Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona, Spain, July 9-13, 2018

ICCFD10-069

Parallel Preconditioners for Pressure-Velocity Matrix Systems for Incompressible Flows

Krishna Chandran*
Corresponding author: kchandrn@iitk.ac.in

* Dept. of Mechanical Engineering
Indian Institute of Technology Kanpur, India.
PINCODE 208016

Abstract: Parallel preconditioners for matrix systems arising from an unstructured finite volume formulation for the general thermal transport problem are studied with the primary focus on the CPU time of simulation. The fluid is assumed incompressible. The pressure, velocity and temperature matrix equations are solved using a preconditioned-BiCGSTAB algorithm with the matrices represented in the compressed sparse row format. The pressure matrices are highly ill-conditioned and require powerful preconditioners. Velocity and temperature matrices being well-conditioned require computationally inexpensive preconditioners such as the diagonal preconditioner. Although SGS and ILU preconditioners have better convergence characteristics, the present work shows that the use of these preconditioners for well-conditioned matrix could prove detrimental to the overall simulation time due to its weak parallelizability. Sparse Aproximate Inverse (SPAI) based preconditioners have better convergence and are highly suitable for parallel computing and hence is used for the pressure matrix. Parallelization is based on the OpenMP framework. The relatively high setup time required to compute the approximate inverse is compensated by tailoring the discretized governing equations for pressure correction such that the approximate inverse needs to be computed only once at the beginning of the simulation. The present work shows that this new formulation enables a computationally inexpensive way of using SPAI preconditioner which guarantees a superior convergence and an overall reduction in CPU time when compared with diagonal, SGS and ILU(0) preconditioners.

Keywords: Parallel Preconditioner, Sparse Approximate Inverse Preconditioner, SIMPLE, Pressure Matrix, Weighted Least Squares Gradient.

1 Introduction

Discretized Navier-Strokes equations generally result in a system of equations which are solved iteratively depending on the number of grid points used. Unstructured finite volume method which has acquired wide popularity in large-scale applications result in linear systems for which the matrix structure may be highly sparse thereby making the iterative solution more difficult. Early finite volume formulations were based on SIMPLE and its variants [1] which were used to resolve the pressure-velocity decoupling by using a staggered grid arrangement. However, this approach is inappropriate for unstructured grid where the pressure and velocity need to be defined at the cell centroid. The smoothing pressure correction approach [2] is a remedy which eliminates the checkerboard oscillations for pressure and velocity defined on a non-staggered grid.

Being implicit in nature the SIMPLE family of algorithms form two types of linear system of equations ($A\mathbf{x}=\mathbf{b}$), one for velocity and the other for pressure. For heat transfer related problems one additional temperature matrix resulting from the discretized energy balance equation needs to be solved. The matrices formed are generally non-symmetric and hence iterative methods such as BiCGSTAB [2] and GMRES [4] are used. Difficulty in storage arises when the matrix dimension becomes large. Focus has been shifted towards compressed storage where only the non-zero entries of the matrix are stored. Compressed Sparse Row [5] is one such storage method where the non-zero entries are stored in a row-wise manner with additional arrays to store the location of non-zero entries. Unstructured formulation generally gives rise to matrices which are ill-conditioned with large condition numbers which affects the speed and accuracy of solution.

Preconditioning of the matrix improves its condition number towards unity. Constructing a good preconditioner could prove to be computationally expensive. Diagonal/Jacobi preconditioners are the cheapest of all the preconditioners but effective only for diagonally dominant systems. Gauss Siedel family of preconditioners such as GS, SGS, SOR and SSOR demand more storage but exhibit an improvement in the convergence of the iterative solver. ILU class of preconditioners are more robust with better convergence properties. Both the Gauss Seidel and ILU based preconditioners are weakly parallelizable. Sparse approximate inverse preconditioners, belong to a class of preconditioners where the approximate inverse is explicitly computed. The high setup time to find the approximate inverse should be minimized by highly parallelizable algorithms. Most of the approximate inverse are based on the Frobenius norm minimization leading to inherent parallelism since each columns of approximate inverse is computed independently. Difficulty lies in capturing a good sparsity pattern for the approximate inverse. Grote and Huckle [6] showed an inexpensive way to compute the location of the non-zero entries of approximate inverse without generating excessive *fill-in*. Chow and Saad [7] proposed an algorithm which automatically generates new entries while the excessive *fill-in* are controlled following a dropping strategy. The algorithm does not assume an a priori sparsity pattern hence, the computational complexity reduces but overall computational cost increases in terms of setup time. The other class of approximate inverse falls in the category of incomplete factorization. Benzi and Tuma [8] proposed a direct method using a bi-conjugation algorithm for the incomplete factorization. The method is weakly parallelizable when compared to the minimization problem. Approximate inverse preconditioners are gaining popularity in GPU architecture. Geveler et al. [9] showed that sparse approximate inverse preconditioners provides strong smoothing for unstructured grids in GPU architecture. Akay et al. [10] studied the scalability aspects of various preconditioners based on topology optimization wherein the continuous removal or addition of the material from the computational domain leads to a highly ill-conditioned system. The authors showed that for such problems sparse approximate inverse SAI preconditioner has better scalability than block ILU. The pressure matrix formed from the pressure Poisson's equation is generally ill-conditioned and requires robust parallel preconditioners which give accelerated convergence. Several recent works [11-14] suggest a growing importance towards the improvement in finding better parallel preconditioners for the pressure matrix.

The present work focuses in the accelerated performance of pressure-velocity-temperature matrix system in terms of overall CPU time of simulation with the implementation of explicit preconditioners for the matrix systems. SPAI is chosen as the preconditioner for the pressure matrix. When approximate inverse preconditioners are used it is imperative that in order to have maximum speed up in terms of CPU time the pressure matrix coefficients should remain constant. In the SIMPLE family of algorithms, coefficients of the pressure matrix depend on the velocity field within the non-linear iteration. A simple modification in the pressure correction equation facilitates the approximate inverse to be computed only once at the beginning of the simulation for the pressure matrix.

2 Numerical methodology

2.1 Discretisation

The governing equations for conservation of mass, momentum and energy for incompressible fluid flow in the finite volume framework are given in vector notation as:

$$\int_{CS} (\rho \mathbf{u}) \cdot \mathbf{dS} = 0 \tag{1}$$

$$\underbrace{\frac{\partial}{\partial t}\int_{CV}(\rho \mathbf{u})\,dV}_{\text{transient}} + \underbrace{\int_{CS}(\rho \mathbf{u}\otimes\mathbf{u}).\mathbf{dS}}_{\text{convective}} = \underbrace{-\int_{CV}\nabla p\,dV}_{\text{pressure gradient}} + \underbrace{\int_{CS}\mu\nabla\mathbf{u}\cdot\mathbf{dS}}_{\text{diffusive}} + \underbrace{\Gamma\int_{CV}\rho\mathbf{g}\beta\left(T-T_{ref}\right)dV}_{\text{body force}} \tag{2}$$

$$\Gamma\left\{\underbrace{\frac{\partial}{\partial t}\int_{CV}\left(\rho c_p T\right)dV}_{\text{transient}} + \underbrace{\int_{CS}\left(\rho c_p T\mathbf{u}\right).\mathbf{dS}}_{\text{convective}} = \underbrace{\int_{CS}k\nabla T\cdot\mathbf{dS}}_{\text{diffusive}}\right\} \tag{3}$$

Here the symbol Γ in Eqs (1) and (2) is 1 for non-isothermal flows and 0 for isothermal flows. The body force term is accounted in the momentum equation (2) due to the Boussinesq approximation. Here, **u** is the velocity vector and *p* is pressure field in three dimension. Equation (1) in the discrete form is given as:

$$\left(\rho_p \mathbf{u}_p^{l+1} - \rho_p \mathbf{u}_p^{l_0}\right)\frac{\Delta V}{\Delta t} + \sum_{f=1}^{Nf}\left(\rho \mathbf{u}^{l+1}\mathbf{u}^{l+1} - \mu\nabla\mathbf{u}^{l+1}\right)_f.\mathbf{dS}_f = -\nabla p_p^{l+1}\Delta V + \Gamma\rho\mathbf{g}\beta\left(T^{l+1}-T_{ref}\right)\Delta V \tag{4}$$

Equation (4) is solved by assuming a pressure field at the previous iteration level value *l*. The notation l_0 denotes the old time level while subscripts *p* and *f* stand for the cell centroid and face values. Higher order upwinding is used to calculate the convective fluxes [15], being linearized at the previous iteration level. The temperature in the body force term is assumed the previous iteration level value in the temperature loop i.e. for every linearized iteration level for velocity and pressure there is an additional temperature loop. Equation (4) reduces to a system of linear algebraic equations for velocity given as:

$$AP_{\mathbf{u}}\mathbf{u}_P^{\frac{l+1}{2}} - \sum_k AE_{\mathbf{u},k}\mathbf{u}_{E,k}^{\frac{l+1}{2}} = \nabla p^l\Delta V + \Gamma\rho\mathbf{g}\beta\left(T^{l+1,m}-T_{ref}\right)\Delta V + \frac{\rho\Delta V}{\Delta t}\mathbf{u}_P^{l_0} \tag{5}$$

$$AP_{\mathbf{u}} = \frac{\rho\Delta V}{\Delta t} + \sum_{k=1}^{Nk}\left[C_{\mathbf{u}pk} + D_{\mathbf{u}pk}\right] = \frac{\rho\Delta V}{\Delta t} + \sum_{k=1}^{Nk}AE_{\mathbf{u},k}$$

$AP_{\mathbf{u}}$ is the diagonal term and $AE_{\mathbf{u},k}$ namely, the sum of convective ($C_{\mathbf{u}pk}$) and diffusive ($D_{\mathbf{u}pk}$) fluxes form the off-diagonal terms.

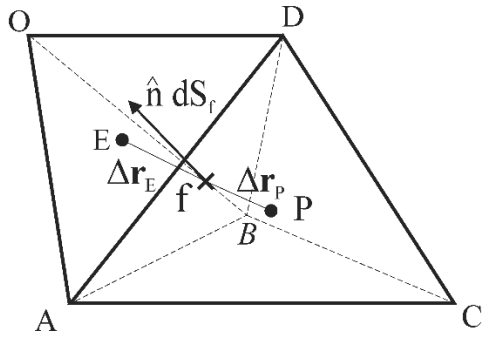

Figure 1. Two neighbouring tetrahedral cells of an unstructured mesh. Points P and E are the cell centroids of the tetrahedral control volumes A-B-C-D and A-B-D-O respectively. Point '*f*' falls on the common face defined by the unit normal $\hat{n}$. The discretized equations are constructed for the control volume A-B-C-D.

The next iteration level for the velocity at (*l*+1)/2 is obtained by solving Eq. 4 with preconditioned-BiCGSTAB, a linear iterative solver. The new values of the fluxes are calculated by updating the previous iteration level values of the velocity with the new one i.e. $\mathbf{u}^l = \mathbf{u}^{(l+1)/2}$. The governing equation for pressure is derived by enforcing the mass balance at the new iteration level, namely

$$\int_{CS} \left(\rho \mathbf{u}\right)_f^l \cdot \hat{n} dS = \int_{CS} \left(\frac{\rho \alpha \Delta V}{AP_{\mathbf{u}}} \nabla p' \right)_f \cdot \hat{n} dS \tag{6}$$

Equation (6) is the pressure Poisson's equation for the pressure correction term p' in the integral form. When discretized, it reduces to another set of linear algebraic equations for pressure correction given as:

$$AP_{p'} p'_P - \sum_k AE_{p',k} p'_{E,k} = \sum_k C_{pk}^l \text{ where } AP_{p'} = \sum_k AE_{p',k} \tag{7}$$

The diagonal term $AP_{p'}$ and the off-diagonal terms $AE_{p',k}$ of the pressure matrix are given as:

$$AE_{p',k} = \frac{\rho \alpha A_{fk}^2}{AP_{\mathbf{u}}}, \; AP_{p'} = \sum_{k=1}^{Nk} AE_{p',k} \tag{8}$$

The term A_{fk} represents the area of the face *k* and α is the under-relaxation factor for velocity. The coefficients of the pressure matrix depend on the velocity field through the term $AP_{\mathbf{u}}$, a sum of transient, convective and the diffusive fluxes. Generally a weighted averaging which is a 2nd order approximation is used to evaluate $AP_{\mathbf{u}}$ at the face A-B-D shown in Fig. 1; specifically:

$$\left(AP_{\mathbf{u}}\right)_f \approx \frac{\Delta \mathbf{r}_E \left(AP_{\mathbf{u}}\right)_p + \Delta \mathbf{r}_P \left(AP_{\mathbf{u}}\right)_E}{\Delta \mathbf{r}_E + \Delta \mathbf{r}_P} \tag{9}$$

By using Eq.(8) it is seen that the coefficient $AP_{\mathbf{u}}$ will vary across the control surface in Eq. (5) since the neighbouring value of $\left(AP_{\mathbf{u}}\right)_E \neq \left(AP_{\mathbf{u}}\right)_P$. In order to make the pressure matrix coefficients constant the term $AP_{\mathbf{u}}$ across all the faces should remain the same. This is achieved if the cell centroid value itself is used as the face value. However, it then becomes a first order approximation. In order to retain the second order accuracy the gradient $\nabla\left(AP_{\mathbf{u}}\right)_p$ needs to be computed. Since the velocity field is known, the central coefficient $AP_{\mathbf{u}}$ is also known at every cell centroid location and the coefficient is now a function of space $AP_{\mathbf{u}} = AP_{\mathbf{u}}\left(\mathbf{r}\right)$. The gradient $\nabla\left(AP_{\mathbf{u}}\right)_p$ is computed using the weighted least squares gradient approach.

2.1.1 The weighted least squares gradient method

Let ϕ be the central coefficient of the discretized momentum equation (5), $\phi = AP_{\mathbf{u}}\left(\mathbf{r}\right)$. The value of ϕ at face *f* is evaluated by expanding the Taylor series at the cell centroid *p*:

$$\phi_f \approx \phi_p + \Delta \mathbf{r}_f \cdot \nabla \phi_p + O\left(\Delta r^2\right)$$

$$\Delta \mathbf{r}_f = \frac{N_f}{\sum_{i=1}^{N_f} \frac{1}{\Delta \mathbf{r}_{fi}}} = \frac{N_f}{\sum_{i=1}^{N_f} \frac{1}{\left|\mathbf{r}_p - \mathbf{r}_{fi}\right|}} \tag{10}$$

Here *f* denotes the face, N_f denotes the number of faces within a control volume, $\mathbf{r}_p$ is the position vector at the cell centroid, $\mathbf{r}_f$ is the position vector at a cell face centroid. A harmonic average of the distance from the cell centroid to all the face centroids is taken to make $\Delta \mathbf{r}_f$ constant. Harmonic average

ensures that the ϕ_f values are not over-predicted, especially when the mesh cells are skewed. For finding $\nabla\phi_p$ the neighbouring cell centroid values of ϕ are used. Let *i* be the neigbouring cell centroid index, then the Taylor series expansion at *i* gives:

$$\phi_i \approx \phi_p + \Delta\mathbf{r}_i \cdot \nabla\phi_p + O\left(\Delta r^2\right) = \phi_p + \left(x_i - x_p\right)\phi_{p,x} + \left(y_i - y_p\right)\phi_{p,y} + \left(z_i - z_p\right)\phi_{p,z} + O\left(\Delta r^2\right)$$

A functional can be defined as follows [16]:

$$\Psi\left(w_i, \phi_i, \phi_{p,x}, \phi_{p,y}, \phi_{p,z}, ...\right) = \sum_{i=1}^{N} w_i \left[\Delta\phi_i - \left\{\Delta x_i \phi_{p,x} + \Delta y_i \phi_{p,y} + \Delta z_i \phi_{p,z}\right\}\right]^2$$

$$\Delta\phi_i = \phi_i - \phi_p, \; w_i = \frac{1}{\left|\mathbf{r}_i - \mathbf{r}_p\right|^2} \tag{11}$$

In Eq. (11) w_i are the weights, *N* denotes the number of neighbouring control volumes. The objective is to find value of the derivatives $\phi_{p,x}, \phi_{p,y}, \phi_{p,z}$ by solving the minimization of the functional $\min_{\phi_{p,x},\phi_{p,y},\phi_{p,z}}\left(\Psi\right) : \frac{\partial\Psi}{\partial\phi_{p,x}} = \frac{\partial\Psi}{\partial\phi_{p,y}} = \frac{\partial\Psi}{\partial\phi_{p,z}} = 0$. The resultant system of equation is solved using the least squares approach to get the values of the derivatives.

$$\begin{bmatrix} \sum w_i \Delta x_i^2 & \sum w_i \Delta x_i \Delta y_i & \sum w_i \Delta x_i \Delta z_i \\ \sum w_i \Delta x_i \Delta y_i & \sum w_i \Delta y_i^2 & \sum w_i \Delta y_i \Delta z_i \\ \sum w_i \Delta x_i \Delta z_i & \sum w_i \Delta y_i \Delta z_i & \sum w_i \Delta z_i^2 \end{bmatrix} \begin{bmatrix} \phi_{p,x} \\ \phi_{p,y} \\ \phi_{p,z} \end{bmatrix} = \begin{bmatrix} \sum w_i \Delta\phi_i \Delta x_i \\ \sum w_i \Delta\phi_i \Delta y_i \\ \sum w_i \Delta\phi_i \Delta z_i \end{bmatrix} \tag{12}$$

Once the values of the derivatives are found it is substituted in Eq. (10) to get the ϕ value interpolated at the face. The computed value at the faces are all the same hence it is taken out of the integral in Eq. (6). Since the pressure matrix now depends only on the geometrical and the fluid parameters which remain constant throughout the simulation it is cost effective to compute approximate parallel preconditioners for the pressure matrix which accelerates the convergence of the iterative solver.

Once the pressure correction field is obtained from Eq. (7) the mass conserving pressure correction term is calculated as follows:

$$p'_m = p' - p'_s = p' - 0.5\left(p - \bar{p}\right) \tag{13}$$

In Eq. (13), p'_s is the smoothing pressure correction term and $\bar{p}$ is evaluated by multidimensional averaging [2]. The term $\bar{p}$ eliminates the checkered board oscillations inherent in the collocated grid formulation. Pressure and velocity at the cell centers are calculated at the new iteration level as:

$$p^{l+1} = p^l + \beta p'_m \tag{14}$$

$$\mathbf{u}^{l+1} = \mathbf{u}^l - \frac{\alpha\ \Delta V}{AP_{\mathbf{u}}}\nabla p'_m \tag{15}$$

In Eqs. (14) and (15), factors α and β are under-relaxation parameters for velocity and pressure. The temperature equation (3) is solved once the new iteration level values of velocity are found. The discrete form of Eq. (3) also forms another set of linear algebraic equation for temperature:

$$AP_T T_P^{l+1,m+1} - \sum_k AE_{T,k} T_{E,k}^{l+1,m+1} = \frac{\rho \Delta V}{\Delta t} T_P^{l_0,m}$$
$$AP_T = \frac{\rho c_p \Delta V}{\Delta t} + \sum_{k=1}^{Nk} \left[C_{Tpk} + D_{Tpk} \right] = \frac{\rho c_p \Delta V}{\Delta t} + \sum_{k=1}^{Nk} AE_{T,k} \quad (16)$$

C_{Tpk}, D_{Tpk} are the respective convective and diffusive fluxes. Here m is the iteration loop variable for temperature. Eq. (16) is repeatedly solved till $\left\| T^{l+1,m+1} - T^{l+1,m} \right\| < \varepsilon$. Once converged, solution algorithm is advanced for the new iteration level value. For convergence, two types of residual norms are evaluated. One is for the non-linear velocity iteration and the other, for mass conservation which is given by $\left\| \nabla^2 p' \right\|_2$. Both these residuals have to converge below a set tolerance limit for time level advancement. The overall calculation procedure is summarized in an algorithmic form as follows:

Let l_0 and (l+1) be the old and new time levels while l and (l+1)/2 are the old and new iteration levels during flow calculation within a time step.

1. Prescribe a pressure field at the previous (initial) time level l_0. Start iterations with $\mathbf{u}^l = \mathbf{u}^{l_0},\ p^l = p^{l_0}, T^{l+1} = T^{l_0}$ for time advancement.
2. Evaluate the face fluxes $C_{\mathbf{u}pk}$ and $D_{\mathbf{u}pk}$ at faces k. Compute the matrix coefficients $AE_{\mathbf{u},k}$ and $AP_{\mathbf{u}}$. Solve Eq. (7) till convergence (tol_**u** < 1.0e-10) to get the intermediate level velocity field $\mathbf{u}^{(l+1)/2}$.
3. Update $\mathbf{u}^l = \mathbf{u}^{(l+1)/2}$. Update nodal values. Evaluate the face fluxes C_{pk}.
4. Compute matrix coefficients $AE_{p',k}, AP_{p'}$ and solve the pressure Poisson equation Eq. (7) till convergence (tol_p' < 1.0e-10) to get the new pressure corrections. Compute the mass-conserving pressure correction p'_m from Eq. (13).
5. Solve Eqs. (14) and (15) and update pressure and velocity at the new time level (l+1).
6. Solve Eq. (16) to get the new iteration level value for temperature at m+1 at the new time level (l+1).
7. Check for convergence: $\left\| T^{l+1,m+1} - T^{l+1,m} \right\| < \varepsilon$. If not converged, update old iteration level values $T^{l+1,m} = T^{l+1,m+1}$ and return to step 2.
8. Calculate momentum and mass residuals and check additionally for convergence. If not converged, update the iteration level values $\mathbf{u}^l = \mathbf{u}^{(l+1)}$, $p^l = p^{(l+1)}$ and return to step 2.
9. Update old time level values with the new as $\mathbf{u}^{l_0} = \mathbf{u}^{l+1},\ p^{l_0} = p^{l+1}, T^{l_0} = T^{l+1}$ Update the nodal values, return to step 1.

2.2 Preconditioning

Linear systems of equations for the velocity **u**,pressure correction p' and temperature T are given as:

$$A_{\mathbf{u}} \mathbf{x}_{\mathbf{u}} = \mathbf{b}_{\mathbf{u}} \quad (17)$$
$$A_{p'} \mathbf{x}_{p'} = \mathbf{b}_{p'} \quad (18)$$
$$A_T \mathbf{x}_T = \mathbf{b}_T \quad (19)$$

The velocity matrix $A_{\mathbf{u}}$ forms a block-diagonal matrix as shown below:

$$\begin{bmatrix} A_u & 0 & 0 \\ 0 & A_v & 0 \\ 0 & 0 & A_w \end{bmatrix} \begin{Bmatrix} \mathbf{x}_u \\ \mathbf{x}_v \\ \mathbf{x}_w \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_u \\ \mathbf{b}_v \\ \mathbf{b}_w \end{Bmatrix} \tag{20}$$

Since submatrices A_u, A_v and A_w have identical entries, their eigenvalues are also identical. Fig. 2 shows the sparsity pattern of pressure, velocity and temperature matrices for a mesh size N=513256 cells.

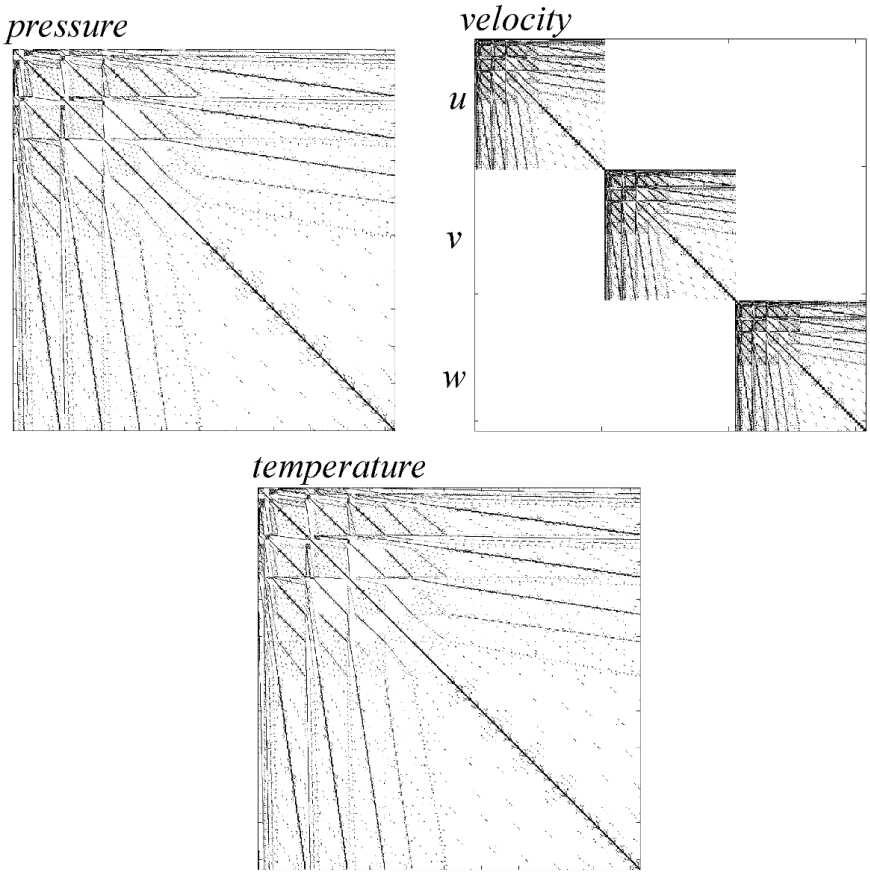

Figure 2. Pressure, velocity and temperature matrix structure for a mesh with N=513256 cells. The dark area shows the location of the non-zero entries, the rest being zero.

Preconditioning enhances the convergence properties and hence the ease of invertibility of a matrix. There are three types of preconditioning [17]:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b} \tag{21}$$

$$AM^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = M^{-1}\mathbf{y} \tag{22}$$

$$M_1^{-1}AM_2^{-1}\mathbf{y} = M_1^{-1}\mathbf{b}, \quad \mathbf{x} = M_2^{-1}\mathbf{y} \quad \text{and} \quad M = M_1M_2 \tag{23}$$

Equations (21-23) define left, right and split preconditioning steps, respectively. Here, M^{-1} is the preconditioner. Only when M^{-1} is explicitly known (example: Jacobi/Diagonal and SPAI) a direct

matrix-matrix multiplication $M^{-1}A$ or AM^{-1} possible, provided that the resultant product matrix does not become dense. In most cases M^{-1} is not explicitly known. Table 1 summarizes the preconditioners considered in the present work.

Table 1. List of preconditioners reported in the present work with the preconditioner shown in matrix form

Preconditioner	$\boldsymbol{M}$ (Eqs. 21-23)
Jacobi/Diagonal (D)	$\boldsymbol{D}$
Symmetric Gauss-Seidel (SGS)	$(D+U)D^{-1}(D+L)$
Incomplete LU (ILU(0))	$\tilde{L}_I\tilde{U}_I+E$
SPAI	$\|I-AM\|_F$

In Table 1 the first three preconditioners are obtained from splitting the matrix *A* into its lower *L*, upper *U* and diagonal *D* form where $A=L+D+U$. When a Jacobi preconditioner is used in Eq. (17), the matrix *A* becomes diagonally normalized (row scaled), i.e.

$$A=\begin{bmatrix} & & D^{-1}U \\ & 1 & \\ D^{-1}L & & \end{bmatrix} \tag{24}$$

The ILU class of preconditioners arises from the incomplete factorization of matrix *A* into its lower and upper triangular matrices. The complete LU factorization leads to *fill-ins* in the zero entries of the original matrix. When these fill-in entries are carefully dropped following a strategy then the factorization is termed incomplete LU factorization. In ILU(0) factorization all fill-in entries are dropped to retain the non-zero structure pattern of the original matrix. In M_{ILU}, additional computations are required to form approximate $\tilde{L}_I$,$\tilde{U}_I$ factors of *A*. The fourth preconditioner is the SPAI or the sparse approximate inverse preconditioner which also falls in the explicit category of preconditioners. The approximate inverse is found out by the Frobenius norm minimization [7]. One disadvantage of such preconditioners is that the setup time requirement i.e. the time required to compute the approximate inverse matrix, is comparatively higher as the matrix size increases. With the new formulation the setup time requirement for the pressure matrix preconditioner is negligible compared to the overall simulation time since it is computed only once at the beginning of the simulation. The modified matrix equation for pressure is given as:

$$M_{p'}^{-1}A_{p'}\mathbf{x}_{p'} = M_{p'}^{-1}D_{\mathbf{u}}\mathbf{b}_{p'} \tag{25}$$

In Eq. (25) $D_{\mathbf{u}}$ is a diagonal matrix which contains face values $\left(AP_{\mathbf{u}}\right)_f$ of the diagonal terms of the velocity matrix using the modified equation (10). $A_{p'}$ is dependent of the geometric and fluid properties which remain constant throughout the simulation. $M_{p'}^{-1}$ is found by Frobenius norm minimization of $\left\|I-A_{p'}M_{p'}\right\|_F^2=\sum_{j=1}^{n}\left\|e_j-A_{p'}m_{p'j}\right\|_2^2$. Every columns are minimized using a self-preconditioned minimal residual (MR) iteration [7]. Within every MR iteration it follows a numerical dropping strategy where only the largest value in the column is retained and the rest of the values are dropped. If the number of *fill-in* entries per column are designated by *k* then SPAI(k) indicates the *k* number of non-zero entries per column after *k* inner MR iteration.

It is well known that explicit preconditioners are highly parallelizable. From Table 1, diagonal and SPAI preconditioners are explicit and highly parallelizable but diagonal preconditioner has a poor convergence with any Krylov solver. Since pressure matrix is well known to be highly ill-conditioned it requires a robust as well as highly parallelizable preconditioners such as SPAI. In the subsequent sections three problems are carried out with all the listed preconditioners applied on the pressure, velocity and temperature matrix systems.

3 Results and discussion

Numerical simulations have been carried out on three different geometries. These include mixed convection inside a 3D double sided lid-driven cavity, unsteady flow past a circular cylinder and pulsatile flow inside an asymmetric abdominal aortic aneurysm. Convergence characteristics of pressure, velocity and temperature matrices with the listed preconditioners in Table 1 decide the best combination of preconditioners which should be chosen to get maximum speed up with respect to the overall simulation time. An in-house unstructured FVM based code written in C++ is used for the simulation. A parallel preconditioned iterative Krylov solver (BiCGSTAB) is used as accelerator for the linear equations. The matrix solver is a general purpose sparse matrix solver and the storage is based on compressed sparse row format. Under relaxation factors are chosen to provide stability and acceleration to the iterative solution.

3.1 Mixed convection in 3D double sided lid driven cavity

In a cubical cavity the top and bottom surface are given a uniform velocity U_0. The top surface is maintained at higher temperature T_{HOT}=350K and the bottom surface is maintained at a lower temperature T_{COLD}=300K. The lateral sides are insulated as shown in Fig. 3. Working fluid considered is air with *Pr* =0.71. Steady state results are analysed and a pseudo transient approach is adopted. A good quality mesh with a total of 513256 tetrahedral mesh cells were used after separately conducting the grid independence.

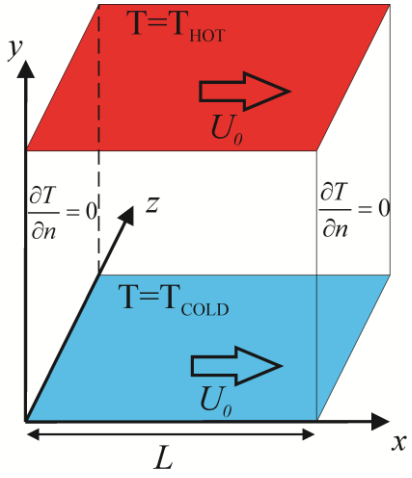

Figure 3: Double sided lid driven cubical cavity

Convection patterns for a range of Reynolds number (*Re*=100, 400 and 1000) and Richardson number (*Ri*=0.001, 1, 10) are studied. A Boussinesq approximation is assumed. Fig. 4 shows the temperature iso-contours at various Reynolds number and Richardson numbers. A low *Ri* signifies a forced convection and a high *Ri* signifies more contribution due to the buoyancy effect. Table 3 shows that the

average Nusselt number (Nu_{avg}) calculated for the top hot surface shows a good match with Ouertatani [18]. Nu_{avg} is calculated as shown below:

$$Nu_{avg} = -\frac{1}{Area} \times \frac{1}{(T_{HOT} - T_{COLD})} \iint \frac{\partial (T - T_{COLD})}{\partial y} dxdy$$

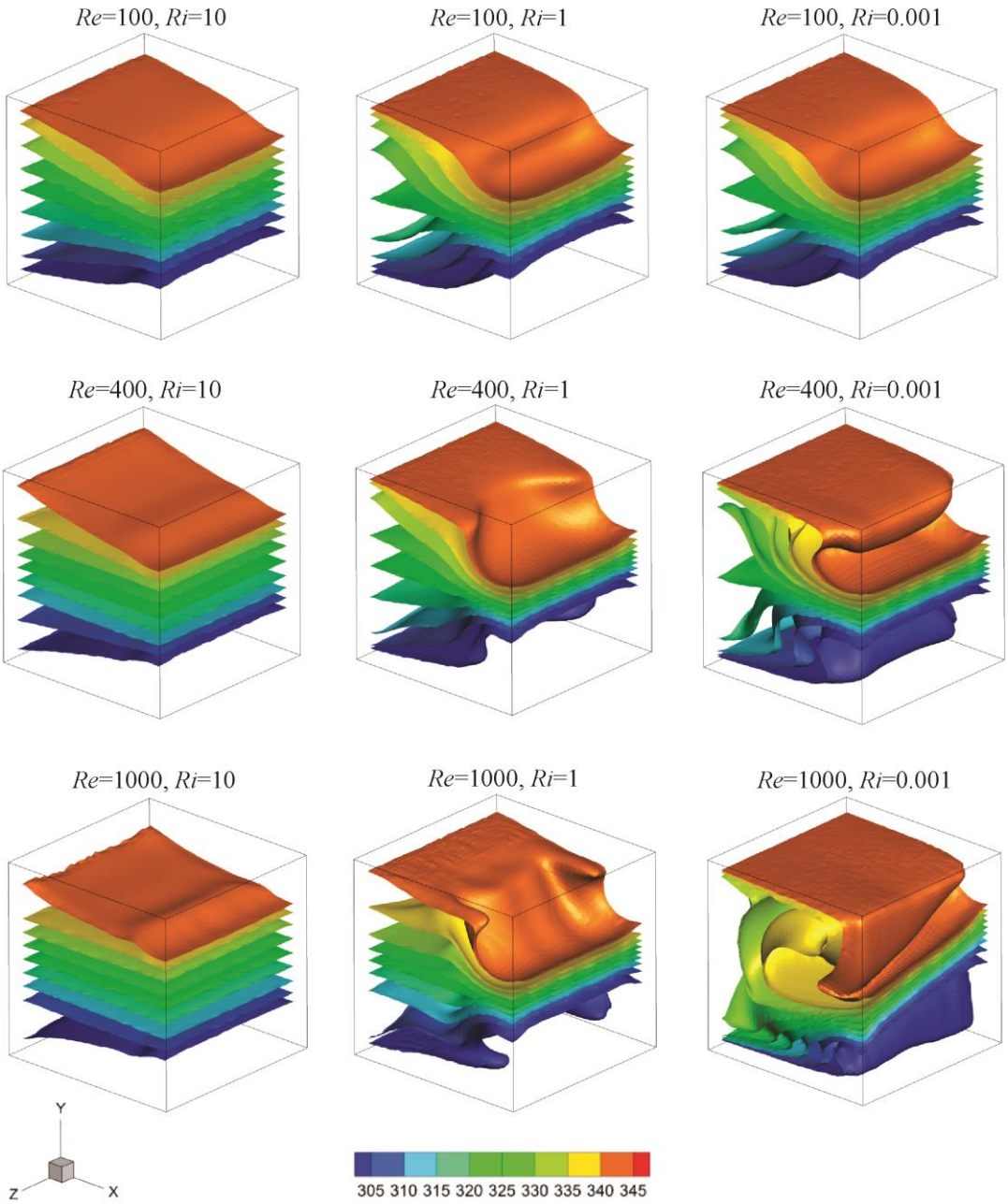

Figure 4: Temperature iso conoturs at various Reynolds numbers and Richardson numbers

Figure 5 shows the convergence characteristics of various preconditioners on the velocity and temperature matrices at *Re* considered for the study. Since variation in *Ri* does not alter the coefficients of both velocity and temperature matrices the convergence characteristic with respect to *Ri* is insignificant. Delay in convergence with increasing *Re* is observed for both velocity and temperature matrices. ILU(0) shows the best convergence rate for all *Re* considered, although SGS shows a closer match with that of ILU(0).

Table 2. Average Nusselt number of the top surface

	Ri=0.001		*Ri*=1		*Ri*=10	
	Present	Ref[18]	Present	Ref[18]	Present	Ref[18]
Re=100	1.79	1.83	1.38	1.35	1.09	1.09
Re=400	3.99	3.96	1.51	1.53	1.14	1.13
Re=1000	7.32	7.28	1.83	1.86	1.15	1.14

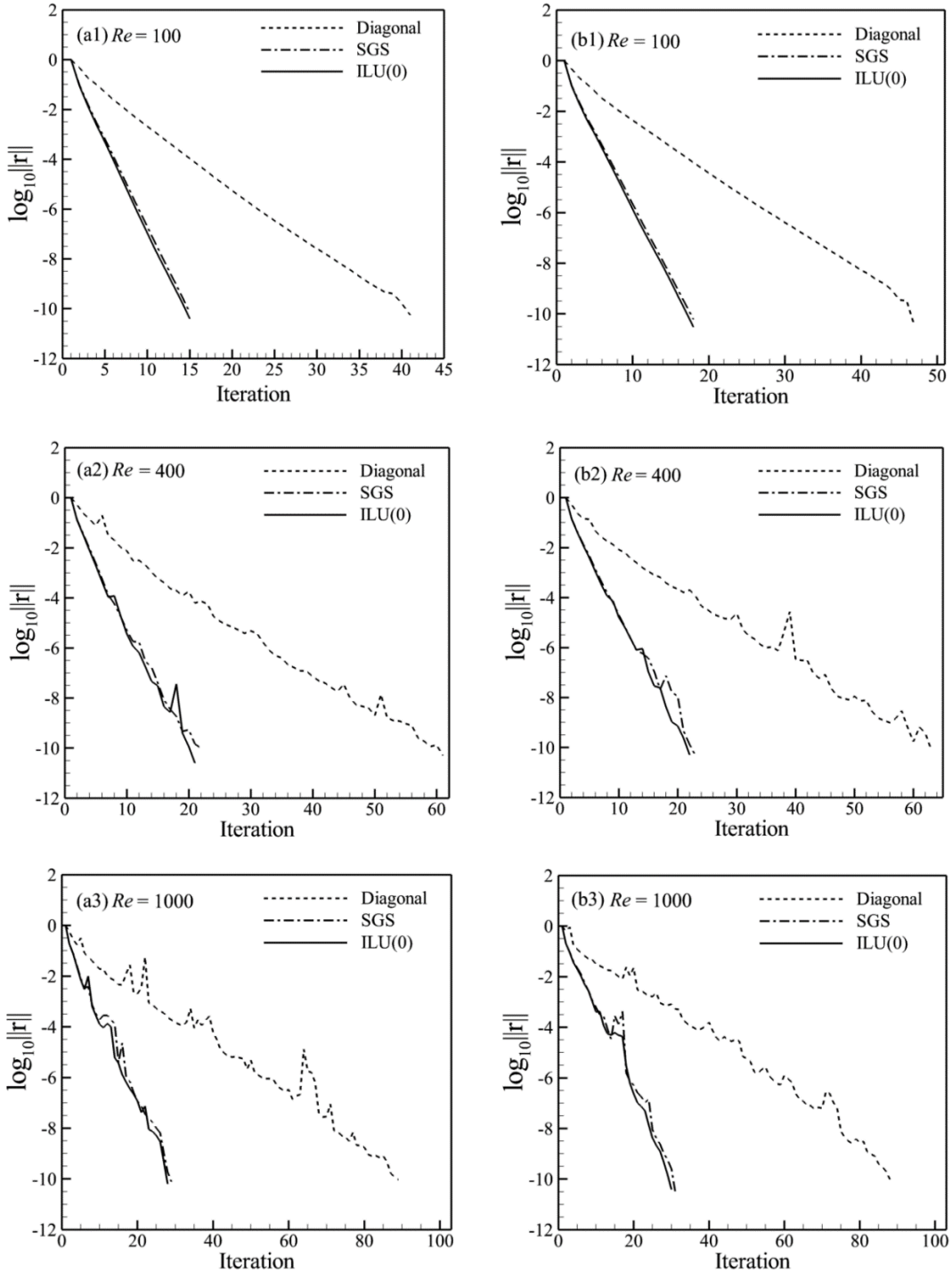

Figure 5: Convergence characteristics for various preconditioners with Reynolds number for velocity (a1, a2 and a3) and temperature (b1, b2 and b3) matrices.

Figure 6 shows the CPU time taken in seconds for the velocity and temperature matrices at various Reynolds numbers with 20 CPU cores. Fig. 6 shows the opposite of what is expected from Fig. 5. Diagonal preconditioner shows the best performance in terms of CPU time as it takes the least CPU time per linear solve when compared to SGS and ILU(0) preconditioner. This shows that the adverse effect due to poor convergence rate is overcome due to the high parallelizability of diagonal preconditioner. Since the velocity and temperature matrices remain well conditioned the CPU time taken per linear solve is insignificant compared to that of pressure matrices. Hence for velocity and temperature matrices diagonal preconditioner is used.

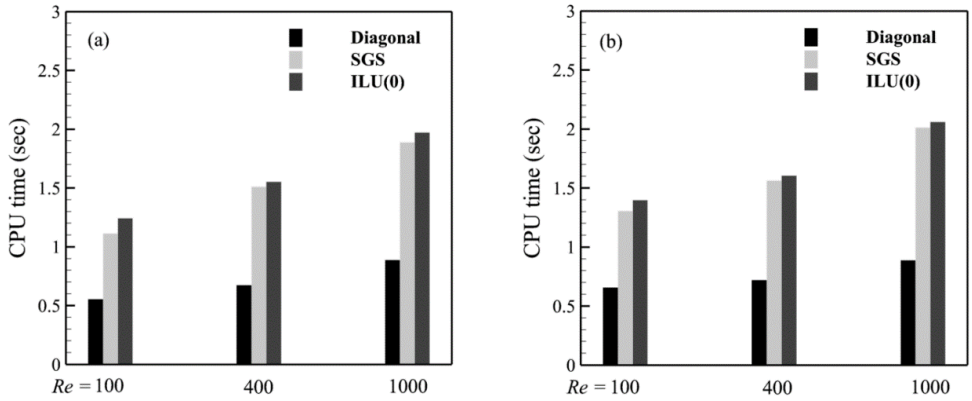

Figure 6: CPU time taken in seconds per linear solve for (a) velocity and (b) temperature matrices at various Reynolds numbers with 20 CPU cores

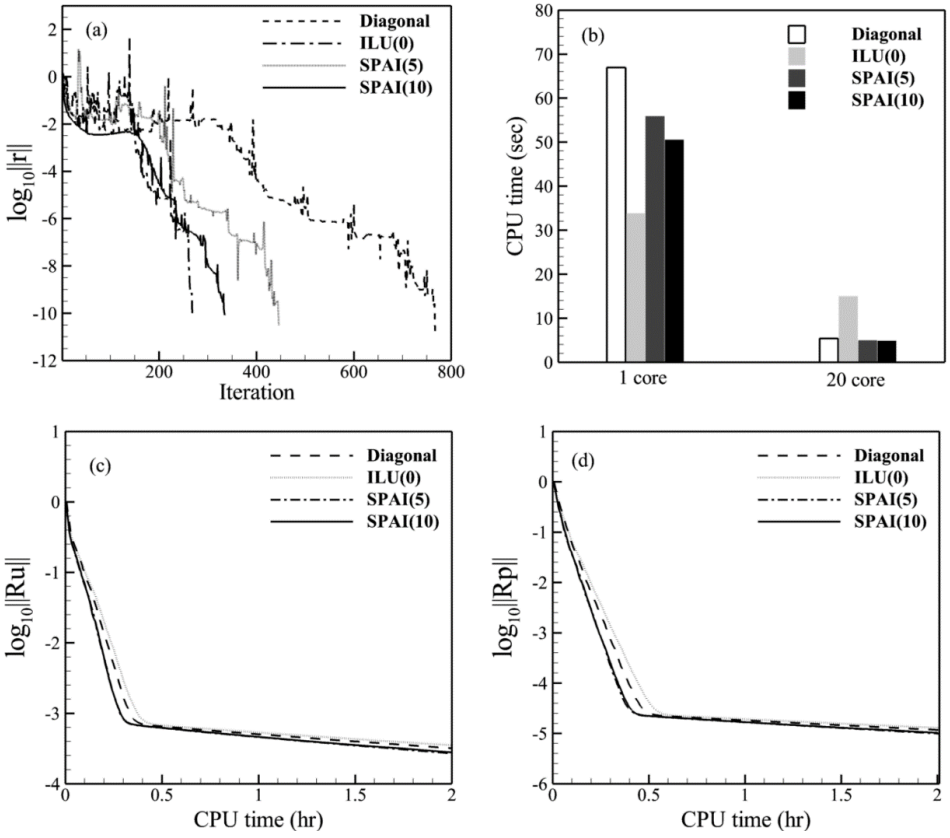

Figure 7: (a) Convergence characteristics and (b) CPU time taken per linear solve for 1 and 20 CPU cores for pressure matrix. (c) Overall non-linear momentum residual convergence for one non-linear SIMPLE iteration and (d) overall mass residual convergence for one non-linear SIMPLE iteration for 20 CPU cores

Figure 7(a) shows the residual convergence with various preconditioners for the pressure matrix for one linear solve. ILU(0) shows the best convergence rate. SPAI preconditioners with 10 *fill-in* (SPAI(10)) per column shows a better convergence than 5 *fill-in* (SPAI(5)) per column. This suggests that as the level of *fill-in* increase per column a better convergence rate is achieved. However, this increases the computational cost of setting up the approximate inverse preconditioner. Fig 7(b) shows the CPU time taken for 1 core and corresponding time taken for 20 core processors. SPAI(5) and SPAI(10) show the best parallel efficiency with almost a linear scale up when compared to other preconditioners. There is a significant difference of around 4 seconds CPU time between SPAI and diagonal preconditioner. It is evident from Fig. 7(b) that the performance of ILU(0) preconditioner gets deteriorated when applied over muti-core processors. This is evident from Figs. 7(c) and 7(d) which shows the residual convergence of momentum and mass for one non-linear SIMPLE iteration which is equivalent to the time taken per time step. The difference in time is significant since several time steps need to be progressed to reach the desired solution.

3.2 Laminar flow past a circular cylinder

Unsteady flow past a circular cylinder is studied at *Re* = 100. The fluid is assumed to be isothermal (Γ=0). Reynolds number is based on the cylinder diameter (D) which is chosen as the length scale. The length of the domain is 20 cylinder diameter (20D). The width and height of the domain are 10D each respectively. The circular cylinder is placed at a distance of 5D from the inflow plane. A uniform velocity U_0, is prescribed at the inlet boundary. Neumann boundary condition is prescribed for velocity at the outflow plane. The lateral walls are given a free slip boundary condition for velocity. For pressure, Neumann boundary condition is prescribed at the walls and the inflow plane while a Dirichlet boundary condition is prescribed at the outlet. A high quality mesh with a total of 1.34×10^6 tetrahedral cells were used. Fig. 8(a) shows vortex shedding behind the circular cylinder at the prescribed Reynolds number. Flow periodicity is a salient feature at this Reynolds number and this is brought in Fig. 8(b) which shows the variation of transverse velocity with time probed at a distance of 1D downstream the cylinder. The vortex shedding corresponds to a single frequency and the non-dimensional frequency or the Strouhal number was found to be *St* = 0.156. This number matches well with *St*=0.154 reported by Zhang and Dalton [19] at *Re*=100.

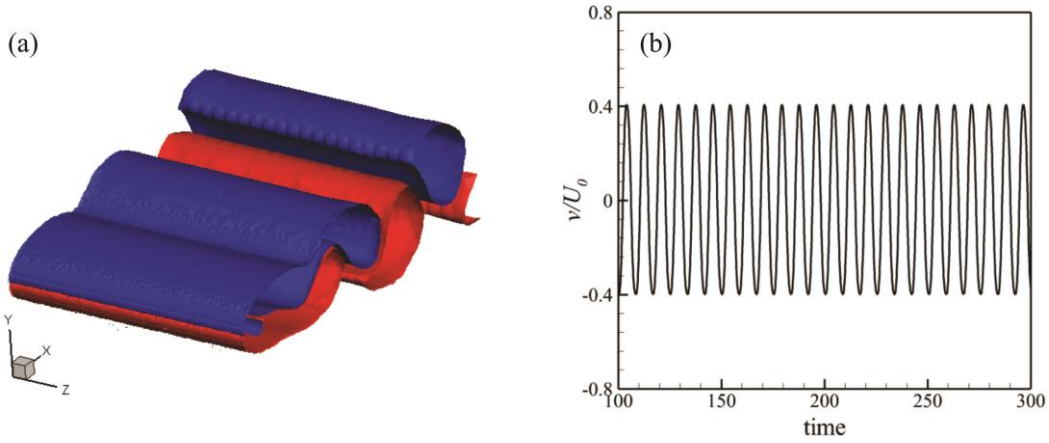

Figure 8: Flow past a circular cylinder at *Re*=100: (a) vorticity iso-contours $\omega_z = \pm 0.15$ (b) transverse velocity (v-velocity) signal variation with dimensionless time

Figure 9(a) shows a similar convergence characteristics as for lid driven cavity problem in Fig. 7(a). ILU(0) still shows a better performance in terms of number of iterations taken to converge below a set tolerance value. Five *fill-in* per column SPAI(5) is used here since it was shown in the previous lid driven cavity problem that the CPU time taken was almost the same for 5 and 10 fill-in entries per column. The number of entries for ILU(0) and SPAI(5) match as ILU(0) mimics the parent matrix which has five entries per row since tetrahedral mesh was used. Hence 5 *fill-in* entries for SPAI makes it ideal for comparing it with ILU(0). Fig. 9(b) shows that the parallel efficiency and scalability of SPAI preconditioner is far superior compared to other preconditioners. While for single core CPU ILU(0) shows superior performance, due to its weak parallelizability both SGS and ILU(0) show poor performance in multi core CPUs. Figs. 10(a) and 10(b) shows the momentum and mass residual convergence for 20 CPU cores. As seen from the previous results when SPAI preconditioner is used for the pressure matrix it is seen that there is a massive improvement in the mass and momentum residual convergence rate per non-linear SIMPLE iteration.

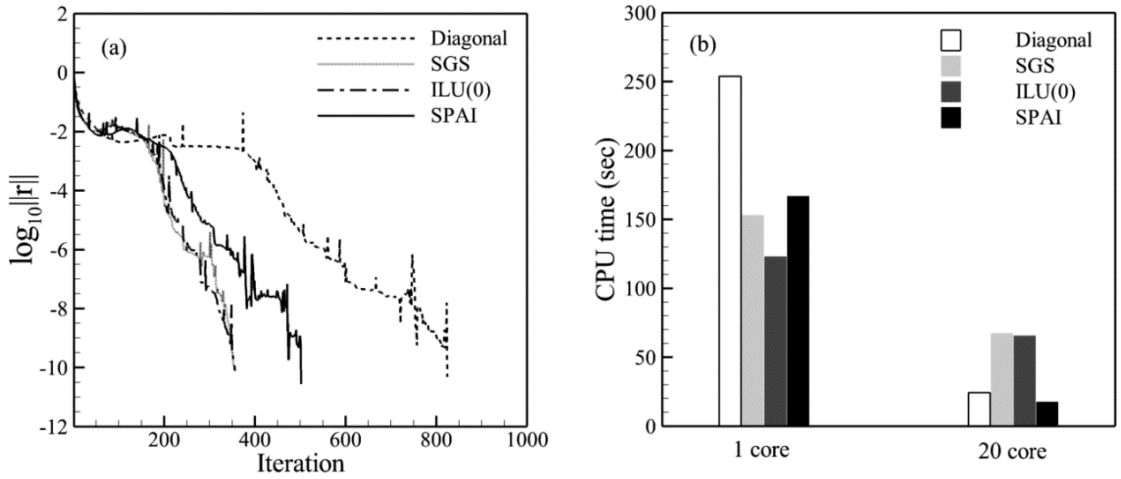

Figure 9: Pressure matrix for flow past a circular cylinder: (a) Convergence characteristics with various preconditioners per linear solve (b) CPU time for 1 and 20 cores per linear solve.

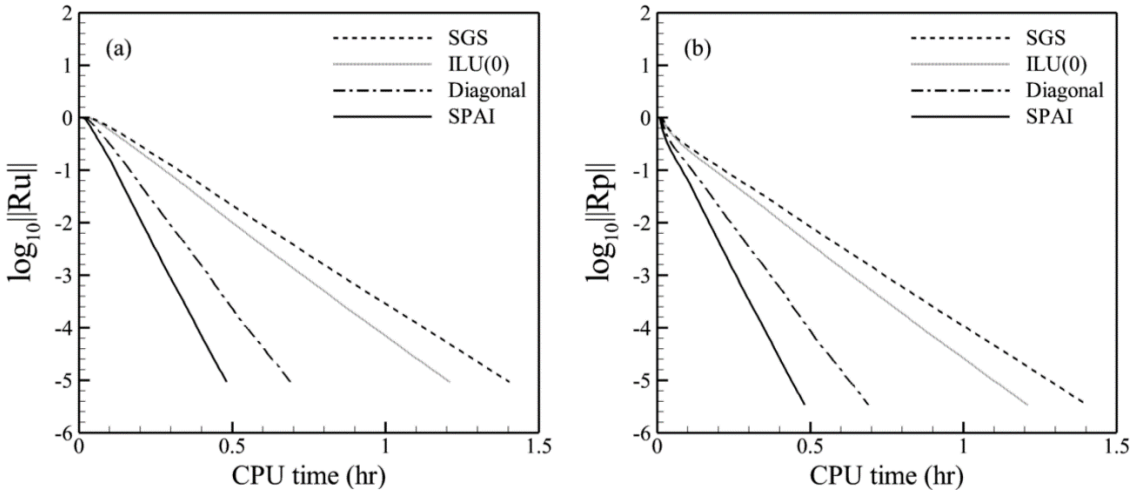

Figure 10: (a) Momentum residual and (b) mass residual convergence with CPU time for various preconditioners. The results are shown for 20 CPU cores for flow past a circular cylinder.

3.3 Pulsatile flow inside an asymmetric abdominal aortic aneurysm (AAA).

Figure 11 shows the layout of an asymmetric abdominal aortic aneurysm. The part resembles a model of an AAA [20]. The model considered is rigid. The working fluid considered has properties of human blood. Density (ρ) of the fluid is 1130 kg/m^3 and viscosity (μ) is 0.004 Pa-s. The flow is assumed to be isothermal (Γ=0). The tube diameter is 2.16 cm. The flow is considered pulsatile in nature mimicking the physiological nature of human blood inside abdominal aorta. Fig. 12 shows the inlet and outlet velocity and pressure waveforms extracted from the data provided in [20]. The extracted waveforms chosen for the simulation correspond to resting human condition. The peak *Re* corresponding to the peak flow rate is found to be *Re* = 1708. The Womersley number (*Wo*) corresponding to the cycle time of 0.85sec was found to be *Wo* = 15.65.

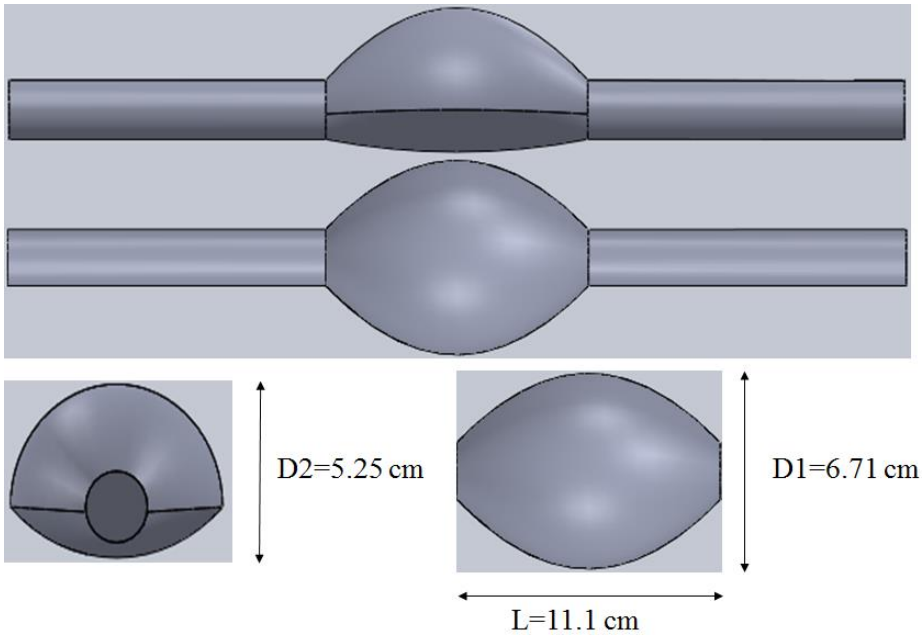

Figure 11: Layout of asymmetric abdominal aortic aneurysm

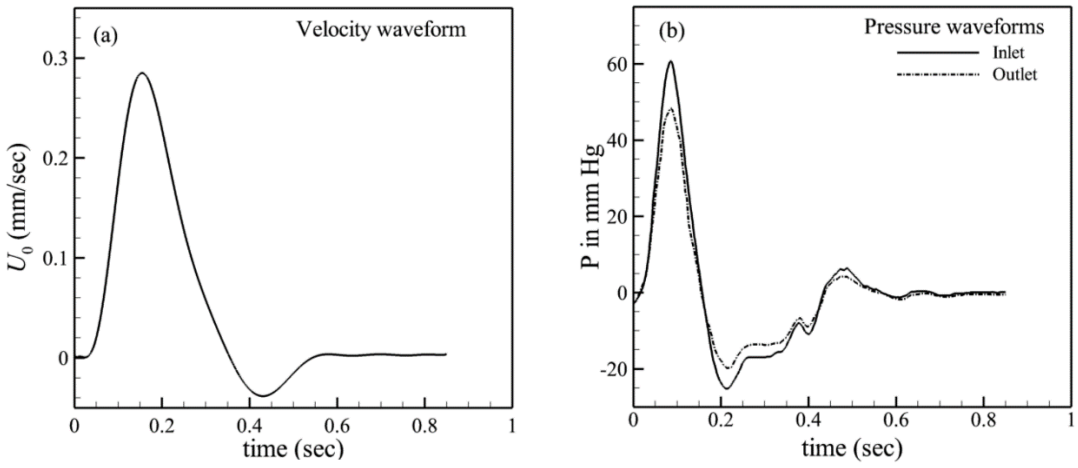

Figure 12: (a) Inlet velocity waveform and (b) inlet and outlet pressure waveforms

Figure 13 show the swirling strength of the flow inside the AAA portion corresponding to the cycle time of 0.3sec. The present numerical results shows a reasonably good match with the experimental results of Deplano et al. [20]. The swirling strength indicates the location of vortices. The vortices are formed in the initial phase and it grows larger in size during the deceleration phase. These vortices remain attached near the inlet section of AAA as shown in Fig. 13 during the positive cycle. When the flow reverses, these vortices disappear from the inlet section and no such vortices are formed till the end of the cycle. The formation of vortices inside an aneurysm is important to study since the local vortices generated during the pulsatile cycle promotes local shearing which may cause tearing away of inner endothelial wall causing rupture of aneurysm which could be fatal.

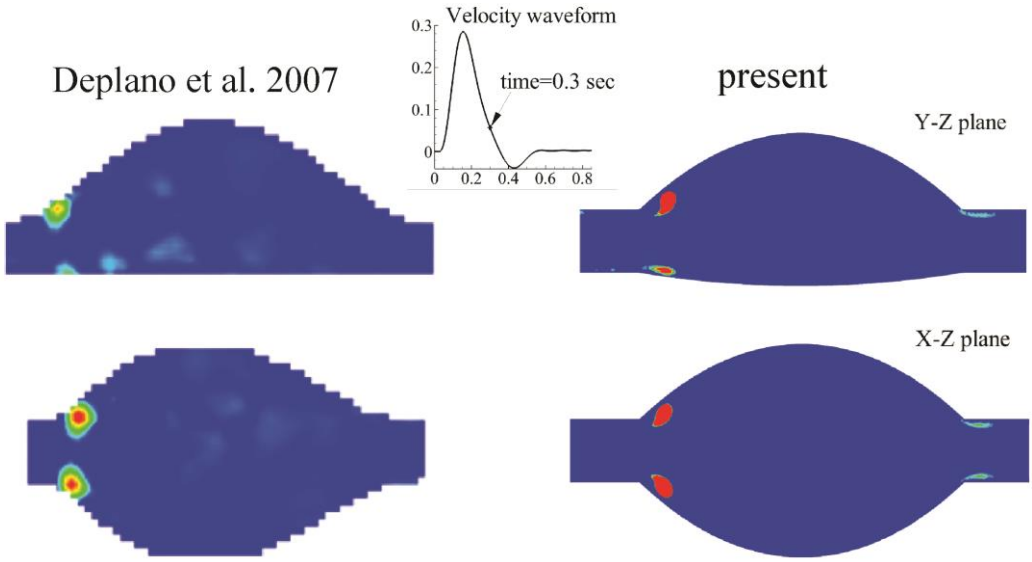

Figure 13: Swirling strength inside the asymmetric bulge corresponding to time = 0.3 sec of cycle time. Right hand side shows the present numerical results and left hand side shows the experimental results of Deplano et al. [20].

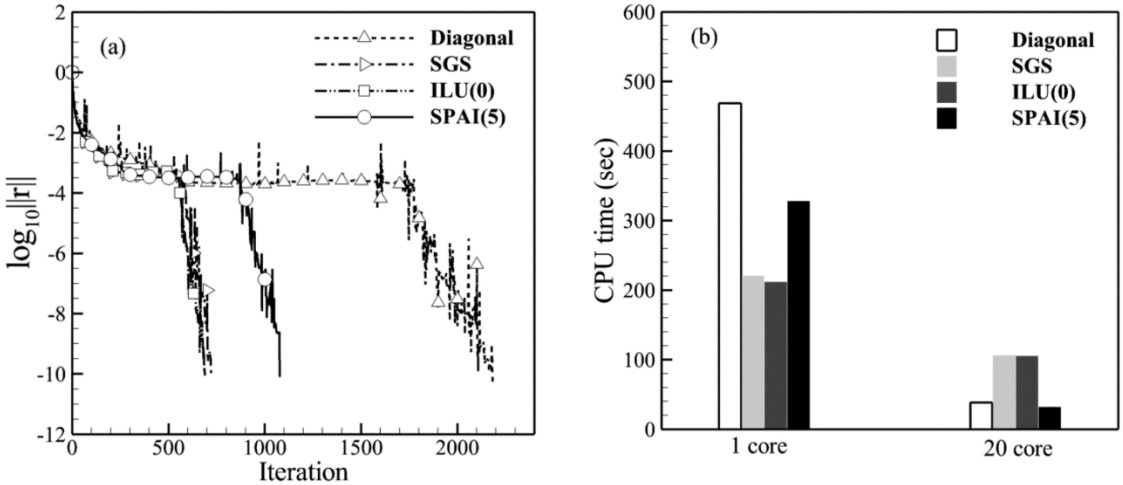

Figure 14: Pressure matrix for asymmetric abdominal aortic aneurysm: (a) Convergence characteristics with various preconditioners per linear solve (b) CPU time for 1 and 20 cores per linear solve.

Figure 14 shows the pressure matrix convergence per linear solve with various preconditioners. For single core CPU ILU(0) preconditioner shows the best convergence characteristics but the CPU time taken is the worst for ILU(0) precoditioner for multi core (20) CPUs. From Fig. 14 (b) it is evident that SPAI(5) shows the best speedup per linear solve. Diagonal preconditioner shows the worst convergence characteristics but due to its high parallelizability it shows a good speed up with multi core CPUs as shown in Fig. 14(b). Fig. 15 (a) and (b) shows the non-linear momentum and mass residual convergence per non-linear SIMPLE iteration for 20 CPU cores. The flow being pulsatile in nature, during the initial cycles a larger magnitude of tolerance limit is imposed for convergence. For the results shown these tolerance limits are given as 10^{-6}. Once the flow develops the tolerance limits are further reduced. This strategy saves computational time. SPAI(5) preconditioner is seen to give the best speedup with 20 CPU cores compared to other preconditioners. Diagonal preconditioner shows high fluctuations in its residual convergence whereas rest of the preconditioners show a smooth, almost monotonic convergence which is desirable especially during the initial stages of simulation. High fluctuations in the residual convergence may result in numerical overshoots in the solution which is undesirable. Hence diagonal preconditioner although it gives faster convergence than ILU(0) and SGS, it may give undesirable results if the convergence is not smoothed out.

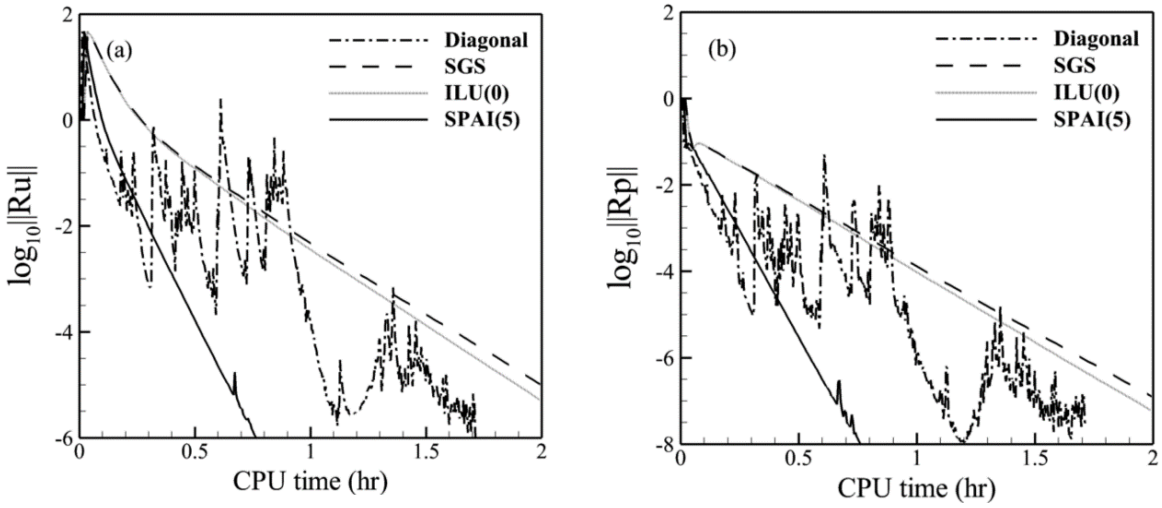

Figure 15: (a) Momentum residual and (b) mass residual convergence with CPU time for various preconditioners. The results are shown for 20 CPU cores for flow inside asymmetric abdominal aortic aneurysm.

4 Conclusions

Parallel preconditioners for the pressure-velocity-temperature matrix systems are analysed on a SIMPLE variant of algorithm applied for incompressible flows. SIMPLE and its variants are well known for its strong pressure-velocity coupling which makes the coefficients of the pressure matrix change with the changing velocity field. This makes it difficult or time consuming to find its approximate that are used as highly parallelizable preconditioners in iterative solvers such as BiCGSTAB. The interpolation of the central coefficient of the discretized velocity term, which occurs with the pressure gradient term in the pressure correction equation, with the neighbouring cell centroid value is what makes the pressure matrix vary with the changing velocity field. The remedy to this problem is to make the pressure matrix coefficients constant such that the approximate inverse need to be computed only once at the beginning of computations. Based on the new formulation which assist

parallel preconditioners, simulations are carried out for three problems and the following conclusions are arrived:

1. The new approach uses a higher order interpolation of the central coefficient of the velocity term $AP_{\mathbf{u}}$ on the faces of the control volume using Taylor series expansion. Weighted least squares gradient method is used to find the gradient $\nabla\left(AP_{\mathbf{u}}\right)$ at the cell centroid. In order to make $AP_{\mathbf{u}}$ on the faces constant, the distance vector $\Delta\mathbf{r}_f$ is made constant by taking the harmonic average of all the distance vectors from the cell centroid to the face centroid within a control volume. Hence, the averaged vector $\Delta\mathbf{r}_f$ remains closer to the smallest of all the distance vectors within the control volume. This ensures stability to the solution by not over estimating the interpolated value the face. The present results shown are for the 2nd order interpolated values of $AP_{\mathbf{u}}$ at the face but are not restricted to the same. 3rd order interpolation could be achieved by estimating $\nabla^2\left(AP_{\mathbf{u}}\right)$ at the cell centroid using the same weighted least squares gradient approach. The new approach has been shown to be stable and is highly suitable for explicit type parallel preconditioners.
2. *Velocity and temperature matrices*: Three problems were studied which include i) mixed convection inside a double sided 3D lid driven cavity, ii) unsteady flow past a circular cylinder at *Re*=100 and iii) pulsatile flow inside an asymmetric abdominal aortic aneurysm. For all the problems considered the velocity and temperature (lid driven cavity) matrices remained well conditioned. This is due the parabolic nature of the governing partial differential equations. ILU(0) preconditioner showed the best convergence characteristics but due to its weak parallelizability it showed poor performance with multi core CPUs. Diagonal preconditioner was found to give superior performance as it is computationally cheaper and highly parallelizable when compared to ILU(0) and SGS for multi core CPUs.
3. *Pressure matrix*: For all the problems considered for the study the pressure matrix was found to be highly ill-conditioned and requires robust preconditioners. Although ILU(0) again showed the best convergence characteristics, it showed extremely poor scalability with multi core CPUs. SPAI based preconditioner is seen to show the best performance with multiple processors for all the problems.
4. For the overall nonlinear momentum residual and mass residual convergence, a combination of diagonal preconditioner for velocity and SPAI for pressure showed the best speedup in multi core CPUs. Although diagonal preconditioners are scalable it showed fluctuating convergence of momentum and mass residual for pulsatile flow inside abdominal aortic aneurysm problem. This is highly undesirable since such fluctuations can cause numerical overshoots which may give unphysical solutions. Hence for pressure matrix robust preconditioners which are also highly parallelizable such as SPAI are required to produce the desired effect.

Nomenclature

$A_{p'}$	pressure correction matrix
$A_{\mathbf{u}}$	block diagonal velocity matrix
$AP_{\mathbf{u}}$	central coefficient of velocity matrix
M^{-1}	preconditioner
p	pressure (dimensionless)
p'	total pressure correction
p'_m	calculated pressure correction
p'_s	smoothing pressure correction
$\mathbf{R_u}$, $\mathbf{R}_p$	momentum and mass residual vector
$\mathbf{u}$	Cartesian velocity vector (dimensionless)
T	unknown temperature
Greek symbols	
α	under relaxation factor for velocity
β	under relaxation factor for pressure correction
Γ	= 0 for isothermal flows, = 1 for non-isothermal flows
Abbreviations	
SPAI	sparse approximate inverse preconditioner

Acknowledgement

All computations for the present work were carried out at the High Performance Computing (HPC 2013) facility of Indian Institute of Technology Kanpur, India.

References

[1] Patankar S. *Numerical heat transfer and fluid flow*. CRC press. 1980.

[2] Date AW. Solution of transport equations on unstructured meshes with cell-centered colocated variables. Part I: Discretization. International Journal of Heat and Mass Transfer, 48(6):1117-27, 2005.

[3] Van der Vorst HA. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 13(2):631-44, 1992.

[4] Saad Y, Schultz MH. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7(3):856-69, 1986.

[5] Saad Y. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics. 2003.

[6] Grote MJ, Huckle T. Parallel preconditioning with sparse approximate inverses. SIAM Journal on Scientific Computing, 18(3):838-53, 1997.

[7] Chow E, Saad Y. Approximate inverse preconditioners via sparse-sparse iterations. SIAM Journal on Scientific Computing, 19(3):995-1023, 1998.

[8] Benzi M, Tuma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. SIAM Journal on Scientific Computing, 19(3):968-94, 1998.

[9] Geveler M, Ribbrock D, Göddeke D, Zajac P, Turek S. Towards a complete FEM-based simulation toolkit on GPUs: Unstructured grid finite element geometric multigrid solvers with strong smoothers based on sparse approximate inverses. Computers and Fluids, 80:327-32, 2013.
[10] Akay HU, Oktay E, Manguoglu M, Sivas AA. Improved parallel preconditioners for multidisciplinary topology optimisations. International Journal of Computational Fluid Dynamics, 30(4):329-36, 2016.
[11] Löhner R, Mut F, Cebral JR, Aubry R, Houzeaux G. Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: Extensions and improvements. International Journal for Numerical Methods in Engineering, 87(1-5):2-14, 2011.
[12] Hsu HW, Hwang FN, Wei ZH, Lai SH, Lin CA. A parallel multilevel preconditioned iterative pressure Poisson solver for the large-eddy simulation of turbulent flow inside a duct. Computers and Fluids, 45(1):138-46, 2011.
[13] McAdams A, Sifakis E, Teran J. A parallel multigrid Poisson solver for fluids simulation on large grids. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 65-74, 2010.
[14] Segal A, ur Rehman M, Vuik C. Preconditioners for incompressible Navier–Stokes solvers. Numerical Mathematics: Theory, Methods and Applications, 3(3):245-75, 2010.
[15] Gohil T, McGregor RH, Szczerba D, Burckhardt K, Muralidhar K, Székely G. Simulation of oscillatory flow in an aortic bifurcation using FVM and FEM: A comparative study of implementation strategies. International Journal for Numerical Methods in Fluids, 66(8):1037-67, 2011.
[16] Sozer E, Brehm C, Kiris CC. Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered cfd solvers. 52nd Aerospace Sciences Meeting AIAA 2014-1440, 2014. https://doi.org/10.2514/6.2014-1440
[17] Benzi M. Preconditioning techniques for large linear systems: a survey. Journal of Computational Physics, 182(2):418-77, 2002.
[18] Ouertatani N, Cheikh NB, Beya BB, Lili T, Campo A. Mixed convection in a double lid-driven cubic cavity. International Journal of Thermal Sciences, 48(7):1265-72, 2009
[19] Zhang J, Dalton C. A three-dimensional simulation of a steady approach flow past a circular cylinder at low Reynolds number. International Journal for Numerical Methods in Fluids, 26(9):1003-22, 1998.
[20] Deplano V, Knapp Y, Bertrand E, Gaillard E. Flow behaviour in an asymmetric compliant experimental model for abdominal aortic aneurysm. Journal of biomechanics, 40(11):2406-13, 2007.