymbol{x}_3 = \boldsymbol{x}_8 - \boldsymbol{x}_4 \equiv \boldsymbol{k}_3 \end{aligned} \tag{5.4}$$

it is easy to show that

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}} = \frac{1}{2}[\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3] \qquad \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| = \frac{1}{8}|\boldsymbol{k}_1 \cdot (\boldsymbol{k}_2 \times \boldsymbol{k}_3)| \equiv \frac{1}{8}\mathrm{vol}\,\Omega \tag{5.5}$$

and

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} = \frac{2}{\mathrm{vol}\Omega} \begin{Bmatrix} (\boldsymbol{k}_2 \times \boldsymbol{k}_3)^\top \\ (\boldsymbol{k}_3 \times \boldsymbol{k}_1)^\top \\ (\boldsymbol{k}_1 \times \boldsymbol{k}_2)^\top \end{Bmatrix} \tag{5.6}$$

Now, the volume integral can be expressed in terms of one-dimensional integrals in $[-1,1]$

$$\begin{aligned} &\int\limits_\Omega \frac{\partial v_i}{\partial x_\alpha} q_j \; d\boldsymbol{x} = \\ &= \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| \Big[\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\Big]_{1,\alpha} \int\limits_{-1}^{1} b'^V_{I_1(i)} b^P_{J_1(j)} \; d\xi_1 \int\limits_{-1}^{1} b^V_{I_2(i)} b^P_{J_2(j)} \; d\xi_2 \int\limits_{-1}^{1} b^V_{I_3(i)} b^P_{J_3(j)} \; d\xi_3 + \\ &+ \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| \Big[\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\Big]_{2,\alpha} \int\limits_{-1}^{1} b^V_{I_1(i)} b^P_{J_1(j)} \; d\xi_1 \int\limits_{-1}^{1} b'^V_{I_2(i)} b^P_{J_2(j)} \; d\xi_2 \int\limits_{-1}^{1} b^V_{I_3(i)} b^P_{J_3(j)} \; d\xi_3 + \\ &+ \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| \Big[\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\Big]_{3,\alpha} \int\limits_{-1}^{1} b^V_{I_1(i)} b^P_{J_1(j)} \; d\xi_1 \int\limits_{-1}^{1} b^V_{I_2(i)} b^P_{J_2(j)} \; d\xi_2 \int\limits_{-1}^{1} b'^V_{I_3(i)} b^P_{J_3(j)} \; d\xi_3 \end{aligned} \tag{5.7}$$

All integrals in (5.7) are calculated approximately using the Gauss quadrature based on the pressure nodes. To this end, the velocity basic functions and their derivatives are interpolated from the Gauss-Lobatto nodes to the Gauss nodes. Using one-dimensional operations defined in Section 2, one can write formulas

$$\begin{aligned} &\int\limits_{-1}^{1} b'^V_r b^P_s \; d\xi \overset{GL}{\approx} w^P_s b'^V_r(\xi^P_s) = w^P_s(\boldsymbol{z})_{s,r} \\ &\int\limits_{-1}^{1} b^V_r b^P_s \; d\xi \overset{GL}{\approx} w^P_s b^V_r(\xi^P_s) = w^P_s(\boldsymbol{l})_{s,r} \end{aligned} \tag{5.8}$$

In order to use the FDM solver, one has to switch back to the triple-index tensor notation. The final formulas are

$$\begin{aligned} \widetilde{\mathbf{D}}_\alpha &= -(\boldsymbol{w}^P \otimes \boldsymbol{w}^P \otimes \boldsymbol{w}^P)(C_{1,\alpha}\boldsymbol{z} \otimes \boldsymbol{l} \otimes \boldsymbol{l} + C_{2,\alpha}\boldsymbol{l} \otimes \boldsymbol{z} \otimes \boldsymbol{l} + C_{3,\alpha}\boldsymbol{l} \otimes \boldsymbol{l} \otimes \boldsymbol{z}) \\ \widetilde{\mathbf{D}}^\top_\alpha &= -(C_{1,\alpha}\boldsymbol{z}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{l}^\top + C_{2,\alpha}\boldsymbol{l}^\top \otimes \boldsymbol{z}^\top \otimes \boldsymbol{l}^\top + C_{3,\alpha}\boldsymbol{l}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{z}^\top) \cdot \\ &\quad \cdot(\boldsymbol{w}^P \otimes \boldsymbol{w}^P \otimes \boldsymbol{w}^P) \end{aligned} \tag{5.9}$$

where the matrix C is defined as

$$\mathsf{C} = \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} \tag{5.10}$$

The local velocity mass matrix and its inverse can be written as follows

$$\begin{aligned}\widetilde{\mathsf{M}}_V &= \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| (\boldsymbol{w}^V \otimes \boldsymbol{w}^V \otimes \boldsymbol{w}^V)\\ \widetilde{\mathsf{M}}_V^{-1} &= \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big|^{-1} (\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)\end{aligned} \tag{5.11}$$

In the above, the structure $\boldsymbol{w}^V$ can be viewed as a diagonal matrix containing the weights of the Gauss-Lobatto quadrature, and $\widehat{\boldsymbol{w}}^V \equiv (\boldsymbol{w}^V)^{-1}$. The final form of the local $\widetilde{\mathsf{E}}$ matrix reads

$$\begin{aligned}&\widetilde{\mathsf{E}} \equiv \sum_{\alpha=1}^{3} \widetilde{\mathsf{D}}_\alpha \widetilde{\mathsf{M}}_V^{-1} \widetilde{\mathsf{D}}_\alpha^\top = \Big|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\Big| (\boldsymbol{w}^P \otimes \boldsymbol{w}^P \otimes \boldsymbol{w}^P) \cdot\\
&\cdot\Big\{(\mathsf{C}\mathsf{C}^\top)_{1,1} (\boldsymbol{z} \otimes \boldsymbol{l} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{z}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{l}^\top) +\\
&+(\mathsf{C}\mathsf{C}^\top)_{2,2} (\boldsymbol{l} \otimes \boldsymbol{z} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{z}^\top \otimes \boldsymbol{l}^\top) +\\
&+(\mathsf{C}\mathsf{C}^\top)_{3,3} (\boldsymbol{l} \otimes \boldsymbol{l} \otimes \boldsymbol{z})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{z}^\top) +\\
&+(\mathsf{C}\mathsf{C}^\top)_{1,2} \big[(\boldsymbol{z} \otimes \boldsymbol{l} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{z}^\top \otimes \boldsymbol{l}^\top) +\\
&+(\boldsymbol{l} \otimes \boldsymbol{z} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{z}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{l}^\top)\big] +\\
&+(\mathsf{C}\mathsf{C}^\top)_{1,3} \big[(\boldsymbol{z} \otimes \boldsymbol{l} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{z}^\top) +\\
&+(\boldsymbol{l} \otimes \boldsymbol{l} \otimes \boldsymbol{z})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{z}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{l}^\top)\big] +\\
&+(\mathsf{C}\mathsf{C}^\top)_{2,3} \big[(\boldsymbol{l} \otimes \boldsymbol{z} \otimes \boldsymbol{l})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{l}^\top \otimes \boldsymbol{z}^\top) +\\
&+(\boldsymbol{l} \otimes \boldsymbol{l} \otimes \boldsymbol{z})(\widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V \otimes \widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top \otimes \boldsymbol{z}^\top \otimes \boldsymbol{l}^\top)\big]\Big\} (\boldsymbol{w}^P \otimes \boldsymbol{w}^P \otimes \boldsymbol{w}^P)\end{aligned} \tag{5.12}$$

Unfortunately, formula (5.12) still cannot be cast into a form amenable to the FDM solver, and an additional restriction on the shape of an element is necessary. To this end, we will assume that the element is rectangular. Then, the inverse Jacobi matrix is expressed as follows

$$\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{x}} = \frac{2}{k_1k_2k_3}\left\{\begin{matrix} \dfrac{k_2k_3}{k_1}\boldsymbol{k}_1^\top \\ \dfrac{k_1k_3}{k_2}\boldsymbol{k}_2^\top \\ \dfrac{k_1k_2}{k_3}\boldsymbol{k}_3^\top \end{matrix}\right\} = 2\left\{\begin{matrix} \dfrac{1}{k_1^2}\boldsymbol{k}_1^\top \\ \dfrac{1}{k_2^2}\boldsymbol{k}_2^\top \\ \dfrac{1}{k_3^2}\boldsymbol{k}_3^\top \end{matrix}\right\}$$

$$\Big|\frac{\partial\boldsymbol{x}}{\partial\boldsymbol{\xi}}\Big|^{-1}\boldsymbol{\mathsf{C}}\boldsymbol{\mathsf{C}}^\top = \Big|\frac{\partial\boldsymbol{x}}{\partial\boldsymbol{\xi}}\Big|\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{x}}\frac{\partial\boldsymbol{\xi}^\top}{\partial\boldsymbol{x}} = \begin{bmatrix} \dfrac{k_2k_3}{2k_1} & 0 & 0 \\ 0 & \dfrac{k_1k_3}{2k_2} & 0 \\ 0 & 0 & \dfrac{k_1k_2}{2k_3} \end{bmatrix} \tag{5.13}$$

Eventually, the local matrix $\widetilde{\boldsymbol{\mathsf{E}}}$ takes the form of

$$\begin{aligned} \widetilde{\boldsymbol{\mathsf{E}}} &= (\boldsymbol{w}^P\otimes\boldsymbol{w}^P\otimes\boldsymbol{w}^P)\Big[\frac{k_2k_3}{2k_1}(\boldsymbol{z}\otimes\boldsymbol{l}\otimes\boldsymbol{l})(\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V)(\boldsymbol{z}^\top\otimes\boldsymbol{l}^\top\otimes\boldsymbol{l}^\top) + \\ &+\frac{k_1k_3}{2k_2}(\boldsymbol{l}\otimes\boldsymbol{z}\otimes\boldsymbol{l})(\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top\otimes\boldsymbol{z}^\top\otimes\boldsymbol{l}^\top) + \\ &+\frac{k_1k_2}{2k_3}(\boldsymbol{l}\otimes\boldsymbol{l}\otimes\boldsymbol{z})(\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V\otimes\widehat{\boldsymbol{w}}^V)(\boldsymbol{l}^\top\otimes\boldsymbol{l}^\top\otimes\boldsymbol{z}^\top)\Big](\boldsymbol{w}^P\otimes\boldsymbol{w}^P\otimes\boldsymbol{w}^P) \end{aligned} \tag{5.14}$$

Consider now the local linear system

$$\widetilde{\boldsymbol{\mathsf{E}}}\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{g}} \tag{5.15}$$

where the matrix $\widetilde{\boldsymbol{\mathsf{E}}}$ is given by (5.14). After some algebra, this system can be transformed into an equivalent form

$$\begin{aligned} &\Big[(k_1^2\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top\otimes\boldsymbol{\mathsf{I}}\otimes\boldsymbol{\mathsf{I}} + \boldsymbol{\mathsf{I}}\otimes(k_2^2\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top\otimes\boldsymbol{\mathsf{I}} + \\ &+\boldsymbol{\mathsf{I}}\otimes\boldsymbol{\mathsf{I}}\otimes(k_3^2\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top\Big](\boldsymbol{w}^P\otimes\boldsymbol{w}^P\otimes\boldsymbol{w}^P)\widetilde{\boldsymbol{y}} = \\ &= \frac{2}{k_1k_2k_3}(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\otimes(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\otimes(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\widetilde{\boldsymbol{g}} \end{aligned} \tag{5.16}$$

which can be viewed as a special case of form (B.1) from Appendix B with

$$\begin{aligned} \boldsymbol{\mathsf{L}}_\alpha &= (k_\alpha^2\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top \qquad \alpha = 1,2,3 \\ \boldsymbol{y} &= (\boldsymbol{w}^P\otimes\boldsymbol{w}^P\otimes\boldsymbol{w}^P)\widetilde{\boldsymbol{y}} \end{aligned} \tag{5.17}$$

and

$$\boldsymbol{r} = \frac{2}{k_1k_2k_3}(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\otimes(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\otimes(\boldsymbol{w}^P\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\widetilde{\boldsymbol{g}} \tag{5.18}$$

Note that all operators L_α can be obtained as $\mathsf{L}_\alpha = \mathsf{L}_0/k_\alpha^2$, where

$$\mathsf{L}_0 = (\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top \tag{5.19}$$

In order to implement the FDM solver, all eigenvalues and eigenvectors of the matrix L_0 should be found. The eigenvalue problem for L_0 can be transformed to a generalized eigenvalue problem as follows

$$[(\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)^{-1}\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top]\boldsymbol{v} = \lambda\boldsymbol{v} \quad\Rightarrow\quad (\boldsymbol{z}\widehat{\boldsymbol{w}}^V\boldsymbol{z}^\top)\boldsymbol{v} = \lambda(\boldsymbol{l}\widehat{\boldsymbol{w}}^V\boldsymbol{l}^\top)\boldsymbol{v} \tag{5.20}$$

and easily solved using standard procedures from the LAPACK library. Note also that, in the spectral element method, the dimension of the above eigenvalue problem will be very small, typically not larger that 10.

6. A projection method for solving a sequence of linear systems with a fixed matrix

The most time-consuming part of the calculations during each time step is the determination of new pressure fields. In order to obtain convergence in a reasonable number of iterations, some nontrivial preconditioning techniques are compulsory. Additionally, the multiplication by the pressure matrix S requires a solution to the internal inversion of the Helmholtz matrix. The approximation of the matrix H^{-1} discussed in Part I can be applied to avoid costly inversion, however, at the price of inaccuracy in the mass balance.

As long as the grid geometry and the time step are fixed, the problem of the pressure evaluation in subsequent time instants consists actually in solving a sequence of algebraic problems with the same matrix and different right-hand sides. Due to the continuity of the flow evolution, the difference between the right-hand side vectors corresponding to subsequent time steps is expected to be small, so the solution in a given time instant may be used as a sensibly accurate initial approximation for iterations in the next time step. One can probably do even better by extrapolating this initial approximation from the recent history of the solution. Such an extrapolation will typically involve several past time steps.

Fisher (1998) proposed a much more sophisticated and efficient approach. It is shortly presented in this paper, however in a slightly modified form, which avoids matrix-vector multiplication appearing in the original formulation.

Consider a sequence of linear systems

$$\mathsf{A}\boldsymbol{x}^{(k)} = \boldsymbol{b}^{(k)} \qquad k = 1, 2, ... \tag{6.1}$$

The matrix of all systems is the same, and it is assumed SPD, large and sparse. The method starts with solving the first system

$$\mathbf{A}\boldsymbol{x}^{(1)} = \boldsymbol{b}^{(1)} \tag{6.2}$$

At this point, we assume that no "special" initial approximations are available. After the solution is obtained, it is normalized as follows

$$\boldsymbol{e}_1 = \frac{\boldsymbol{x}^{(1)}}{\|\boldsymbol{x}^{(1)}\|_A} \tag{6.3}$$

The norm induced by the SDP matrix A is defined as usual as

$$\|\boldsymbol{x}^{(1)}\|_A = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_A} \equiv \sqrt{\langle \boldsymbol{x}, \mathbf{A}\boldsymbol{x}\rangle} \tag{6.4}$$

Assume now that we want to solve the k-th system in sequence (6.1) ($k > 1$), and that we have, previously generated, a set of A-normalized and A-orthogonal vectors $\boldsymbol{e}_1, ..., \boldsymbol{e}_{k-1}$. Then we construct the vector $\widetilde{\boldsymbol{x}}$ as follows

$$\widetilde{\boldsymbol{x}} = \sum_{j=1}^{k-1} \alpha_j \boldsymbol{e}_j \tag{6.5}$$

The coefficients of linear combination (6.5) are chosen in such a way that the orthogonality conditions hold

$$\langle \boldsymbol{e}_j, \boldsymbol{b}^{(k)} - \mathbf{A}\widetilde{\boldsymbol{x}}\rangle_A = 0$$

which implies that

$$\alpha_j = \langle \boldsymbol{e}_j, \boldsymbol{b}^{(k)}\rangle \tag{6.6}$$

The new vector is now computed as

$$\widetilde{\boldsymbol{b}} = \mathbf{A}\widetilde{\boldsymbol{x}} \tag{6.7}$$

and the sought solution is expressed as a sum

$$\boldsymbol{x}^{(k)} = \widetilde{\boldsymbol{x}} + \boldsymbol{x}' \tag{6.8}$$

The correction vector $\boldsymbol{x}'$ will be found from the following linear system

$$\mathbf{A}\boldsymbol{x}' = \boldsymbol{b}^{(k)} - \widetilde{\boldsymbol{b}} \equiv \boldsymbol{b}' \tag{6.9}$$

and the iterations of the PCGM start from the zero vector. Note that the residual vectors of original system (6.1) and new system (6.9) (explicitly available in PCGM iterations) are the same

$$\boldsymbol{r} \equiv \boldsymbol{b}^{(k)} - \mathbf{A}\boldsymbol{x}^{(k)} = \boldsymbol{b}' - \mathbf{A}\boldsymbol{x}' \equiv \boldsymbol{r}' \tag{6.10}$$

and the following stopping criterion is convenient

$$\frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{b}^{(k)}\|_2} < \varepsilon \tag{6.11}$$

In order to continue the calculation, we have to create the next vector $\boldsymbol{e}_k$. Theoretically, it would be sufficient to A-normalize the vector $\boldsymbol{x}'$ and obtain

$$\boldsymbol{e}_k = \frac{\boldsymbol{x}'}{\|\boldsymbol{x}'\|_A} \tag{6.12}$$

Indeed, it is easy to show that if $\boldsymbol{x}'$ were the exact solution to (6.9), then we would have $\boldsymbol{x}' \perp \text{span}\{\boldsymbol{e}_1, ..., \boldsymbol{e}_{k-1}\}$. However, the vector $\boldsymbol{x}'$ is only an approximate solution, so the explicit orthogonalization is recommended. To this end, we write

$$\boldsymbol{x}'_\perp = \boldsymbol{x}' - \sum_{j=1}^{k-1} \beta_j \boldsymbol{e}_j \tag{6.13}$$

where

$$\beta_j = \langle \boldsymbol{e}_j, \boldsymbol{x}' \rangle_A = -\langle \boldsymbol{e}_j, \boldsymbol{r}' \rangle \tag{6.14}$$

The A-norm of the orthogonal projection $\boldsymbol{x}'_\perp$ should be calculated

$$\|\boldsymbol{x}'_\perp\|_A^2 = \|\boldsymbol{x}'\|_A^2 - \sum_{j=1}^{k-1} \beta_j^2 \tag{6.15}$$

It can be shown that the A-norm of the vector $\boldsymbol{x}'$ can be computed without additional matrix-vector multiplication, namely

$$\|\boldsymbol{x}'\|_A^2 \equiv \langle \boldsymbol{x}', \mathbf{A}\boldsymbol{x}' \rangle = \langle \boldsymbol{x}^{(k)}, \boldsymbol{b} - \boldsymbol{r}' \rangle - \sum_{j=1}^{k-1} (\alpha_j + \beta_j)^2 + \sum_{j=1}^{k-1} \beta_j^2 \tag{6.16}$$

Thus, the new vector $\boldsymbol{e}_k$ is obtained as

$$\boldsymbol{e}_k = \frac{\boldsymbol{x}'_\perp}{\|\boldsymbol{x}'_\perp\|} \tag{6.17}$$

where

$$\|\boldsymbol{x}'_{\perp}\|_A \equiv \sqrt{\langle \boldsymbol{x}', \mathbf{A}\boldsymbol{x}' \rangle} = \sqrt{\langle \boldsymbol{x}^{(k)}, \boldsymbol{b} - \boldsymbol{r}' \rangle - \sum_{j=1}^{k-1} (\alpha_j + \beta_j)^2} \tag{6.18}$$

If the number of the linear systems in sequence (6.1) is large, it is practical to interrupt the process of generation of the A-orthogonal basis having N systems completed, and then restart the process by setting

$$\boldsymbol{e}_1 = \frac{\boldsymbol{x}^{(N)}}{\|\boldsymbol{x}^{(N)}\|_A}$$

The number N should be large enough to ensure fast convergence of the iterative solver of system (6.9). On the other hand, the value of N may be limited by the memory available for the additional storage of the basic vectors. Fisher (1998) presented test calculations using various values of N, the largest being equal to 80. The obtained results show that the number of iterations necessary to solve each system (6.9) saturates with increasing N. If such behavior is generic, it may a reasonable decision to use the lowest N ensuring the "saturated" iteration. However, the authors' experience suggests that using even larger values of N is advantageous. Indeed, anytime the sequence of the basic vector is restarted a "transient state" is observed, i.e., during next calculations the number of iterations is much larger than that in the "saturated state". Thus, the larger N is used the better overall performance is obtained, since "transient states" occur more seldom.

7. Numerical simulations and parallel efficiency

The test calculations have been carried out for flows in the T-shaped domain presented in Fig. 2. The hexahedral grids used in the calculations contained 1648 elements (grid A) or 3064 elements (grid B). Densities of the velocity/pressure meshes of the collocation points in each element have been set to $(M_V, M_P) = (5, 3)$ or $(M_V, M_P) = (6, 4)$. However, in the case of the larger grid B, only the smaller density (5,3) of the collocation mesh has been used in order to avoid excessive computational times (all tests have been run on a PC computer with a single Pentium 4 processor)

The dimension of the computational problem depends on the number of spectral elements and the density of the collocation mesh:

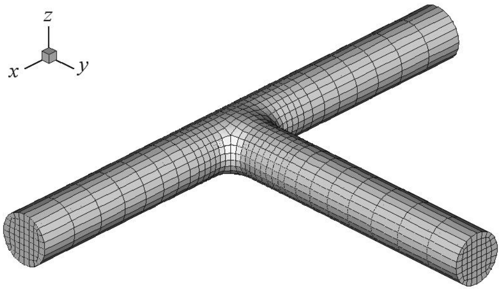

Fig. 2. Computational domain for test calculations (the large grid with 3064 spectral elements is shown)

- grid A with (5,3) collocation mesh: $3 \cdot 99854$ and 44496 unknowns for the velocity and pressure fields, respectively
- grid A with (6,4) collocation mesh: $3 \cdot 197177$ and 105472 unknowns for the velocity and pressure fields, respectively
- grid B with (5,3) collocation mesh: $3 \cdot 189230$ and 83268 unknowns for the velocity and pressure fields, respectively.

The goal of the first part of test calculations was to compare results obtained for different discretization parameters. The sample flow was driven by an oscillating static pressure applied symmetrically to inlet sections 1 and 2. The pressure at outlet Section 3 was fixed in time and equal to zero. The temporal variation of the static pressure at both inlets was given by the following formulae

$$P_{1,2}(t) = 1500 \sin^2(2\pi t) \tag{7.1}$$

The kinematic viscosity ν was set to 0.1 and the time step to $\Delta t = 0.0002$. In order to get an idea how these parameters refer to any physical flow, it was assumed that the length was measured in centimeters. The spatial extent of the computational domain was several centimeters and the diameters of the pipes were about 0.5 cm. Then it was assumed that the density of the fluid was that of water ($1000\,\mathrm{kg/m^3}$), which made the corresponding amplitude of the static pressure to be 150 Pa and the viscosity $10^{-5}\,\mathrm{m^2/s^2}$. The latter value is approximately ten times larger than the viscosity of water, and two and a half times larger than the blood viscosity in large- and medium-sized arteries.

The results of the calculations include time histories of volume fluxes through the inlet and outlet sections and instantaneous patterns of the static pressure and velocity magnitude. The time variation of the volume fluxes obtained for different discretization parameters are presented in Fig. 3. The driving inlet pressure is also plotted to show the phase shift between the forcing and the fluid response. The instantaneous pressure distribution in the symmetry plane of the domain, calculated at the instant of the maximum flow rate is presented in Fig. 4a, while the corresponding field of the velocity magnitude is shown in Fig. 4b. It is clearly seen that most of the hydraulic pressure loss is located (as expected) in the area of pipe connection.

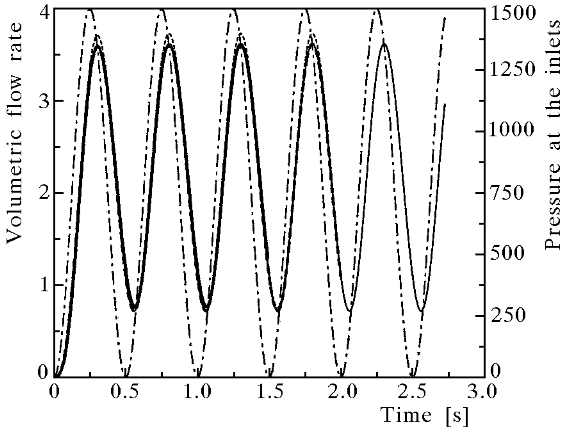

Fig. 3. The volume flux of the test flow plotted as a function of time, calculated for different grids and collocation meshes: grid A with (5,3) mesh (solid line), grid A with (6,4) mesh (thicker dashed line) and grid B with (5,3) mesh (thinner dashed line). The thick grey line corresponds to reference result obtained with FIDAP. The dashed-dotted line depicts the time dependence of the prescribed inlet pressure (formulae (7.1) in the text)

In order to verify the obtained results, analogous tests have been done at the Foundation of Cardiac Surgery Development with the use of the finite-element FIDAP package. The calculations have been carried out on a hexahedral grid containing about 38 thousands of elements. The boundary conditions at all inlet/outlets have been formulated in terms of the static pressure: it has been assumed transversally uniform at each section, with the temporal variation as described above. Additionally, the tangential component of the velocity field was assumed to be zero identically at all inlet/outlet sections. Such setting of the inlet/outlet data seems to be most similar (although not equivalent) to the integral boundary conditions implemented in the spectral solver. Indeed, the results obtained with FIDAP and the spectral solver were in very good agreement (see Fig. 3). In certain aspects, the spectral solver is superior, especially as far as the accuracy of the mass (or volume) balance of the fluid is concerned. Indeed, the relative error of the mass balance in the spectral so-

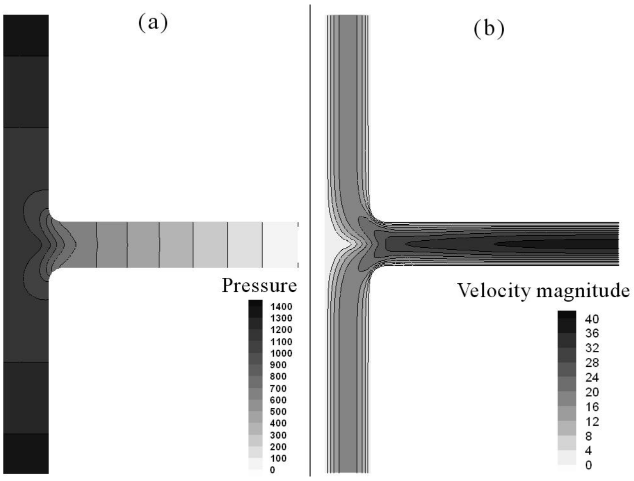

Fig. 4. Instantaneous patterns of the static pressure (a) and the magnitude of the velocity field (b) calculated at the symmetry plane of the test domain, at the time of the maximal flow rate

lver remains of the order of 10^{-12}, while in the FIDAP calculations it was practically difficult to achieve the accuracy level of 1%. Some improvement of FIDAP accuracy would be certainly achieved with further reduction of convergence tolerances, but at the expense of excessive CPU time. Surprisingly enough, the overall performance of the FIDAP and spectral solver turned out to be comparable.

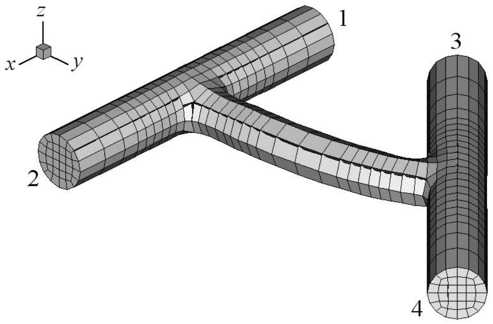

Fig. 5. Simple geometric model of the modified Blallock-Taussig (mBT) shunt, covered with the grid containing 1672 spectral elements

The numerical simulation of a pulsatile flow in a more complicated flow domain depicted in Fig. 5 has also been attempted. The presented configuration of the ducts can be viewed as a geometric model of the (modified) Blallock-

Taussig shunt. The calculations have been performed using a grid with 1672 spectral elements and (5,3) inner collocation meshes ($3 \cdot 101231$ and 45144 unknowns for the velocity and pressure, respectively). The AP-type conditions have been set for all inlet/outlet sections. The section-averaged static pressure applied at sections 1 and 2 was zero, while at sections 3 and 4 it changed in time similarly to real physiological pressure pulses at 120 heartbeats per minute (Fig. 6). The fluid motion has been accelerated from rest. The remarkable observation is that, initially both sections 3 and 4 are the inlets, while at later time the flow direction at section 4 reverses and this section remains the flow outlet for the rest of the simulation. Instantaneous patterns of velocity and pressure fields in the symmetry plane, calculated at during the maximum flow rate, are presented in Fig. 7.

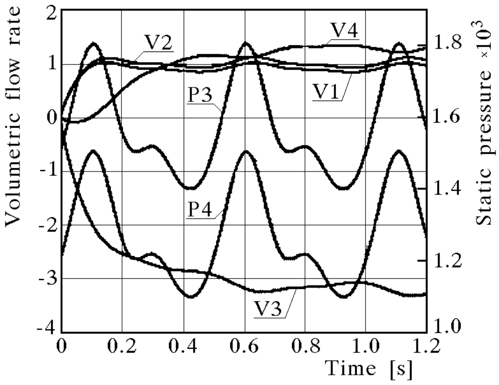

Fig. 6. Temporal variation of the volume fluxes calculated for the computational domain in Fig. 5 using AP-type conditions at all inlet/outlet sections. The lines V3 and P3 show, respectively, the volume flux and static pressure at the inlet section 3. The lines V4 and P4 represent the same data for the outlet section 4. The lines V1 and V2 show the temporal variations of the volume fluxes at the outlet sections 1 and 2, respectively. The pressure applied at these outlets was fixed in time and equal to zero

Finally, the numerical simulations of a pulsatile flow with the VF-type inlet/outlet conditions have been performed. The results shown in the paper have been obtained for the geometric model of the mBT shunt divided into 3760 spectral elements with (5,3) internal mesh of collocation points. The algebraic problems solved at each time step contained $3 \cdot 230303$ and 101520 unknowns for the velocity and pressure fields, respectively. The VF-type conditions were formulated as follows:

- section 3 was the VF-type inlet. The time variation of volumetric flow rate is depicted in the top part of Fig. 8

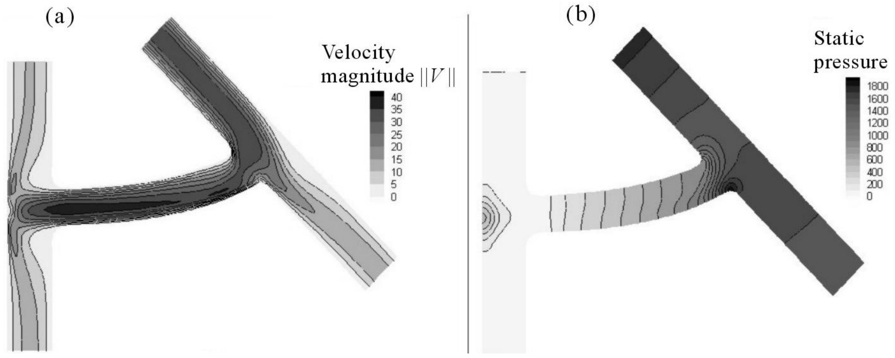

Fig. 7. Pulsatile flow in the mBT shunt calculated using AP-type conditions at all inlet/outlet sections: instantaneous patterns of the magnitude of the velocity (a) and the static pressure (b) calculated in the symmetry plane, at the instant of the maximal flow rate

- sections 1 and 4 were the VF-type outlets. It was assumed that the flow distribution between all outlets was fixed in time: 25% of the incoming volume flux was leaving the computational domain through section 4, while remaining 75% were divided equally between outlets 1 and 2. The latter one was of the AP-type (with the average static pressure set to zero) in order to make the pressure problem uniquely solvable.

As described earlier in the text, the Lagrange multipliers calculated at each time step along with the flow field correspond to instantaneous values of the section-averaged static pressure at all VF-type inlets/outlets. The obtained results are shown in Fig. 8 (bottom part). The reader may note that, initially, the large pressure difference between sections 3 and 4 is necessary in order to avoid the transient reversed flow through section 4, observed in the simulation with pure AP-type conditions described above. Since outlets 1 and 2 are located nearly symmetrically with respect to the shunt's downstream end, the corresponding pressure difference is very small. For presentation purposes, the pressure at outlet 1 shown in Fig. 8 has been multiplied by a factor of -100 (it is actually slightly smaller than the "reference" pressure at outlet 2, equal to zero). Figure 9 shows patterns of the velocity magnitude and pressure fields calculated at the time when the maximum flow rate is attained ($t = 0.6$).

One of the major problems with numerical simulations of three-dimensional flows is computational efficiency. The spectral element discretization in space is an efficient method in the sense that a sufficient spatial resolution can be achieved with a relatively small number of unknowns. On the other hand, there is an "overhead" caused by more complicated forms of data structures and

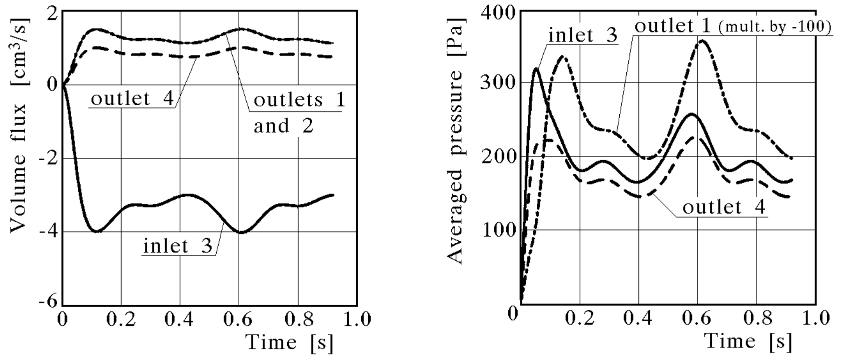

Fig. 8. Time histories of the volume fluxes and the averaged static pressures at the inlet/outlet sections of the geometric model of the mBT shunt, obtained in the simulation with VF-type inlet/outlet conditions. The corresponding fluid density is $\rho = 10^3\,\mathrm{kg/m^3}$ and the kinematic viscosity is $\nu = 10^{-5}\,\mathrm{m^2/s^2}$

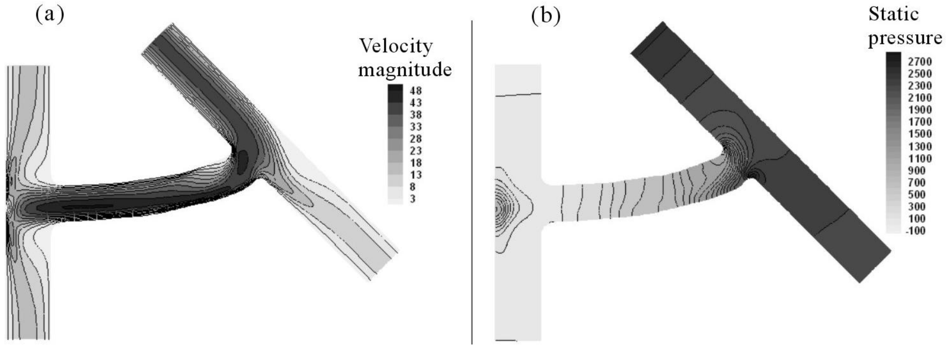

Fig. 9. Pulsatile flow in the geometric model of mBT shunt using VF-type inlet/outlet conditions: instantaneous patterns of the velocity magnitude (a) and the pressure (b) fields in the symmetry plane, calculated at the instant of the maximal flow rate

operations performed on the elementary level, in comparison with "ordinary" FEM. Consequently, the algebraic problems appearing during the solution process are smaller but less sparse and more difficult for efficient preconditioning.

In the current study, most of the numerical simulations described above have been carried out on a Pentium 4 2.6 GHz personal computer. The computational efficiency can be evaluated using a CPU time required to perform one time step of numerical simulation. Clearly, it depends on the number of spectral elements, the dimension of the local collocation mesh and the length of the time step. The most time consuming part of each step is the pressure solver, thus it is essential that it works as quickly as possible. This is achieved by using the two-level preconditioning procedure (Sections 4 and 5) and "smart" selection of the initial approximation based on the projection method (Section 6) originally suggested by Paul Fisher. The pressure preconditioner has been tested on the smaller grid used in the mBT flow simulation (1672 spectral elements, (5,3) internal collocation meshes), at the occasion of determination of particular Stokes solutions (2.12). It has been established that the efficiency of the preconditioning depends on the number of iterations admitted for the internal FDM-based preconditioner (Section 5).

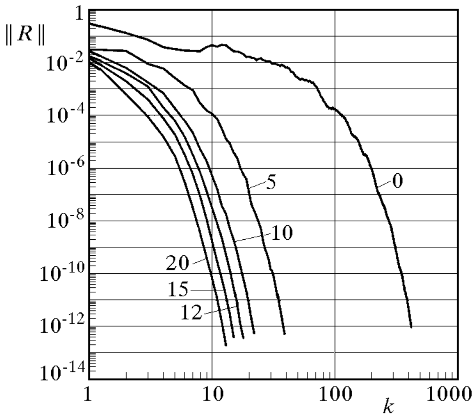

Fig. 10. Convergence history of the pressure preconditioner P5 working with different numbers of iterations of the FDM-based low-level preconditioner

The convergence histories for different numbers of internal iterations are shown in Fig. 10. The use of 15-18 iterations gives effective reductions of CPU times by the factor of 10. Spectacular effects on the computational efficiency are brought also by the Fisher's "trick" (Section 6). We have found it extremely advantageous to use long (even up to 100) sequences of previous pressure states to predict the optimal initial approximation. Only in a few initial steps of each sequence, the pressure solver requires a larger number of iterations to converge. Typically, 8-15 iterations are necessary for the first few steps after the re-start, and after that the number of steps drops immediately. In most of the later steps during each sequence, the pressure solver makes only one or three iterations. This typical behaviour is shown in the Fig. 11. A thick, almost horizontal line corresponds to linear regression, giving an idea about the average numerical cost per one time step. For the case of the T-shape region with 1648 spectral elements and (5,3)-collocation mesh, the average time for one step is around 16 seconds. It also grows almost linearly with the number of spectral elements and the number of local collocation nodes. Convergence of the pressure solver has found to be slower if the dimension of the local collocation mesh was larger. Consequently, the method performs better with a larger number of spectral elements having smaller collocation meshes than in the case of a smaller number of spectral elements having larger collocation meshes.

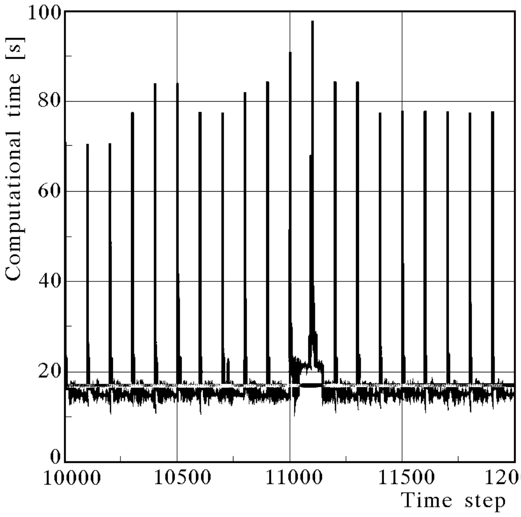

Fig. 11. The figure shows the effect of the Fisher's projection method for predicting an initial approximation for the PCG iterations in the pressure solver. The black line shows the history of the CPU's time of consecutive time steps corresponding to the simulation of flow in the test domain (grid A and (5,3) collocation mesh), carried out on Pentium 4 2.6 GHz workstation. The peeks occur when the pressure base is restarted (every 100 steps), then the "saturated state" is achieved and maintained for the rest of each cycle. The result is significant reduction of the average CPU time per one time step. The thick grey line presents the linear regression of the black-line relation

Another issue is the choice of the time step. If the time step is small, the convergence of the iterative process for the Helmholtz problem is faster since the matrix $\mathbf{A}$ is more diagonally dominant. Consequently, also the pressure solver runs more efficiently. On the other hand, the overall number of time steps needed to cover a certain time interval is larger. Certainly, some restriction on the admissible time step comes from numerical stability since the time-integration method is based on a mixed explicit-implicit approach and, as such, it is only conditionally stable. Typically, the stability limits are due to an explicit treatment of the convection part of the Navier-Stokes equations. However, during the numerical test we encountered unstable behaviour that may be rather related to the "integral" nature of the inlet/outlet conditions used in the study. The characteristic pattern of development of the numerical instability is presented in Fig. 12. Initially, small regions of rapidly growing velocity appear at one or more inlet sections. Later, these regions penetrate the flow domain, merge and lead to a "blow-up" of the numerical solution. It is interesting that the rapid increase of the velocity magnitude is driven mostly by its transversal components. The mechanism of this instability has not yet been resolved, and further computational tests are necessary.

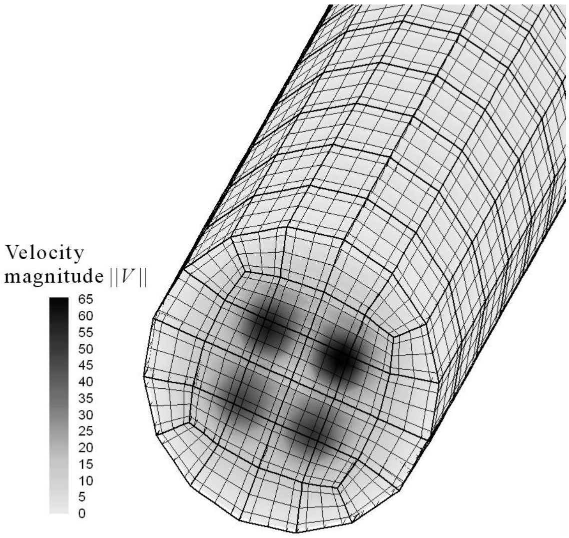

Fig. 12. Characteristic pattern of the initial stage of numerical instability encountered in computations with larger flow rates: small regions of large local velocity appear at one or more inlet/outlet sections

In view of computational complexity of the solver, the parallelization of the code, using the MPI message passing library (version 1.2 of the standard) has been attempted. The parallelization had to be performed in two different ways, due to significant structural differences between various part of the sequential code. During each time step, most of the calculations are organized in the loops running over spatial dimensions and over all spectral elements. In some parts of the code, the spatial-dimensions order is superior meaning that the loops over the elements run as lower-level calculations. Such parts of the code consume about 40% of the computational time. In the remaining 60%, the hierarchy of the computations is reversed: the loops running over the elements are external, and lower-level operations are now spatial-dimensions oriented. Another difficulty is imposed by the data model used in the sequential solver which requires disassembling and assembling of local data from and to global data structures several times in each time step. Moreover, the global structures are used for permanent storage of the velocity and pressure representation, which allows one to avoid the redundant storage of the interfacial data but, on the other hand, may be another source of problems with efficient parallelization.

After the code profiling, the parallelization has been performed on 14 subroutines, which take up approximately 99% of the computational time. The parallelization of the remaining parts of the code was impossible (input/output and indivisible operations), or not sufficiently profitable. The efficiency of the current implementation is summarized in Table 1.

Table 1

Number of CPU's	Efficiency	Acceleration
1	1.00	1.00
2	0.83	1.66
4	0.71	2.84
8	0.53	4.26

The efficiency is defined as the ratio between the parallel code execution time divided by the number of utilized CPUs, and time of execution of a single CPU code. The tests have been performed on four dual Athlon MP2200+ computers, with the Gigabit Ethernet interconnection. To simulate the scalability for 8 separate computers, the communication between 2 CPUs located in the same computer was also routed through the Gigabit Ethernet network, instead of using shared memory.

When using faster interprocessor communication in the computer, the following performance results have been obtained (Table 2).

Table 2

Number of CPU's	Efficiency	Acceleration
1	1.00	1.00
2	0.91	1.81
4	0.76	3.03
8	0.65	5.19

The communication model used in the parallel solver is the synchronous message passing. Due to the large communication overhead in such a model, we develop an asynchronous parallel code that should allow larger speedups and better efficiency.

To improve the performance, we consider re-implementing of the interprocessor communication in one computer with OpenMP, and replacing global data structures by local ones. Such a modification would impose a smaller communication overhead, though some global data relocations may still be necessary, due to changing the order of space- dimensions-oriented and spectral-elements-oriented calculations in each time step.

8. Summary and final remarks

The obtained results prove correctness of the formulation of the numerical method and its spectral element implementation. On the other hand, numerical computations revealed some weak points, which restrain practical usability of the code in the current version. The most important limitation is due to numerical instability of the time integration scheme. The mechanism of this instability has not yet been recognized, although the obtained results may indicate that the source of unstable behaviour lies in the integral inlet/outlet conditions rather that explicit treatment of convection terms. To resolve this vital issue, a further theoretical work and numerous test calculations are necessary. The latter task can be accomplished within a reasonable time only when an efficient parallel version of the code is developed. The MPI-based version using synchronous communications can work efficiently on systems with only a few processors. The performance of such implementation might be slightly improved with appropriate reorganization of source codes (modification of data structures, unified ordering of the element-oriented and coordinate-oriented part of the code, etc.). However, an essential enhancement of the computational efficiency will be achieved when the asynchronous communication is implemented. Such a version of the code is currently tested and the expected speed-up on 32 processors has been estimated to be 20 times. This ratio will certainly increase when appropriate code modifications are made. Further development of the algorithms used in the code is also planned. In particular, the option of curvilinear spectral elements will be incorporated and more sophisticated versions of the pressure solver preconditioning, based on domain decomposition, will be considered.

Appendix A. The Method of Preconditioned Conjugate Gradients

For the completeness of the exposition, we present here a description of the PCG method.

Consider a symmetric and positive definite system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. The PCG method can be summarized as follows

Choose $\boldsymbol{x}^{(0)}$, compute $\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}$, compute $\widetilde{\boldsymbol{r}}^{(0)} = \mathbf{R}^{-1}\boldsymbol{r}^{(0)}$ (preconditioning), $\boldsymbol{p}^{(0)} = \widetilde{\boldsymbol{r}}^{(0)}$.

For $k = 1, 2, ...$

$$\alpha_k = -\frac{(\widetilde{\boldsymbol{r}}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, \boldsymbol{A}\boldsymbol{p}^{(k)})} \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{p}^{(k)}$$

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} + \alpha_k \boldsymbol{A}\boldsymbol{p}^{(k)} \rightarrow \text{ convergence test}$$

$$\widetilde{\boldsymbol{r}}^{(k+1)} = \mathbf{R}^{-1}\boldsymbol{r}^{(k+1)} \qquad \text{(preconditioning)}$$

$$\beta_k = \frac{(\widetilde{\boldsymbol{r}}^{(k+1)}, \boldsymbol{r}^{(k+1)})}{\widetilde{\boldsymbol{r}}^{(k)}, \boldsymbol{r}^{(k)})} \qquad \boldsymbol{p}^{(k+1)} = \widetilde{\boldsymbol{r}}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}$$

Appendix B. Fast Diagonalization Method (FDM)

The Fast Diagonalization Method (see, for instance Couzy and Deville, 1994) is an algorithm for rapid solving of linear algebraic systems with a special tensor-product structure.

Consider a linear algebraic operator

$$\mathcal{L} = \mathbf{L}_1 \otimes \mathbf{I} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{L}_2 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{L}_3 \tag{B.1}$$

where the square matrices $\mathbf{L}_\alpha$, $\alpha = 1, 2, 3$ are given, the symbol $\mathbf{I}$ refers to the identity matrix and the symbol $\otimes$ denotes the tensor product. The dimension of all matrices in (B.1) is equal to N.

The operator $\mathcal{L}$ can be written in an index notation as follows

$$(\mathcal{L})_{i\alpha,j\beta,k\gamma} = (\mathbf{L}_1)_{i\alpha}\delta_{j\beta}\delta_{k\gamma} + \delta_{i\alpha}(\mathbf{L}_2)_{j\beta}\delta_{k\gamma} + \delta_{i\alpha}\delta_{j\beta}(\mathbf{L}_3)_{k\gamma} \tag{B.2}$$

This operator acts on triple-index structures and returns triple-index results according to the formula

$$(\mathcal{L}\boldsymbol{w})_{i,j,k} = \sum_{\alpha=0}^{N-1}(\mathbf{L}_1)_{i\alpha}w_{\alpha,j,k} + \sum_{\beta=0}^{N-1}(\mathbf{L}_2)_{j\beta}w_{i,\beta,k} + \sum_{\alpha=0}^{N-1}(\mathbf{L}_3)_{k\gamma}w_{i,j,\gamma} \tag{B.3}$$

Assume that the matrices $\mathbf{L}_\alpha$, $\alpha = 1, 2, 3$ have complete sets of linearly independent eigenvectors. Then, following equalities hold

$$\mathbf{L}_\alpha \mathbf{P}_\alpha = \mathbf{P}_\alpha \boldsymbol{\Lambda}_\alpha \qquad \alpha = 1, 2, 3 \tag{B.4}$$

where $\mathbf{P}_\alpha$ is the fundamental matrix for $\mathbf{L}_\alpha$ and $\boldsymbol{\Lambda}_\alpha$ is a diagonal matrix containing the corresponding eigenvalues. Then, the following formula for the inverse operator $\mathcal{L}^{-1}$ can be proved

$$\mathcal{L}^{-1} = (\mathbf{P}_1 \otimes \mathbf{P}_2 \otimes \mathbf{P}_3)\boldsymbol{\Lambda}^{-1}(\mathbf{P}_1^{-1} \otimes \mathbf{P}_2^{-1} \otimes \mathbf{P}_3^{-1}) \tag{B.5}$$

where the operator $\boldsymbol{\Lambda}$ is defined as

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_1 \otimes \mathbf{I} \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\Lambda}_2 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{I} \otimes \boldsymbol{\Lambda}_3 \tag{B.6}$$

Consider a system of tlinear equations expressed by the operator $\mathcal{L}$

$$\mathcal{L}\boldsymbol{y} = \boldsymbol{r} \tag{B.7}$$

According to formulas (B.5) and (B.6), the above system can be solved in the following way

- compute the product $\boldsymbol{r}' = (\mathbf{R}_1^{-1} \otimes \mathbf{R}_2^{-1} \otimes \mathbf{R}_3^{-1})\boldsymbol{r}$, with the formula

$$\begin{gathered}(\boldsymbol{r}')_{ijk} = \sum_{\alpha=0}^{N-1} (\mathbf{R}_1^{-1})_{i\alpha} \Big\{ \sum_{\beta=0}^{N-1} (\mathbf{R}_2^{-1})_{j\beta} \Big[\sum_{\gamma=0}^{N-1} (\mathbf{R}_3^{-1})_{k\gamma} (\boldsymbol{r})_{\alpha\beta\gamma} \Big] \Big\} \\ i,j,k = 0,...,N-1\end{gathered} \tag{B.8}$$

- find the solution to the "diagonal" linear system

$$(\boldsymbol{\Lambda}_1 \otimes \mathbf{I} \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\Lambda}_2 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{I} \otimes \boldsymbol{\Lambda}_3)\boldsymbol{r}'' = \boldsymbol{r}'$$

  which is

$$(\boldsymbol{r}'')_{ijk} = (\lambda_i^3 + \lambda_j^2 + \lambda_k^1)^{-1} (\boldsymbol{r}')_{ijk} \tag{B.9}$$

- compute the product $\boldsymbol{y} = (\mathbf{R}_1 \otimes \mathbf{R}_2 \otimes \mathbf{R}_3)\boldsymbol{r}''$ with the formula

$$\begin{gathered}(\boldsymbol{y})_{ijk} = \sum_{\alpha=0}^{N-1} (\mathbf{R}_1)_{i\alpha} \Big\{ \sum_{\beta=0}^{N-1} (\mathbf{R}_2)_{j\beta} \Big[\sum_{\gamma=0}^{N-1} (\mathbf{R}_3)_{k\gamma} (\boldsymbol{r}'')_{\alpha\beta\gamma} \Big] \Big\} \\ i,j,k = 0,...,N-1\end{gathered} \tag{B.10}$$

The most important feature of the above procedure is its computational efficiency. It can be noticed that the total numerical cost can be estimated as, roughly, $6 \cdot N^4$ floating point operations. It means that solving linear system (B.7) is only twice as expensive as the calculation of the "matrix-vector" product (B.3).

Acknowledgements

This work has been supported by the State Committee for Scientific Research (KBN), grant No. 7 T11F 018 20.

References

1. Cahouet J., Chabard J.P., 1986, Some fast 3D finite element solvers for the generalized Stokes problems, *Internat. J. Numer. Meth. Fluids*, **8**, 869-895

2. Couzy W., Deville M.O., 1994, Spectral-element for the Uzawa pressure operator applied to incompressible flows, *J. Sci. Comput.*, **9**, 2, 107-122

3. Deville M.O., Fisher P.F., Mund E.H., 2002, *High-Order Methods for Incompressible Fluid Flow*, Cambridge University Press

4. Fisher P.F., 1998, Projection techniques for iterative solution of $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ with successive right-hand sides, *Comput. Methods. Appl. Mech. Eng.*, **163**, 193-204

5. Formaggia L., Gerbeau J.F., Nobile F., Quarteroni A., 2000, *Numerical Treatment of Defective Boundary Conditions for the Navier-Stokes Equations*, EPFL Preprint 20.2000

6. Karniadakis G.E., Sherwin S., 1999, *Spectral/hp Element Methods for CFD*, Oxford University Press

7. Maday Y., Meiron D., Patera A.T., Ronquist E.M., 1993, Analysis of iterative methods for the steady and unsteady Stokes problem: application to spectral element discretizations, *SIAM J. Sci. Comp.*, **14**, 2, 310-337

8. Ronquist E.M., 1991, A domain decomposition method for elliptic boundary value problems: application to unsteady incompressible fluid flow, *Proceedings of 5th Conference on Domain Decomposition Methods for Partial Differential Equations* (Ed. D.E. Keyes et al.), 545-557

9. Saad Y., 2003, *Iterative Methods for Sparse Linear Systems*, 2nd Ed. SIAM

Wyznaczenie nieustalonych przepływów cieczy lepkiej w trójwymiarowym układzie przewodów. Część II: Implementacja metody spektralnych elementów skończonych i przykładowe wyniki

Streszczenie

Druga część pracy omawia zagadnienia związane z implementacją metody spektralnych elementów skończonych oraz prezentuje wybrane wyniki obliczeń. Przedstawiono również wybrane algorytmy: skalowanie (ang. *preconditioning*) procesu iteracyjnego dla ciśnienia, metodę FDM oraz metodę efektywnego rozwiązywania sekwencji układów liniowych dużych rozmiarów z ustaloną macierzą. Omówiono wyniki obliczeń testowych. Przedyskutowano trudności i ograniczenia aktualnej implementacji oraz zaprezentowano wstępne efekty paralelizacji solwera spektralnego.

Manuscript received April 23, 2004; accepted for print May 25, 2004