Commun. Comput. Phys.
doi: 10.4208/cicp.300412.010213a

Vol. **14**, No. 4, pp. 1103-1119
October 2013

On Direct and Semi-Direct Inverse of Stokes, Helmholtz and Laplacian Operators in View of Time-Stepper-Based Newton and Arnoldi Solvers in Incompressible CFD

H. Vitoshkin* and A. Yu. Gelfgat

School of Mechanical Engineering, Faculty of Engineering, Tel-Aviv University, Ramat Aviv 69978, Tel-Aviv, Israel.

Received 30 April 2012; Accepted (in revised version) 1 February 2013

Communicated by Jie Shen

Available online 27 May 2013

Abstract. Factorization of the incompressible Stokes operator linking pressure and velocity is revisited. The main purpose is to use the inverse of the Stokes operator with a large time step as a preconditioner for Newton and Arnoldi iterations applied to computation of steady three-dimensional flows and study of their stability. It is shown that the Stokes operator can be inversed within an acceptable computational effort. This inverse includes fast direct inverses of several Helmholtz operators and iterative inverse of the pressure matrix. It is shown, additionally, that fast direct solvers can be attractive for the inverse of the Helmholtz and Laplace operators on fine grids and at large Reynolds numbers, as well as for other problems where convergence of iterative methods slows down. Implementation of the Stokes operator inverse to time-stepping-based formulation of the Newton and Arnoldi iterations is discussed.

AMS subject classifications: 76D05

Key words: CFD, Newton iteration, Arnoldi iteration, Stokes operator.

1 Introduction

This study is motivated by many successful applications of an inverse Stokes operator as preconditioners for steady state Newton solvers and Arnoldi eigensolvers [1, 2]. The Stokes operator inverse is considered as an intrinsic part of a pressure/velocity coupled time-dependent Navier-Stokes solver, which connects between time-dependent calculations and direct numerical solution for steady states and their stability. For examples of

*Corresponding author. *Email address:* gelfgat@tau.ac.il (A. Yu. Gelfgat)

a successful use of this technique and results on multiplicity and stability of various flow states we refer to [3–6] and references therein. An extension of this approach for calculation of leading eigenvalues in a prescribed frequency range is proposed in the recent paper [7]. A further extension to study of non-modal optimal disturbances growth is described in [8].

To become an effective preconditioner and to be applied as in [1–8] to large Reynolds number flows, the Stokes operator must be evaluated with a large time step. The latter becomes especially difficult when three-dimensional flows are studied on fine grids making most of traditional iterative methods to converge unacceptably slowly. In particular, we are interested in coupled incompressible pressure-velocity solvers, which are more computationally demanding than segregated ones, but possess important advantages: more stable time integration, correct calculation of pressure at each time step, and a possibility to proceed without pressure boundary conditions. Applied as preconditioners to Newton and Arnoldi solvers the coupled methods are expected to perform well if the Stokes operator with a large time step can be inversed within a relatively short CPU time. Considering 2D stability problems one can apply a direct sparse solver to inverse the 2D Stokes operator [9], however this becomes too memory demanding for fine three-dimensional grids. A similar approach with the same restrictions in 3D cases was implemented in [10] for explicitly calculated Jacobian matrices. At the same time, our recent pressure-velocity coupled multigrid solver [11], which performs well at small time steps fails to converge at large steps needed for 3D stability studies [10,11]. Based on the above experience, in this paper we recall the well-known factorization of the Stokes operator, which we use for computation of its inverse, applying fast direct methods where possible. Using the finite volume method, we arrive to an analog of the Uzawa scheme [12], in which only one matrix, called "pressure matrix" has to be inversed iteratively. As a result, we arrive to a time-stepping method, which may be too CPU-time consuming for a straight-forward time-integration, but yields the inverse of the Stokes operator with a large time step, for which we are seeking.

Another important observation of this study relates to fully three-dimensional time-dependent CFD modelling at very large Reynolds numbers, where all the known iterative methods slow down or fail. Here we observe that for calculation on fine grids and at large Reynolds numbers the eigenvalue decomposition based direct solver [13] becomes more efficient than iterative solvers. Since computational requirements of the direct solver do not depend on the time step and Reynolds number, all the time steps are completed within the same CPU time, which is an attractive feature by itself. This observation is not completely new, but the fact still is not widely recognized. We show also that the eigenvalue decomposition based direct solver allows for an efficient parallelization in a distributed memory multiprocessor computer.

As a preliminary step, we examine computational performance of the above direct solver when implemented in a pressure-velocity segregated time-integration solver. We consider a series of well-known natural convection benchmarks for the purpose of comparing performance of the direct solvers with that of an iterative method. We have chosen

the BiCGstab(2) iteration as a representative example of modern Krylov-subspace-based iteration methods. Our test calculations show that the direct methods perform better than BiCGstab(2) on fine grids and for problems with large Reynolds or Grashof numbers. The same conclusion holds for geometric agglomerated multigrid (GAMG) solver implemented in the OpenFoam package. Since performance of the direct method is independent of problem governing parameters and the time step size, we argue that they can be attractive not only as a part of the Stokes operator inverse, but also for computations at large Reynolds numbers with a high spatial resolution.

The above test calculations allow us, in particular, to estimate computational cost of inverse of the Helmholtz operators needed for the inverse of the Stokes operator. We observe that at small time steps, i.e., for the straight-forward time integration, the pressure Laplacian inversed by the fast direct method can be a good preconditioner. The preconditioned BiCGstab(2) iteration converges in 2-3 iterations for small time steps and within 6-8 iterations for large ones, which we consider as an acceptable performance.

Finally, we perform a test calculation for the time-stepping based Newton steady solver and Arnoldi eigensolver. These computations show that the present approach removes the memory restrictions of the technique proposed in [9], however remains too slow on a scalar computer. It is shown that in contrast to computations of [10], the present approach is efficiently scalable, so that a competitive performance for three-dimensional problems can be expected if massively parallel computations are involved.

2 Symbolic factorization of the Stokes operator

Consider numerical semi-implicit time integration of the incompressible Navier-Stokes equations where linear pressure and velocity terms are treated implicitly and all the other terms - explicitly. Independently on a spatial discretization this leads to a system of linear algebraic equations with the Stokes operator that links velocity and pressure. The Stokes operator acting on velocity $\mathbf{v}=(u,v,w)$ and pressure p can be expressed as

$$\begin{bmatrix} H_u & 0 & 0 & -\nabla_p^x \\ 0 & H_v & 0 & -\nabla_p^y \\ 0 & 0 & H_w & -\nabla_p^z \\ \nabla_u^x & \nabla_v^y & \nabla_w^z & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ R_w \\ 0 \end{bmatrix}. \tag{2.1}$$

Here ∇^x, ∇^y and ∇^z are the first derivatives in the x-, y- and z- directions and $H=\Delta-I/\delta t$ are Helmholtz operators. Δ is the Laplacian operator, I is the identity operator and δt is the time step. The lower indices show on which variable an operator acts. The right hand sides contain the non-linear terms and all other terms that are treated explicitly. The left hand side 4×4 operator matrix assembles the 3D Stokes operator.

For any spatial discretization, we can associate the vector of unknowns in Eq. (2.1) with a vector assembled from all scalar unknowns of the problem, and the operators of

the left hand sides as matrices containing a discretization of the corresponding operator. By assigning the lower indices, we also take into account a possibility of different boundary conditions, as well as discretization of different terms, which takes place, e.g., on staggered grids. Therefore, in all further considerations we assume that $H_u \neq H_v \neq H_w$, $\nabla_u^x \neq \nabla_p^x$, $\nabla_v^y \neq \nabla_p^y$, and $\nabla_w^z \neq \nabla_p^z$.

Treating the Stokes operator as a 4×4 matrix, we derive its LU decomposition. Assigning identity operators to the main diagonal of L yields:

$$\begin{bmatrix} H_u & 0 & 0 & -\nabla_p^x \\ 0 & H_v & 0 & -\nabla_p^y \\ 0 & 0 & H_w & -\nabla_p^z \\ \nabla_u^x & \nabla_v^y & \nabla_w^z & 0 \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ \nabla_u^x H_u^{-1} & \nabla_v^y H_v^{-1} & \nabla_w^z H_w^{-1} & I \end{bmatrix} \begin{bmatrix} H_u & 0 & 0 & -\nabla_p^x \\ 0 & H_v & 0 & -\nabla_p^y \\ 0 & 0 & H_w & -\nabla_p^z \\ 0 & 0 & 0 & C \end{bmatrix}, \tag{2.2}$$

where

$$C = \nabla_u^x H_u^{-1} \nabla_p^x + \nabla_v^y H_v^{-1} \nabla_p^y + \nabla_w^z H_w^{-1} \nabla_p^z. \tag{2.3}$$

It can be easily shown that assigning identity matrices to the main diagonal of U, or application of the Sherman-Morrison-Woodbury formula, or Schur complement equation lead to an equivalent result. The matrix C is an analog of the Schur complement matrix (or Uzawa matrix), which arises in the well-known Uzawa method [12, 20]. Our matrix C, however, is different since we took into account that velocity divergence and pressure gradient operators on staggered grids act on variables defined at different nodes and also result at different nodes. Therefore, these operators cannot always be connected via the transpose operation. Additionally, computation of the divergence in the whole flow region involves velocity boundary values, while computation of the pressure gradient does not use pressure boundary values. Contrarily to the Uzawa matrix, which is symmetric and positive semi defined [20], the matrix C is not necessarily symmetric and its positive definition should be examined for each scheme separately.

It is well known that the Uzawa method usually is not applied directly, but is used as a starting point for definition of various segregated pressure-velocity solvers [12, 20]. We, however, are interested in the implementation of the Stokes operator factorization directly. Using Eqs. (2.2) and (2.3) the solution is obtained in three steps:

Algorithm 2.1:

---

1. Solve $\hat{u} = H_u^{-1} R_u$, $\hat{v} = H_v^{-1} R_v$ and $\hat{w} = H_w^{-1} R_v$ for $\hat{u}$, $\hat{v}$ and $\hat{w}$.
2. Solve $p = -C^{-1}(\nabla_u^x \hat{u} + \nabla_v^y \hat{v} + \nabla_v^z \hat{w})$ for p.
3. Solve $u = \hat{u} + H_u^{-1} \nabla_p^x p$, $v = \hat{v} + H_v^{-1} \nabla_p^y p$, and $w = \hat{w} + H_w^{-1} \nabla_p^z p$.

Thus, calculation of the solution of a 3D problem (2.1) requires 6 inverses of the Helmholtz operator and one inverse of the matrix C. Since the matrix C defines solution for the pressure, we call it "pressure matrix". Note, that if to assume very small time

step δt, and at the steps 2 and 3, as well as in Eqs. (2.1)-(2.3) apply $H \approx I/\delta t$, the operator C turns into the Laplacian of pressure, and step 3 can be interpreted as a projection of intermediate solution ($\hat{u}$, $\hat{v}$, $\hat{w}$) on the divergence-free space. Thus, we arrive to the standard Chorin's projection method.

For the fully coupled implementation of the steps 1-3, one needs inverse of the Helmholtz operators and of the pressure matrix C. The inverse of the Helmholtz operators is usually a part of a pressure-velocity segregated code. Since at small time steps, the Helmholtz operator is close to the identity operator, its iterative inverse typically does not involve any numerical difficulties. As it was mentioned above, in some applications involving Newton and Arnoldi iterations for computation of steady solutions and analysis of their stability, the inverse Stokes operator with a large time step is used as a preconditioner [1,2]. In these cases an iterative inverse of the Helmholtz operators can be problematic. In such cases, the direct methods discussed in the next Section can be called for.

The main difficulty in the implementation of the proposed Stokes operator inverse is computation of the inversed pressure matrix C. It is easy to see that this matrix is singular, which is a usual consequence of the pressure defined up to an additive constant. As we already mentioned, at small time steps this matrix is close to approximation of the pressure Laplacian, thus causing well known problems when inverted. Apparently, a Dirichlet point should be added to make the matrix regular. The pressure boundary conditions, however, can be avoided if calculation of the pressure gradient in step 3 does not involve pressure boundary values. The simplest example of that is the use of staggered grids, as was proposed by Patankar [21], and was implemented, e.g., in our earlier studies [9–11]. In general, to make pressure boundary conditions unnecessary the low-order numerical scheme (i) must close the system of the momentum equation by the continuity equation ($div\mathbf{v} = 0$ in the non-boundary points) and not by a pressure or pressure correction equation that always requires boundary conditions; and (ii) approximation of the pressure gradient in the momentum equations should not involve pressure boundary values.

Since computation of the whole matrix C can be CPU-time and memory consuming, the most natural way of its inverse is implementation of one of the Krylov subspace iteration methods, which requires only calculation of multiplication of the matrix by a vector (the action of a matrix). The latter can be done by using Eq. (2.3) at the cost of two and three Helmholtz operator inverses for 2D and 3D problems, respectively. Our numerical experiments, showed that inverse of C on the 100^2 stretched grid by the BiCGstab(2) method requires 80-100 iterations. At small time steps, the Helmholtz operators tend to the identity operator, so that the whole matrix C tends to the approximation of the Laplacian of pressure. Thus, at small time steps the number of iterations can be significantly decreased by use of the inverse pressure Laplacian matrix as a preconditioner. The latter makes the resulting matrix close to the unity, so that for a 100^2 stretched grid the BICGstab(2) method converges in less than 2-4 iterations for a dimensionless time step of 0.01, and in 6-8 iterations if the time step is increased to the value of 1. The ver-

ification of the above three step algorithms against some known benchmark results is straight-forward and is not reported here. It is emphasized that evaluation of the action of matrix C requires two or three inverses of the Helmholtz operators in 2D and 3D cases, respectively. Therefore it is a CPU-time consuming operation, so that time integration is affordable only if C can be inversed in few iterations, or computation of its action can be done faster.

It is clear at this stage that implementation of the above algorithm to the Stokes operator inverse for time-dependent computations will be much more computationally demanding than most pressure-velocity segregated techniques. A comparison with the latter, if meaningful at all, is beyond the scope of the present paper. It should be emphasized however, that if direct methods are applied to the inverse of the Helmholtz operators, then the inverse of the pressure matrix C remains the only iterative part of the algorithm. We call it semi-direct inverse of the Stokes operator.

A weak dependence of the proposed Stokes operator inverse on the time step allows one to perform calculations with large time steps, which we consider as a prerequisite to applications of methodology of [1, 2] to calculate developed steady three-dimensional flows and to study their stability. The reader is referred to above papers for further details.

3 Computation of the pressure matrix using the tensor-products

As is discussed above, inverse of the pressure matrix C can be done by one of Krylov subspace iteration methods. Implementation of these methods involve computation of the Krylov basis, for which one needs calculation of the matrix-vector product $C\mathbf{v}$, called action of C on a vector $\mathbf{v}$. In the following we show that the matrix C, as well as its action, can be computed via tensor products of smaller matrices composed from the eigenvectors and eigenvalues of one-dimensional operators.

First, we recall the result of Lynch *et al*. [13] that allows one to inverse the Helmholtz operators with any time step and at any Reynolds number within the same computational effort.

Consider a Laplace operator acting on a scalar function $u(x,y,z)$, defined on a region $0\le x\le a$, $0\le y\le b$, $0\le z\le c$. The function u satisfies Dirichlet, Neumann or mixed linear homogeneous boundary conditions. We assume that the region is covered by an orthogonal grid, that divides the three directions into N_x, N_y and N_z points. We denote discretization of the second derivatives as operators D_{xx}, D_{yy}, D_{zz} that are one-dimensional and act on a row or a column of the grid function $u_{ijk}=u(x_i,y_j,z_k)$. Representing D_{xx}, D_{yy} and D_{zz} by matrices and following notations of the Kronecker (tensor) product we write the discretized Poisson equation as

$$\Delta u=[D_{xx}\otimes I_y\otimes I_z+I_x\otimes D_{yy}\otimes I_z+I_x\otimes I_y\otimes D_z z]u=f, \tag{3.1}$$

where I_x, I_y and I_z are identity matrices of the order N_x, N_y and N_z, respectively, $\otimes$ denotes the tensor product, and $f_{ijk}=f(x_i,y_j,z_k)$ is the discretized right hand side of the

Poisson equation. For the following we assume that the eigenvalue decompositions of matrices D_{xx}, D_{yy} and D_{zz} are known and are represented as

$$D_{xx}=E_x\Lambda_x E_x^{-1}, \qquad D_{yy}=E_y\Lambda_y E_y^{-1}, \qquad D_{zz}=E_z\Lambda_z E_z^{-1}. \tag{3.2}$$

Here E_x, E_y and E_z are square matrices of the order N_x, N_y and N_z, respectively, whose columns are eigenvectors of the matrices D_{xx}, D_{yy} and D_{zz}. Λ_x, Λ_y and Λ_z are diagonal matrices having the eigenvalues of D_{xx}, D_{yy} and D_{zz} on their diagonals. According to [13] the solution of Eq. (3.1) can be represented as

$$u=\left(E_x\otimes E_y\otimes E_z\right)\Lambda^{-1}\left(E_x^{-1}\otimes E_y^{-1}\otimes E_z^{-1}\right)f, \tag{3.3}$$

where

$$\Lambda=(\Lambda_x\otimes I_y\otimes I_z)+(I_x\otimes\Lambda_y\otimes I_z)+(I_x\otimes I_y\otimes\Lambda_y) \tag{3.4}$$

is a diagonal matrix of the order $N_xN_yN_z$, whose diagonal values are $\Lambda_{ij}=\Lambda_{x,i}+\Lambda_{y,j}+\Lambda_{z,k}$. For the Helmholtz equation $(\Delta+aI)u=f$ the solution is also given by Eq. (3.3) with $\Lambda_{ij}=\Lambda_{x,i}+\Lambda_{y,j}+\Lambda_{z,k}+a$.

The eigenvalue decompositions (3.2) are computed in $\mathcal{O}(N_x^3)$, $\mathcal{O}(N_y^3)$ and $\mathcal{O}(N_z^3)$ operations, respectively. For a time-marching procedure this computation is needed only once, so that its computational cost can be neglected if the number of time steps is sufficiently large. Once these are known, the calculation of the solution via Eqs. (3.3) and (3.4) requires $2N_xN_yN_z(N_x+N_y+N_z)$ multiplications and $N_xN_yN_z$ divisions by Λ_{ijk}. Therefore the total amount of multiplications and divisions is $N_xN_yN_z(2N_x+2N_y+2N_z+1)$. In the following we refer to Eqs. (3.3) and (3.4) as the tensor product factorization (TPF) solver proposed in [13]. As mentioned, this solver is often applied together with spectral and pseudospectral methods [14–16], however its application together with lower-order spatial discretization, remains rare [17–19]. Regarding the lower-order methods, two additional comments should be made. First, increase of an approximation order by use of longer stencils will not increase the computational cost of TPF implementation, however performance of any iterative methods will be affected due to lesser sparseness of the matrices. Second, short three- or five-points stencils usually used in lower order methods allow one to replace the eigenvalue decomposition in one of directions by the Thomas algorithm. This will retain the direct inverse of the matrices, but will decrease the overall computational time. Below we call this approach the TPT solver. Apparently, the direction with a maximal number of grid points should be chosen for such a replacement. The Thomas algorithm applied for N grid points and a scheme defined on the 3-point stencil requires $5N$ multiplications and divisions, which is significantly less than N^2 multiplications needed for computations of the mass-vector product. Assuming in the above $N_x=N_y=N_y\gg 5$ we see that application of the Thomas algorithm in one direction reduces the number of operations almost twice for a 2D case and by a factor of approximately $2/3$ for a 3D case.

Clearly, the TPT or TPF factorization can be used to inverse Helmholtz operators involved in the definition of the pressure matrix C, Eq. (2.3). Then, to complete computation of the action of C, the first derivative operators ∇^x, ∇^y, and ∇^z can be evaluated numerically. At the same time, defining the latter operators via the tensor products as

$$\nabla^x_u = D^x_u \otimes I_y \otimes I_z, \qquad \nabla^y_v = I_x \otimes D^y_v \otimes I_z, \qquad \nabla^z_w = I_x \otimes I_y \otimes D^z_w; \tag{3.5a}$$

$$\nabla^x_p = D^x_p \otimes I_y \otimes I_z, \qquad \nabla^y_p = I_x \otimes D^y_p \otimes I_z, \qquad \nabla^z_p = I_x \otimes I_y \otimes D^z_p. \tag{3.5b}$$

The terms of Eq. (2.3) can be expressed in the following form:

$$\nabla^x_u H^{-1}_u \nabla^x_p = \left(D^x_u E^x_u \otimes E^y_u \otimes E^z_u\right) \Gamma^{-1}_u \left(E^{x\,-1}_u D^x_p \otimes E^{y\,-1}_u \otimes E^{z\,-1}_u\right), \tag{3.6a}$$

$$\nabla^y_v H^{-1}_v \nabla^y_p = \left(E^x_v \otimes D^y_v E^y_v \otimes E^z_v\right) \Gamma^{-1}_v \left(E^{x\,-1}_v \otimes E^{y\,-1}_v D^y_p \otimes E^{z\,-1}_v\right), \tag{3.6b}$$

$$\nabla^z_w H^{-1}_w \nabla^z_p = \left(E^x_w \otimes E^y_w \otimes D^z_w E^z_w\right) \Gamma^{-1}_w \left(E^{x\,-1}_w \otimes E^{y\,-1}_w \otimes E^{z\,-1}_w D^z_p\right), \tag{3.6c}$$

where Γ are diagonal $(N_x N_y N_z) \times (N_x N_y N_z)$ matrices, whose diagonal elements are

$$(\Gamma_u)_{ijk} = -\tau^{-1} + (\Lambda^u_x)_i + \left(\Lambda^u_y\right)_j + (\Lambda^u_z)_k, \tag{3.7a}$$

$$(\Gamma_v)_{ijk} = -\tau^{-1} + (\Lambda^v_x)_i + \left(\Lambda^v_y\right)_j + (\Lambda^v_z)_k, \tag{3.7b}$$

$$(\Gamma_w)_{ijk} = -\tau^{-1} + (\Lambda^w_x)_i + \left(\Lambda^w_y\right)_j + (\Lambda^w_z)_k. \tag{3.7c}$$

The expressions (3.6a)-(3.7c) form the analytical representation of the pressure matrix C. Formally, it can be calculated and inversed, which completes a fully direct inverse of the Stokes operator. Practically, evaluation of the above Kronecker products is CPU-time consuming, but possible. The matrix C is not sparse, so that its direct inverse is practically unaffordable. These expressions can also be used for evaluation of the matrix C action. Calculating the one-dimensional matrices like $D^u_x E^u_x$ and $E^{u^{-1}}_x D^p_x$ and keeping them in memory, makes the computational effort needed for each of Eqs. (3.6a)-(3.6c) equal to that of Eq. (3.3), so that action of C consumes computational time exactly equal to computation of three Helmholtz operator inverses using TPF. Moreover, the similar structure of Eqs. (3.6a)-(3.6c) and Eq. (3.3) allows for the same parallelization algorithm, an example of which is given in Appendix.

Calculating the matrix C and observing its components we find that many of them are, in fact, numerical zeroes. After nullifying all the components that are at least 10 orders of magnitude smaller than the leading term of the same row, we discover that the matrix C is, in fact, a sparse matrix. The sparseness is most profound when the uniform grid is used, for which more than 95% of the matrix elements are numerical zeroes. This allows us to apply a sparse matrix solver and to keep the *LU* decomposition of the resulting matrix. Then carrying out of step 2 of Algorithm 2.1 reduces to computation of sparse back/forward substitutions, which makes the inverse of the Stokes operator fully analytical. For stretched grids this approach is possible, however not efficient due to a lower level of sparseness.

4 Test calculations

4.1 TPF and TPT solvers versus BiCGstab(2)

In the following we show several examples illustrating that the eigenvalue decomposition approach may be more effective than an iterative solution. In particular, this justifies the use of TPT/TPF direct solvers for the Stokes operator inverse. As a representative example, we consider several benchmark problems on natural convection in laterally heated two-and three-dimensional cavities [15,23–25]. We solve the Helmholtz and Poisson equations both by Jacobi (diagonal) preconditioned BiCGstab(2) iteration, by the tensor product factorization (TPF) method, and by the tensor product method combined with the Thomas algorithm (TPT) in one of the directions. In all the results reported both approaches yielded numerical solutions that coincided at least to within the tenth decimal digit. After establishing the equivalence of all the three solutions we compare the consumed CPU times.

The choice of Jacobi preconditioner for the BiCGstab(2) iteration is justified by diagonal dominance of the matrices and its negligible computational cost. We are aware of the fact that Krylov subspace iteration methods, like BiCGstab(2), with a smarter choice of a preconditioner, can perform faster than they do in the following test calculations. However, the choice of a preconditioner is usually problem-dependent, the effect we want to avoid. Moreover, we believe that the qualitative conclusions we derive will hold also in the case of more efficient iteration techniques applied to large Reynolds number CFD problems.

Along with the BiCGstab(2), we tried to apply multigrid solvers for inverse of the same operators. For grids with 100 and more nodes in each spatial direction the classical geometric multigrid algorithm with a 4-level V-cycle was implemented with different smoothers and under-relaxation at the prolongation step. Performance of SOR and LSOR with regular, zebra and RB ordering, ADI and SIP smoothers was checked. A considerable effort was devoted to optimization of the relaxation parameters of the smoothers and prolongation, however this version of multigrid method never converged faster than BiCGstab(2). We also tried a geometric agglomerated multigrid (GAMG) solver implemented in Open Foam package. For the pressure equations on fine 100^2 and 100^3 grids this solver converged approximately two times faster than BiCGstab(2).

We performed the time integration with the finite volume in space and three-time level discretization as it was done in [10, 11]. The test problems were convection of air ($Pr = 0.71$) in a laterally heated square cavity [23, 24], in a two-dimensional cavity with height-to-width ratio $A = 8$ [25], and in a laterally heated cubical box [15]. The finite volume staggered grids were stretched near the boundaries. The numerical method and the code are already completely verified [9–11]. Here we are interested only in comparison of consumed CPU times. To do that we start from the laterally heated square cavity and perform time-dependent calculations for $Gr = 10^5$ until convergence to a steady state. Then we set the Grashof number to $Gr = 10^6$, use the calculated steady state as an initial

condition, and carry out 10,000 time steps. Then we again increase the Grashof number by an order of magnitude to $Gr=10^7$, and perform 10,000 more time steps. All the runs were carried out on a 100^2 nodes grid with the time step $\delta t=0.01$, using either the BiCGstab(2), TPF or TPT method. We also examined the supercritical oscillatory flow in a tall vertical cavity [25] and compared the computation cost of a calculation over 5 oscillation periods. The calculation was carried out on the grid with 100×800 nodes, for $A=8$ and $Gr=4.8\times 10^7$. To perform calculations on an already converged limit cycle solution, we used a snapshot of the oscillatory flow computed in [11] on the same grid as the initial state. The time step was $\delta t=0.001$.

For all two-dimensional problems the TPF and TPT methods consumed approximately the same CPU time for all of the unknown functions: u, v, T (which satisfy a Dirichlet problem for the Helmholtz equation) and δp (which satisfies a Neumann problem for the Poisson equation). The TPT method was faster in agreement with the above estimations. The BiCGstab(2) iterations converged much faster for the Helmholtz problems than for the Poisson problem. This is because the Helmholtz operator, which is close to being a perturbation of the identity operator, is far better conditioned than the Laplacian with Neumann boundary conditions, and hence requires many fewer BiCGstab(2) iterations to converge.

For the two-dimensional tests, BiCGstab(2) required between 10 and 33 times longer to solve the Poisson problem than to solve one of the Helmholtz problems. In consequence, BiCGstab(2) is about 3 to 5 times *faster* than TPF for each of the Helmholtz problems and about 5 times *slower* than TPF for the Poisson problem. Moreover, with the increase of the Grashof number, the BiCGstab iterations converge slower, which is quite expected, while the CPU time consumptions of the TPF method does not change. Therefore, one can expect that for the flows with larger Grashof (or Reynolds) numbers, the TPF (or TPT) approach can become even more attractive. This suggests combining BiCGstab (or another iterative solver) to calculate the temperature and velocity with the TPF or TPT solver to calculate the pressure.

In the test calculations for the three-dimensional problem, we also started from $Gr=10^5$. After carrying out 10,000 time steps, with the time step $\delta t=0.001$, we increased the Grashof number to $Gr=10^6$ and then, after another 10,000 time steps to $Gr=10^7$. These calculations were performed for stretched grids consisting of 50^3, 75^3 and 100^3 nodes. To explore the potential scalability of the TPF and TPT approaches we carried out these computations twice, using either scalar or vector processors. The results can be summarized as follows. In the three-dimensional calculations, the TPF approach is always faster than BiCGstab. For a scalar processor, the ratio $t_{\mathrm{BiCG}}/t_{\mathrm{TPF}}$ of CPU times is between 1.5 (for a Helmholtz problem) and 12 (for a Poisson problem), while for a vector processor, this ratio is between 2.5 and 50. Some details are given in Table 1. As expected, the CPU time consumed by BiCGstab(2) increases with the Grashof number, as well as with the grid refinement. The CPU time consumed by the TPF and TPT approaches is Grashof-number (or Reynolds-number) independent and grows with the mesh refinement according to the operation counts discussed in Section 3. As mentioned, the ratios $t_{\mathrm{BiCG}}/t_{\mathrm{TPF}}$ and

Table 1: CPU times (sec) consumed for 10,000 time steps by the BiCGstab iterative solver (t_{BiCG}) and by the present eigenvalue decomposition solvers (t_{TPF} and t_{TPT}) for convection of air ($Pr=0.71$) in laterally heated cubical cavity. Calculation on a single vector Xeon(R) CPU 5355 2.66 GHz processor.

Problem	$Gr=10^5$, 50^3 grid			$Gr=10^6$, 50^3 grid			$Gr=10^7$, 50^3 grid		
Variable	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}
T	76.2	29.12	20.40	76.3	28.9	20.25	76.3	29.0	20.32
u	132.2	27.39	19.19	133.7	27.3	19.13	135.94	27.5	19.27
v	131.9	27.45	19.23	133.3	27.3	19.13	135.51	27.6	19.34
w	72.2	27.29	19.12	42.3	27.3	19.13	72.4	27.4	19.20
δp	724.7	29.32	20.54	1166.5	29.2	20.46	1222.1	29.3	20.53
total	1137.3	140.6	98.49	1582.2	140.0	98.09	1642.3	140.7	98.65

Problem	$Gr=10^5$, 75^3 grid			$Gr=10^6$, 75^3 grid			$Gr=10^7$, 75^3 grid		
Variable	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}
T	369.9	104.8	72.24	269.2	105.1	72.45	269.1	105.1	72.45
u	547.4	117.2	80.79	551.1	117.1	80.72	551.4	117.5	81.00
v	543.3	116.7	80.45	546.5	116.5	80.31	547.1	116.9	80.58
w	257.6	109.5	75.48	257.9	109.5	75.48	257.7	109.7	75.62
δp	3186.4	104.4	71.97	4696.8	104.7	72.17	5454.7	104.9	72.31
total	4904.6	552.6	380.93	6321.5	552.8	381.14	7080.0	554.0	381.97

Problem	$Gr=10^5$, 100^3 grid			$Gr=10^6$, 100^3 grid			$Gr=10^7$, 100^3 grid		
Variable	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}	t_{BiCG}	t_{TPF}	t_{TPT}
T	872.7	384.4	262.81	877.1	383.9	262.47	6256	384.7	263.01
u	1298.4	354.8	242.57	1307.4	355.1	242.78	1310.1	355.1	242.78
v	1293.0	357.5	244.42	1300.3	357.7	244.55	1302.4	356.9	244.01
w	600.5	361.9	247.43	601.4	360.9	246.74	601.3	361.9	247.43
δp	9036.4	383.7	262.33	12133.4	383.77	262.38	14152.1	384.2	262.67
total	13101.0	1842.3	1259.55	16219.6	1841.3	1258.92	17991.5	1842.8	1259.89

$t_{\mathrm{BiCG}}/t_{\mathrm{TPT}}$ obtained on a vector processor are significantly larger than the ratio obtained for a scalar processor. The proportion between the ratios corresponding to the scalar and vector processors for 50^3 and 75^3 grids varies between the values 3 and 4, which corresponds to the length of vector (equal to 4) in the processor used. This ratio is larger for finer grids of 75^3 and more nodes, which also can be expected. The ratios $t_{\mathrm{BiCG}}/t_{\mathrm{TPF}}$ and $t_{\mathrm{BiCG}}/t_{\mathrm{TPT}}$ obtained on a vector processor (Table 1) for 75^3 grid is larger than that obtained for a 100^3 grid, so that the grid refinement does not necessarily lead to a more profound difference in the consumed CPU times. At the same time in all the 3D cases considered, the TPF and TPT methods perform significantly faster than BiCGstab(2), and the CPU time ratios $t_{\mathrm{BiCG}}/t_{\mathrm{TPF}}$ and $t_{\mathrm{BiCG}}/t_{\mathrm{TPT}}$ grow with the increase of the Grashof (Reynolds) number. This shows that time-dependent fully three-dimensional calculations may benefit if iterative solvers are replaced by the fast direct ones. More details on these test

calculations can be found in `http://arxiv.org/abs/1107.2461`.

To verify the above conclusion using another independent code we have solved the benchmark problems for convection in the square and cube cavities by Open Foam package. Since Open Foam uses different schemes the comparison cannot be straight-forward, so that we focus on the time consumed by the Open Foam pressure solver. First, we solved the pressure problem using different solver offered by the package and found that for our test problems the generalized geometric/algebraic multigrid (GAMG) solver with the Gauss-Seidel smoother converges faster than all other versions of smoothers and preconditioned BiCG algorithms. Then, using this version of GAMG we repeated calculations for 100^2 and 100^3 grids. Comparing the consumed CPU times we observe that GAMG solver performs almost two times faster than our version of BiCGstab(2). However, with the increase of the Grashof number CPU time consumed by the GAMG solver grows similarly to BiCGstab(2) and for large Grashof numbers remains significantly larger than that consumed by TPF and TPT solvers.

4.2 Test calculations for the Stokes operator inverse

For test calculations below we consider the above benchmark on convection in a laterally heated square cavity. First we studied computational cost of the inverse of the matrix C by the BiCGstab(2) iteration for a time-marching code. We used the same discretization in space and time as in the previous calculations, however, no fractional time step or pressure correction is needed when the Stokes operator is inversed at every time step.

The total CPU time spent for the inverse of C when a steady state at $Gr=10^5$ is calculated, starting from the solution at $Gr=10^4$, was 611 sec for 8200 time steps. Then, for 10,000 time steps the code consumed 919 and 1021 sec for the inverse of C at $Gr=10^6$ and 10^7, respectively. Thus, the observed slow-down of the convergence with the increase of the Grashof number is not large. It is clear, however, that total computational cost of a single time step is significantly larger than that of a segregated method, so that this approach should be applied only in the cases when, say, correct pressure values are needed, or numerical stability of the time integration should be improved.

Performance of the Newton method was tested for calculation of the steady state at $Gr=10^7$ using a solution for $Gr=5\times10^6$ as initial guess. The whole Newton process converges in 6 iterations. The time step δt was varied between 1 and 100, however, for $\delta t\geq 80$ the iterations for the inverse of C diverge. The total consumed CPU time, the number of BiCGstab(2) iterations needed for calculation of the current Newton correction and the maximal number of BiCGstab(2) iterations needed for the inverse of pressure matrix C, are reported in Table 2. We observe that starting from $\delta t=2$ the whole Newton iteration process completes in 700-800 seconds. Clearly, the number of iterations and the consumed CPU time are problem dependent, making it impossible to find a generally optimal time step. Thus, for example, calculation of the steady state at $Gr=10^8$ using the solution at $Gr=5\times10^6$ and $\delta t=10$ as an initial guess consumes more than 2 CPU hours. The same Newton process as reported in Table 2, implemented with the approach pro-

Table 2: Number of BiCGstab(2) iterations needed to compute the Newton correction N_{Newton} for the first five Newton iterations and maximal number of BiCGstab(2) iterations N_C needed for the inverse of pressure matrix C. Computation of steady state of the benchmark problem of [23,24] at $Gr=10^7$ starting from the steady state at $Gr=5\times10^6$ as an initial guess. For $\delta t\geq 80$ iterations for the inverse of C diverge. Single scalar Intel 2.4 GHz processor.

iteration	1		2		3		4		5		t_{total}
δt	N_{Newton}	N_C	N_{Newton}	N_C	N_{Newton}	N_C	N_{Newton}	N_C	N_{Newton}	N_C	(sec)
1	64	368	21	141	41	229	11	79	1	52	1312
2	48	139	31	130	29	72	9	43	1	64	846
5	61	96	27	70	31	67	9	63	5	55	762
10	74	96	38	61	17	58	37	50	2	31	793
20	99	64	34	124	45	55	19	46	10	33	845
30	94	326	40	300	54	54	35	29	13	24	806
40	33	122	39	42	38	38	52	28	5	22	720
50	140	228	40	225	43	41	46	33	18	38	799
60	132	235	46	51	54	34	51	67	1	19	753
70	129	135	43	33	45	31	69	112	1	17	729

Table 3: Number of Krylov vectors $N_{vectors}$, maximal number of BiCGstab(2) iterations needed to compute the Krylov vectors for Arnoldi iterations $N_{Arnoldi}$ and maximal number of BiCGstab(2) iterations N_C needed for the inverse of pressure matrix C. Calculation of the dominant eigenvalues corresponding to the steady state of the benchmark [23,24]: $\lambda=(-0.02812,0)$ at $Gr=10^7$ and $\lambda=(-0.032,0.8663)$ at $Gr=10^8$.

$Gr=10^7$					$Gr=10^8$				
δt	$N_{vectors}$	$N_{Arnoldi}$	N_C	t_{total} (sec)	t	$N_{vectors}$	$N_{Arnoldi}$	N_C	t_{total} (sec)
					0.08	16	834	746	38013
2	16	128	931	17438	0.1	16	877	755	38491
5	16	99	180	7603	0.2	16	672	903	42660
10	16	103	875	6012	0.3	16	592	968	47491
20	16	116	801	5980	0.4	16	532	979	50949
30	16	141	641	6091	0.5	16	484	945	52224
40	16	149	736	5658	0.6	16	480	979	55974

posed in [10], completes in less than 5 seconds on the same processor. Clearly the present approach is more than 100 times slower. At the same time, compared to the approach of [10], it has two important advantages. First, it removes a heavy memory restriction that allows one to compute fully developed 3D steady state flows. Second, the TPT decomposition applied for inverse of the Helmholtz and Laplace operators is scalable, while the backward/forward substitutions of sparse and packed LU decompositions used in [10] are not scalable.

Performance of the inversed Stokes operator preconditioned Arnoldi iteration (for computation of the leading eigenvalues is illustrated in Table 3. The same benchmark problem as above was considered for two values of the Grashof number 10^7 and 10^8. The ARPACK package was implemented. In both cases 16 Krylov basis vectors are sufficient

to meet the ARPACK convergence criterion of 10^{-6}. In the first case, $Gr=10^7$, the leading eigenvalue is real and no shift is needed. The numerical experiment shows that iterations converge for the time step $2\leq\delta t\leq 40$ and diverge beyond this interval. For $10\leq\delta t\leq 40$ the whole process converges in less than an hour, and again most of CPU time is spent for the inverse of the pressure matrix C. The whole process, as well as the convergence of the C inverse, significantly slows down when the Grashof number is increased to $Gr=10^8$ and the leading eigenvalue becomes complex. The convergence of the whole process is observed now for $0.08\leq\delta t\leq 1$, which shows that not only optimal time step, but also the convergence yielding one is problem-dependent. The CPU time needed to calculate a single eigenvalue is beyond 10 hours, while the approach of [10] yields the same result in less than one minute. It is stressed again, however, that the present approach removes the memory restrictions and allows for scalable computations, which is not the case of [10].

5 Concluding remarks

In this study we offer methods for direct and semi-direct inverse of the Stokes operator using an extended Uzawa method. Our pressure matrix C is an analog of the Uzawa matrix, however not necessarily symmetric or positive semi defined. As in the Uzawa method, the inverse of the pressure matrix C is the main bottleneck of the whole calculation. Analytical expressions for a direct calculation of the pressure matrix are derived, seemingly for the first time. We have discussed a possibility of direct inverse of the matrix C, which, along with the direct methods used for the Helmholtz operator inverse, would make inverse of the whole Stokes operator fully analytical. Unfortunately, this approach appears to be computationally effective only for uniform grids. For the general case of non-uniform grid we propose a semi-analytic inverse of the Stokes operator, where only the pressure matrix C is inversed iteratively by one of Krylov-subspace iteration methods.

We have shown that factorization of the Stokes operator followed by a fast direct method inverse of the Helmholtz and Laplacian operators, and the preconditioned Krylov subspace iterations used to inverse the pressure matrix, allow one to perform computations with a large time step. The latter is needed for application of inverse Stokes operator preconditioned Newton and Arnoldi methods for calculation of three-dimensional steady states, and analysis of their modal and non-modal stability [1–8]. The corresponding test calculations showed that in this way the heavy memory demands of the approach [10] can be removed. On the other hand, the calculations can become fast enough only with a massive parallelization of the basic Laplace/Helmholtz operator direct inverse. The parallelization is algorithmically straight-forward and is presented as well.

We have shown additionally that the tensor product factorization (TPF) method, possibly combined with the Thomas solver (TPT), is sometimes, but not always, faster than iterative methods, when applied in pressure/velocity segregated solvers. Our numerical experiments made for incompressible Boussinesq equations showed that the direct TPF

and TPT solvers can perform faster than an iterative method on fine grids and for large Reynolds (Grashof) numbers, when convergence of any iterative method slows down. It is emphasized that since the methods are direct, their computational cost depends only on the problem size, but not on governing parameters, which may make it attractive for cases where iterative methods converge too slowly.

Acknowledgments

We gratefully acknowledge the contribution of Laurette Tuckerman to this work. We are deeply thankful to Yuri Feldman for his help with multigrid methods and scalability analysis. This research was supported by a grant from the Ministry of Science, Culture & Sport, Israel & the Ministry of Research, France, grant No. 3-4293, and by the German-Israeli Foundation, grant No. I-954-34.10/2007. Parallel HPC computations were supported by the LinkSCEEM-2 project, funded by the European Commission under the 7th Framework Program through Capacities Research Infrastructure, INFRA-2010-1.2.3 Virtual Research Communities, Combination of Collaborative Project and Coordination and Support Actions (CP-CSA) under grant agreement no RI-261600.

Appendix: Parallelization of TPT and TPF methods

In spite of the fact that computation of a solution according to Eq. (3.3) consists only of matrix multiplications, a straight-forward application of scalable linear algebra tools (e.g., ScaLapack) does not yield a good speedup due to too large amount of inter-CPU communications. To arrive to an efficient scalability we note that the eigenvector matrices E_x, E_y, E_z, and their inverses are small compared to the matrices containing right hand sides and solutions. This allows one to keep copies of these matrices in local memory of each CPU, so that no redistribution is needed. The matrices containing right hand sides are distributed between the processors. Assume that the processors are arranged in a grid $p\times q$, so that the size of a block of $N_x\times N_y\times N_z$ r.h.s. matrix contained in each CPU is $N_x\times n_y\times n_z$, where $n_y=N_y/p$ and $n_z=N_z/q$. For simplicity we assume that n_y and n_z are exact integers. This distribution allows us to perform multiplication by E_x and E_x^{-1}, or application of the Thomas algorithm in x-direction independently on each CPU. Multiplication by two other eigenvector matrices $E_y\otimes E_z$ (or $E_y^{-1}\otimes E_z^{-1}$) is done in the following way. Assume that R_{iml} is the $N_x\times N_y\times N_z$ matrix, which is a result of multiplication of the r.h.s. matrix F_{ijk} by $E_y\otimes E_z$:

$$R_{iml}=\sum_{k=1}^{N_z}\sum_{j=1}^{N_y}E_{z,lk}E_{y,mj}F_{ijk}=\sum_{K=1}^{q}\sum_{J=1}^{p}\hat{R}_{iml}^{KJ}, \tag{A.1a}$$

$$R_{iml}^{KJ}=\sum_{k=n_z(K-1)+1}^{n_zK}\sum_{j=n_y(J-1)+1}^{n_yJ}E_{z,lk}E_{y,mj}F_{ijk}. \tag{A.1b}$$

The partial sums $\hat{R}^{KJ}_{iml}$ are calculated in each CPU independently, while the final summation is performed as a series of reduction operations. Each CPU controls those reduction operations that result in values distributed in its local memory. The latter helps to achieve a uniform distribution of work between the processors. The main disadvantage of this approach is a necessity of keeping partial sums $\hat{R}^{KJ}_{iml}$ of the size $N_x \times N_y \times N_z$ in each CPU. The numerical tests performed on a cluster with 64 CPUs showed that we reach speedup of 28 and 58 on 32 and 64 CPUs, respectively.

References

[1] L.S. Tuckerman, D. Barkley, Bifurcation analysis for time-steppers, in Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems, K. Doedel and L. Tuckerman (Eds.), IMA Volumes in Mathematics and Its Applications, vol. 119 (2000), Springer, New York, pp. 453-466.

[2] L.S. Tuckerman, F. Bertagnolio, O. Daube, P. Le Quéré, D. Barkley, Stokes preconditioning for the inverse Arnoldi method, in Continuation Methods for Fluid Dynamics, D. Henry and A. Bergeon (Eds.), Notes on Numerical Fluid Dynamics, vol. 74 (2000), Vieweg, Göttingen, pp. 241-255.

[3] K. Borońska, L.S. Tuckerman, Extreme multiplicity in cylindrical Rayleigh-Bénard convection. I. Time dependence and oscillations, Phys. Rev. E, 81 (2010), 036320.

[4] K. Borońska, L. S. Tuckerman, Extreme multiplicity in cylindrical Rayleigh-Bénard convection. II. Bifurcation diagram and symmetry classification, Phys. Rev. E, 81 (2010), 036321.

[5] C. Beaume, A. Bergeon, E. Knobloch, Homoclinic snaking of localized states in doubly diffusive convection, Phys. Fluids, 23 (2011), 094102.

[6] L. Lu, G. Papadakis, Investigation of the effect of external periodic flow pulsation on a cylinder wake using linear stability analysis, Phys. Fluids, 23 (2011), 094105.

[7] X. Garnaud, L. Lesshaft, P.J. Schmid, J.-M. Chomaz, A relaxation method for large eigenvalue problems, with an application to flow stability analysis, J. Comput. Phys., 231 (2012), 3912-3927.

[8] D. Barkley, H.M. Blackburn, S.J. Sherwin, Direct optimal growth analysis for timesteppers, Int. J. Numer. Meths. Fluids, 57 (2008), 1435-1458

[9] A. Yu. Gelfgat, Stability of convective flows in cavities: Solution of benchmark problems by a low-order finite volume method, Int. J. Numer. Meths. Fluids, 53 (2007), 485-506.

[10] Yu. Feldman, Direct numerical simulation of transitions and supercritical regimes in confined three dimensional recirculating flows, PhD Thesis, Tel-Aviv University, 2011.

[11] Yu. Feldman, A. Yu. Gelfgat, On pressure-velocity coupled time-integration of incompressible Navier-Stokes equations using direct inversion of Stokes operator or accelerated multigrid technique, Comput. Struct., 87 (2009), 710-720.

[12] M.O. Deville, P.F. Fischer, E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge, 2002.

[13] R.E. Lynch, J.R. Rice, D.H. Thomas, Direct solution of partial differential equations by tensor product methods, Numer. Math., 6 (1964), 185-199.

[14] R. Peyret, Spectral Methods for Incompressible Viscous Flows, Springer, 2002.

[15] E. Tric, G. Labrosse, M. Betrouni, A first incursion into the 3D structure of natural convection of air in a differentially heated cavity, from accurate numerical solutions, Int. J. Heat Mass Transf., 43 (1999), 4043-4056.

[16] M. A. Herrada, C. Del Pino, J. Ortega-Casanova, Confined swirling jet impingement on a flat plate at moderate Reynolds numbers, Phys. Fluids, 21 (2009), 013601.
[17] E. Barbosa, O. Daube, A finite difference method for 3D incompressible flows in cylindrical coordinates, Computers & Fluids, 34 (2005), 950-971.
[18] E. Vedy, S. Viazzo, R. Schiestel, A high-order finite difference method for incompressible fluid turbulence simulations, Int. J. Numer. Meths. Fluids, 42 (2003), 1155-1188.
[19] T. Bjøntegaard, Y. Maday, E.M. Rønquist, Fast tensor-product solvers: Partially deformed three-dimensional domains, J. Sci. Comput., 39 (2009), 28-48.
[20] C. Bacuta, A unified approach for Uzawa algorithms, SIAM J. Numer. Anal., 44 (2006), 2633-2649.
[21] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, McGraw-Hill, New York, 1980.
[22] M.-J. Ni, M.A. Abdou, A bridge between projection methods and SIMPLE type methods for incompressible Navier-Stokes equations, Int. J. Numer. Meths. Fluids, 72 (2007), 1490-1512.
[23] G. de Vahl Davis, I.P. Jones, Natural convection in a square cavity: A comparison exercise, Intl. J. Numer. Meths. Fluids, 3 (1983), 227-264.
[24] P. Le Quéré, Accurate solutions to the square thermally driven cavity at high Rayleigh number, Computers & Fluids, 20 (1991), 29-41.
[25] M.A. Christon, P.M. Gresho, S.B. Sutton, Computational predictability of time-dependent natural convection flows in enclosures (including a benchmark solution), Int. J. Numer. Meths. Fluids, 40 (2002), 953-980.