A block preconditioner for scalable large scale finite element incompressible flow simulations*

Damian Goik and Krzysztof Banaś[1][0000−0002−4045−1530]

AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Kraków, Poland
pobanas@cyf-kr.edu.pl

Abstract. We present a block preconditioner, based on the algebraic multigrid method, for solving systems of linear equations, that arise in incompressible flow simulations performed by the stabilized finite element method. We select a set of adjustable parameters for the preconditioner and show how to tune the parameters in order to obtain fast convergence of the standard GMRES solver in which the preconditioner is employed. Additionally, we show some details of the parallel implementation of the preconditioner and the achieved scalability of the solver in large scale parallel incompressible flow simulations.

Keywords: finite element method · Navier-Stokes equations · solvers of linear equations · block preconditioning · algebraic multigrid

1 Introduction

Stabilized finite elements are one of the popular techniques for solving Navier-Stokes equations of incompressible flows [5]. We are interested in the strategy, in which the finite element method is applied for space discretization, with some form of implicit discretization in time, either for transient problems or for pseudo-transient continuation employed to achieve steady-state [11]. The resulting non-linear systems are usually solved either by some form of Newton iterations or Picard, fixed-point, iterations [7] – we select the latter technique in our numerical examples.

In each of the considered scenarios there is a sequence of systems of linear equations to be solved. Due to the incompressibility condition, in the form of the requirement for divergence free velocity field, the systems are ill conditioned and the standard Krylov subspace methods with typical preconditioners (like ILU(k) – incomplete factorization algorithms) become inefficient [16, 18].

We aim at developing a solver for large scale parallel incompressible flow simulations. Therefore we renounce the direct solvers, due to their super-linear complexity and poor parallel scalability [14, 12]. We try to find a scalable and sufficiently strong preconditioner, in order to guarantee the convergence of the standard restarted GMRES method [17].

* The work was realized as a part of fundamental research financed by the Ministry of Science and Higher Education, grant no. 16.16.110.663

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

We consider block preconditioners [15, 16] that split the linear systems into two parts: the first related to velocity components, with time derivative terms and better convergence properties, and the second related to the pressure, i.e. the incompressibility condition.

For the latter part we use an algorithm based on the algebraic multigrid, as the black-box version of the multigrid method [19], the only method that properly takes into account the infinite speed of propagation of pressure changes throughout the domain [7].

Several versions of algebraic multigrid (AMG) have been proposed for dealing with the pressure related part of the system that arise in block preconditioners for the Navier-Stokes equations [8]. The extensive studies in [6] showed the strong deterioration of convergence properties of the solvers for the increasing Reynolds and CFL numbers in the case of using smoothed aggregation multigrid. In the recent article [20] classical AMG is compared with smooth aggregation AMG for a large scale transient problem.

In the current paper we propose a block preconditioner based on the algorithm outlined in [18], combined with the classical AMG algorithm by Stuben [19]. We investigate the optimization options available for the standard AMG and show how to obtain, based on the proper selection of techniques and parameters, good convergence properties together with the scalability for the whole solver in large scale stationary incompressible flow simulations.

2 Problem statement

We solve the Navier-Stokes equations of incompressible fluid flow, formulated for the unknown fluid velocity $\boldsymbol{u}(\boldsymbol{x}, t)$ and pressure $p(\boldsymbol{x}, t)$ that satisfy:

$$\begin{aligned} \rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \nu\boldsymbol{\nabla}^2\boldsymbol{u}\right) + \boldsymbol{\nabla} p &= \boldsymbol{f} \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 \end{aligned} \tag{1}$$

together with boundary conditions:

$$\begin{aligned} \boldsymbol{u} &= \hat{\boldsymbol{u}}_0 \quad \text{on } \Gamma_D \\ (\nu\boldsymbol{\nabla}\boldsymbol{u})\boldsymbol{n} - p\boldsymbol{n} &= \boldsymbol{g} \quad \text{on } \Gamma_N \end{aligned}$$

where ν and ρ denote kinematic viscosity and density of fluid respectively, and $\boldsymbol{f}$ is a source term that includes gravity forces (the system is considered in the dimensional form). The vector fields $\hat{\boldsymbol{u}}_0$ and $\boldsymbol{g}$ are given on two disjoint parts of the boundary of the 3D computational domain Ω, Γ_D for velocities and Γ_N for stresses, respectively.

We discretize the Navier-Stokes equations using the spaces of continuous, piecewise linear polynomials. For velocity and pressure unknowns we consider the spaces $V^h_{\boldsymbol{u}}$ and V^h_p (vector valued for velocities), with functions that satisfy Dirichlet boundary conditions, while for test functions we apply the spaces $V^h_{\boldsymbol{w}}$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

and V_r^h, with zero values on the Dirichlet parts of the boundary. We use SUPG stabilized finite element formulation [9] for space discretization that can be written (in index notation with the summation convention for repeated indices):

Find approximate functions $\boldsymbol{u}^h \in V_{\boldsymbol{u}}^h$ and $p^h \in V_p^h$ such that the following statement:

$$\int_\Omega \rho \frac{\partial u_j^h}{\partial t} w_j^h d\Omega + \int_\Omega \rho u_{j,l}^h u_l^h w_j^h d\Omega + \int_\Omega \rho \nu u_{j,l}^h w_{j,l}^h d\Omega - \int_\Omega p^h w_{j,j}^h d\Omega$$
$$- \int_\Omega u_{j,j}^h r^h d\Omega + \sum_e \int_{\Omega_e} R_j^{NS}(\boldsymbol{u}^h, p^h) \tau_{jl} R_l^{NS}(\boldsymbol{w}^h, r^h) d\Omega$$
$$+ \sum_e \int_{\Omega_e} u_{j,l}^h \delta w_{j,l}^h d\Omega = \int_\Omega f_j w_j^h d\Omega - \int_{\Gamma_N} g_j w_j^h d\Gamma$$

holds for every test function $\boldsymbol{w}^h \in V_{\boldsymbol{w}}^h$ and $r^h \in V_r^h$.

Above, $R_j^{NS}(\boldsymbol{u}^h, p^h)$ and $R_l^{NS}(\boldsymbol{w}^h, r^h)$ denote residuals of the Navier-Stokes equations computed for respective arguments, while τ_{jl} and δ are coefficients of SUPG stabilization [9].

Since in the current paper we are mainly interested in stationary problems and pseudo-transient continuation technique, we use the implicit Euler time discretization, due to its stability. When applied to (2), it leads to a non-linear problem for each time step. We use Picard's (simple) iterations for solving non-linear problems that finally lead to a series of linear problems.

The structure of each original linear system consists of 4x4 blocks for every finite element node in the mesh (three velocity components and pressure). For the purpose of applying block preconditioning the system is rearranged. In the vector of unknowns, first, all velocity components at all nodes are placed (we will denote that part of $\boldsymbol{u}$ by $\mathbf{u_v}$), followed by the pressure degrees of freedom (denoted by $\mathbf{u_p}$). With this approach the system of equations can be written as:

$$\begin{pmatrix} \mathbf{D_{vv}} & \mathbf{D_{vp}} \\ \mathbf{D_{pv}} & \mathbf{D_{pp}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u_v} \\ \mathbf{u_p} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}}_\mathbf{w} \\ \mathbf{b_q} \end{pmatrix} \tag{2}$$

For classical mixed formulations without stabilization terms, the part $\mathbf{D_{vp}}$ is just the transpose of $\mathbf{D_{pv}}$, while the part $\mathbf{D_{pp}}$ vanishes. For the stabilized formulation additional terms appear in $\mathbf{D_{vp}}$, $\mathbf{D_{pv}}$ and $\mathbf{D_{pp}}$, while $\mathbf{D_{vv}}$ keeps its diagonally dominant form, due to the discretized time derivative term. The matrix $\mathbf{D_{vv}}$ depends additionally on the solution at the previous Picard's iteration, while the right hand side $\bar{\mathbf{b}}_w$ depends on the solution at the previous time step.

We solve the system (2) using the restarted GMRES method with left preconditioning [17]. At each GMRES iteration the two most time consuming steps are the multiplication of the residual vector (for the whole system) by the system matrix and then the application of the preconditioner. Formally, the preconditioner is represented as a matrix, that tries to approximate the inverse of the system matrix (the better the approximation, the faster the GMRES convergence) and the action of the preconditioner is represented as matrix-vector multiplication. In

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

practice, the preconditioner matrix is usually not formed, instead, an algorithm is applied for the input vector, that is equivalent to a linear operator. For block preconditioners, the algorithm becomes complex, with several matrices involved, and iterative methods used for approximating inverses.

3 A multigrid based block preconditioner for linear equations

Following the approach in SIMPLE methods for solving Navier-Stokes equations [15], we observe that the inverse of the system matrix in Eq. 2 can be decomposed into the following product:

$$\begin{pmatrix} \mathbf{D_{vv}} & \mathbf{D_{vp}} \\ \mathbf{D_{pv}} & \mathbf{D_{pp}} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{I} & -\mathbf{D_{vv}^{-1}}\mathbf{D_{vp}} \\ 0 & \mathbf{I} \end{pmatrix} \times \begin{pmatrix} \mathbf{D_{vv}^{-1}} & 0 \\ 0 & \mathbf{S^{-1}} \end{pmatrix} \times \begin{pmatrix} \mathbf{I} & 0 \\ -\mathbf{D_{pv}}\mathbf{D_{vv}^{-1}} & \mathbf{I} \end{pmatrix}$$

where $\mathbf{S}$ is the Schur complement for $\mathbf{D_{pp}}$

$$\mathbf{S} = \mathbf{D_{pp}} - \mathbf{D_{pv}}\mathbf{D_{vv}^{-1}}\mathbf{D_{vp}}$$

The action of the inverse of the system matrix on a vector (with the parts related to velocity components and pressure denoted by $\mathbf{z_v}$ and $\mathbf{z_p}$ respectively) can be written as:

$$\begin{pmatrix} \mathbf{D_{vv}} & \mathbf{D_{vp}} \\ \mathbf{D_{pv}} & \mathbf{D_{pp}} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{z_v} \\ \mathbf{z_p} \end{pmatrix} = \begin{pmatrix} \mathbf{D_{vv}^{-1}} \left(\mathbf{z_v} - \mathbf{D_{vp}} \bar{\mathbf{z}}_\mathbf{p} \right) \\ \bar{\mathbf{z}}_\mathbf{p} \end{pmatrix}$$

with

$$\bar{\mathbf{z}}_\mathbf{p} = \mathbf{S^{-1}} \left(\mathbf{z_p} - \mathbf{D_{pv}}\mathbf{D_{vv}^{-1}}\mathbf{z_v} \right)$$

The above formulae would correspond to the application of the perfect preconditioner (being the exact inverse of the system matrix), that would guarantee the convergence of GMRES in a single iteration [17]. However, the construction of the presented exact form does not satisfy the requirement for the preconditioner to be relatively cheap, hence, some appoximations have to be performed.

We consider the approximation based on the SIMPLEC (Semi-Implicit Pressure Linked Equation Corrected) algorithm, where the action of the preconditioner is split into three steps [8]:

1. solve approximately: $\mathbf{D_{vv}}\tilde{\mathbf{z}}_\mathbf{v} = \mathbf{z_v}$
2. solve approximately: $\tilde{\mathbf{S}}\hat{\mathbf{z}}_\mathbf{p} = \mathbf{z_p} - \mathbf{D_{pv}}\tilde{\mathbf{z}}_\mathbf{v}$
3. substitute: $\hat{\mathbf{z}}_\mathbf{v} = \tilde{\mathbf{z}}_\mathbf{v} - \tilde{\mathbf{D}}_\mathbf{vv}^{-1}\mathbf{D_{vp}}\hat{\mathbf{z}}_\mathbf{p}$

where any vector with parts $\mathbf{z_v}$ and $\mathbf{z_p}$ is used as an input, and the output is stored in $\hat{\mathbf{z}}_\mathbf{v}$ and $\hat{\mathbf{z}}_\mathbf{p}$ (with an intermediate vector $\tilde{\mathbf{z}}_\mathbf{v}$). Above, $\tilde{\mathbf{S}}$ is an approximation to the original Schur complement matrix $\mathbf{S}$, that changes the original block

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

$\mathbf{D}_{\mathbf{vv}}^{-1}$ to some approximation. The approximation to $\mathbf{D}_{\mathbf{vv}}^{-1}$ is also used in step 3 of the algorithm (denoted their by $\tilde{\mathbf{D}}_{\mathbf{vv}}^{-1}$), although these two approximations to $\mathbf{D}_{\mathbf{vv}}^{-1}$ can be different.

The presented above three-step algorithm is used as the preconditioner in our GMRES solver, with the input parts $\mathbf{z_v}$ and $\mathbf{z_p}$ provided by the product of the system matrix and a suitable GMRES residual vector.

There are several factors influencing the quality of the preconditioner, and in consequence the convergence and scalability of the solver. The first factor is the quality of the approximation to $\mathbf{D}_{\mathbf{vv}}^{-1}$ in Step 1 of the procedure. In our implementation we solve approximately the system, simply by employing some number of Gauss-Seidel iterations to $\mathbf{D_{vv}}$. The convergence is sufficient to decrease the residual fast, partially due to the diagonal dominance of time derivative terms in $\mathbf{D_{vv}}$.

The second factor is the accuracy of obtaining $\hat{\mathbf{z}}_\mathbf{p}$ in Step 2. It is influenced by the solution procedure for the system of equations, as well as the choice of the approximation to $\mathbf{D}_{\mathbf{vv}}^{-\mathbf{1}}$ in the approximate Schur complement matrix $\tilde{\mathbf{S}}$.

Usually the approximation to $\mathbf{D}_{\mathbf{vv}}^{-\mathbf{1}}$ in $\tilde{\mathbf{S}}$ uses the diagonal form, with the matrix having inverted diagonal entries of $\mathbf{D_{vv}}$ as the simplest choice (the original SIMPLE algorithm). In the SIMPLEC approach, adopted in our implementation, the diagonal matrix approximating $\mathbf{D}_{\mathbf{vv}}^{-\mathbf{1}}$ have at each diagonal position the inverse of the sum of the absolute values of the entries in the corresponding row of $\mathbf{D_{vv}}$. We use the same approximation to $\mathbf{D}_{\mathbf{vv}}^{-\mathbf{1}}$ in Step 3 of the SIMPLEC algorithm.

Given the approximation to $\tilde{\mathbf{S}}$, the most important, from the computational point of view, is the choice of the solution procedure for the associated subsystem. We want to apply an iterative solver, since direct solvers become infeasible for large scale 3D problems. However, the system is difficult to solve due to its ill-conditioning (related to infinite speed of propagation of pressure changes in incompressible flows). Usually a multilevel solver is required in order to guarantee good convergence rates, independent of the mesh size (classical Krylov solvers with ILU preconditioning are not scalable with that respect). In the case of the stabilized methods, the addition of non-zero $\mathbf{D_{pp}}$ deteriorates the convergence by a large factor [18]. This is because without the stabilization the system has positive eigenvalues exclusively and the GMRES worst case convergence is different for matrices having eigenvalues of both signs [17].

We select a classical algebraic multigrid method, AMG [19], for the approximate solution of the system in Step 2 of the SIMPLEC algorithm. Gauss-Seidel iterations are used for smoothing at each level of the solver, with the coarser systems obtained using the Galerkin projection. The system at the last level is solved exactly using a direct solver. For the approximate solution we employ a single V-cycle of the AMG algorithm. Standard restriction and prolongation AMG operators [19] are used for projecting the solution between levels.

The key to the efficiency of the solver lies in the procedure for creating coarser levels, i.e. the selection of the degrees of freedom from the finer level that are retained at the coarser level. The construction of the hierarchy of system levels

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

is done during the levels set-up phase of the solver, performed once per system solution and followed by some number of V-cycle iterations. Usually one can select less levels that would lead to slower convergence but a faster single V-cycle iteration or more levels, with faster convergence and slower individual V-cycle iterations.

The cost of the set-up phase depends on the number of levels and a particular algorithm used for coarse level creation. We use a classical approach, that is based on partitioning, at each level, the degrees of freedom into two sets: interpolatory (retained at the coarser level) and non-interpolatory (removed form the system). In order to partition the DOFs, the notions of dependence and importance are introduced [19]. The importance is a measure of how many other rows are influenced by the solution for a given row. The relation opposite to the influence is called dependence.

We adopt the following formula for finding the set S_i of DOFs influencing a given DOF [10] (with the system matrix entries denoted by a_{ij}):

$$S_i \equiv \{j \neq i : -a_{ij} \geq \alpha \max_{k \neq i}(-a_{ik})\}$$

with the parameter $\alpha \in (0, 1)$ specifying the threshold for the strength of influence, that determines the inclusion of a DOF into the set S_i (we call α the strength threshold). An important observation is that only a single row of a matrix itself defines what influences a particular DOF, making this definition easy to use when the matrix is distributed on different computational nodes.

The DOFs are selected in the order of the number of DOFs that a given DOF influences. After selection of a DOF all DOFs that are influenced by this DOF are moved to the set of noninterpolatory degrees of freedom and the algorithm continues.

In our solution we chose to run this algorithm on every computational node separately, thus there is a possibility of dependencies between two DOFs on the boundaries of the subdomains. In our approach we let the owner of a DOF to assign it either to the set of interpolatory or noninterpolatory degrees of freedom and then broadcast this decision to all adjacent subdomains. Because of that procedure, the actual selections for different numbers of subdomains are not the same. This can lead to different convergence properties, that eventually make the numerical scalability of the solver problem dependent.

After the partition of DOFs, an interpolation matrix is created with each DOF having it's row and each interpolatory DOF having it's corresponding column. The rows associated with the interpolatory DOFs have just a single entry equal to one in a column related to this DOF. The rest of the rows is filled by the classic direct interpolation rules [19]. The interpolation matrix is used for the restriction and prolongation operations, as well as for the creation of the coarse systems using the Galerkin projection.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

4 Parallel implementation

The whole solution procedure is implemented within the finite element framework ModFEM [13] with the PETSC library [1] employed for linear algebra operations. ModFEM is a general purpose finite element software framework, with modular structure [2], that uses special problem dependent modules to create codes for different application domains [4]. In our setting the generic ModFEM modules are used to manage computational grids (in particular to perform domain decomposition for parallel execution) and to calculate element stiffness matrices and load vectors for the particular incompressible flow problem formulation that we employ.

We created a special ModFEM module for solving systems of linear equations that implements the algorithm described in the paper. The module receives local element matrices and vectors for the system of linear equations during Navier-Stokes simulations, assembles them to the global system matrix and right hand side vector, creates the other necessary preconditioner matrices, in particular the AMG levels structure, and performs preconditioned residual computations for the GMRES solver.

The module is built around matrix and vector data structures provided by the PETSC library. The PETSC linear algebra data structures and operations (including the sparse matrix-matrix product) serve as building blocks for the preconditioner responsibilities. Apart from basic matrix and vector operations, the only PETSC algorithm utilized during the set-up and solution phases is parallel matrix successive over-relaxation which is adapted to serve as Gauss-Seidel smoother. The parallel successive over-relaxation executes a configurable number of local iterations for each subdomain and a configurable number of global block Jacobi iterations, where blocks correspond to subdomain matrices. Such hybrid algorithm, frequently used in parallel iterative solvers, results in lower convergence rates than the Gauss-Seidel method at the global level, but has much lower cost and provides good scalability [3].

During the parallel solution procedures, communication steps are required only for global vector operations (norm, scalar product) and the exchange of data during Gauss-Seidel/Jacobi iterations. The scheme for exchanging data for ghost nodes is created by the generic ModFEM domain decomposition module and passed to the special module.

In our implementation we finally have the following set of control parameters to achieve the best GMRES convergence when using the developed preconditioner (in terms of CPU time for reducing relative error by a specified factor):

- the number of Gauss-Seidel iterations in step 1 of the SIMPLE procedure
- the form of $\mathbf{D_{vv}^{-1}}$ in the construction of the preconditioner for step 2
- the method and accuracy of solving the system in step 2
  - the number of pre-smooth steps at each level
  - the number of post-smooth steps at each level
  - the number of outer, additive Schwarz (block Jacobi) iterations and the number of inner, multiplicative Schwarz (block Gauss-Seidel) iterations for each smoothing step

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

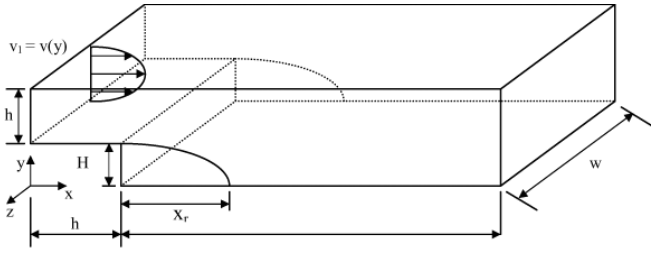


Fig. 1. 3D backward facing step - problem definition.

- the number of AMG levels and the size of the last level, for which a direct solver is used
- strength threshold which affects how many DOFs are dropped on subsequent levels

5 Numerical example

As a numerical example for testing the performance of the developed solver we take a well known stationary flow problem – backward facing step in its 3D form (Fig. 1). All boundary conditions are assumed as no-slip with zero velocity, except the inflow boundary with parabolic (with respect to y dimension) inflow velocity. The parameters are chosen in such a way, that the Reynolds number of the flow is equal to 800, which makes the finite element as well as the linear solver convergence difficult, but still remains in laminar regime. The step height, H, and the inflow height, h, are both assumed equal to 0.5.

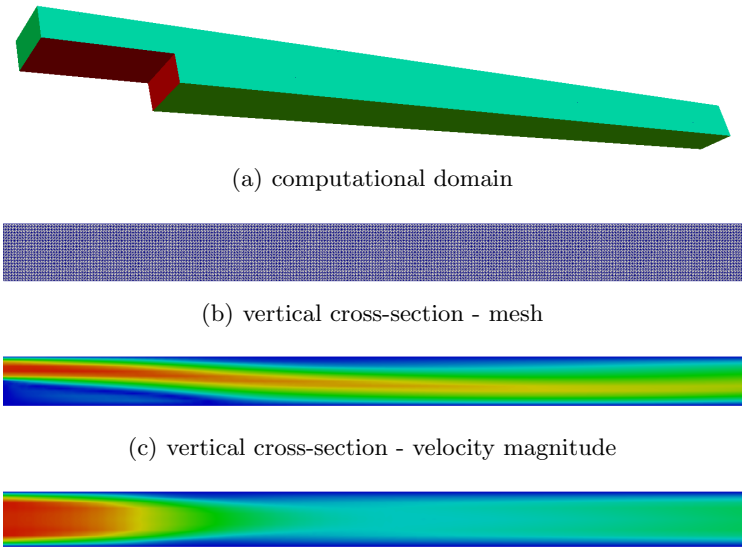
The actual computational domain with the dimensions 1x1x20 is triangulated with an initial (G0) mesh having 38 000 elements and 23 221 nodes (Fig. 2). For this mesh a stationary solution is obtained and then the mesh is uniformly refined to produce a new, generation 1 (G1), mesh with 304 000 elements and 168 441 nodes (673 764 degrees of freedom in the solved linear system). For this mesh the solution procedure is continued until the convergence and the same procedure is repeated for the next meshes. The uniform refinements produce the meshes:

– generation 2 (G2): 2 432 000 elements, 1 280 881 nodes, 5 123 524 DOFs
– generation 3 (G3): 19 456 000 elements, 9 985 761 nodes, 39 943 044 DOFs

The numerical experiments were performed on different numbers of nodes from the Prometheus system at Cyfronet AGH computing centre. Each node has two 12-core Intel Xeon E5-2680v3 CPUs (2.5GHZ) and 128GB DRAM and runs under Centos7 Linux version.

We present the performance measurements for a single typical time step and one non-linear iteration during the simulation. For the purpose of our tests we performed the GMRES solver iterations with high accuracy, stopping the process when the relative residual dropped by the factor of 10^9.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15



(a) computational domain

(b) vertical cross-section - mesh

(c) vertical cross-section - velocity magnitude

(d) horizontal cross-section at height H+0.5h - velocity magnitude

Fig. 2. 3D backward facing step problem – computational domain, G0 mesh and Re=800 solution contours for a part of the computational domain.

During the tests we tried to establish the best set of control parameters to be used for the multigrid phase of solving the system with approximated Schur complement. Fig. 3 presents the comparison of execution times for a single iteration of our solver on mesh G2, obtained with different sets of parameters. The symbols used for different lines on the plot contain encoded parameter values, in such a way that the symbol `nr-lev-a-pre-b-post-c-in-d-out-e-alpha-f` indicates the configuration with: `a` levels, `b` presmoothing steps, `c` post-smoothing steps, `d` inner, Gauss-Seidel iterations within single multigrid smoothing step, `e` outer, block Jacobi (additive Schwarz) iterations within single multirid smoothing step and the value of strength threshold α equal to `f` (the value 0 for the last parameter indicates a very small coefficient, in our experiments equal to 0.0001).

Similar results were obtained for executions on other mesh sizes, thus some general guidelines regarding the solver tuning can be deduced:

- additional AMG levels always speed up the convergence (less GMRES iterations required) and execution time, hence the number of levels should be such that additional level would not reduce the number of rows on the last level
- more aggressive coarsening tends to produce better level structure in terms of memory consumption and iterations execution time
- there is not much to be gained by manipulating local to global iterations ratio

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

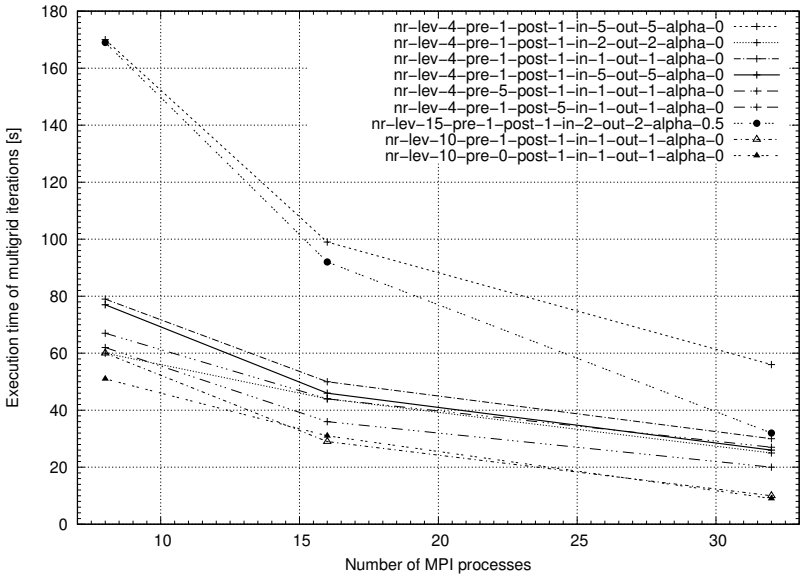


Fig. 3. 3D backward facing step problem – execution time on the Prometheus cluster for the iteration phase of the multigrid algorithm for different combinations of parameters (description of line symbols in the text)

– time spent in GMRES should be prioritized over time spent in the preconditioner i.e. adding more iterations does not always improve the convergence

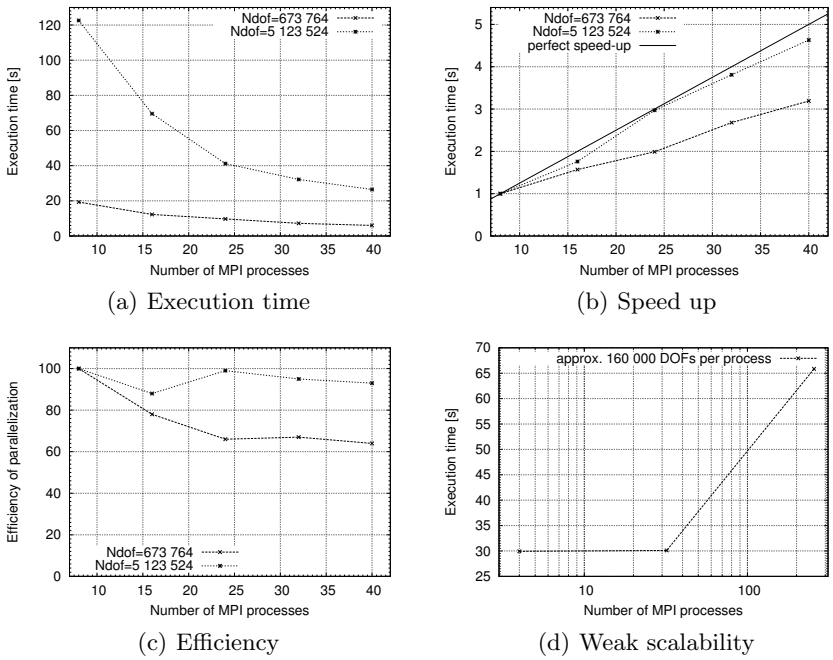
The best configuration in our experiments had the most aggressive coarsening possible, no limit imposed on the number of AMG levels and just one post-smoothing iteration with one global and local iteration, without pre-smoothing.

For this best configuration we show in Table 1 the comparison of the total linear system solution time on a single cluster node for our developed solver and a high performance direct solver (in this case the PARDISO from the Intel MKL library). The execution of the ModFEM code was done in MPI only mode, while the direct PARDISO solver was running in OpenMP only mode.

Table 1. 3D backward facing step problem – execution time (in seconds) for solving the system of linear equations (673 764 DOFs) using two solvers: the direct PARDISO solver from the Intel MKL library and the GMRES solver with the developed block preconditioner based on AMG

	number of cores				
solver	1	2	4	8	16
PARDISO	176.46	109.09	60.34	39.17	25.19
GMRES+AMG	115.51	56.61	29.93	16.87	10.39

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15



(a) Execution time (b) Speed up

(c) Efficiency (d) Weak scalability

Fig. 4. 3D backward facing step problem – parallel performance metrics for two problem sizes (Ndof equal to 673 764 and 5 123 524): execution time, speed up and parallel efficiency for the growing number of cluster processors plus weak scalability results as the execution time for 4, 32 and 256 processes with approx. 160 000 DOFs per process (subdomain)

In order to assess the computational scalability of the developed solver we performed a series of tests for the best algorithmic version. Fig. 4 presents the parallel performance characteristics obtained for two problem sizes (the number of DOFs equal to 673 764 and 5 123 524) and parallel runs on different number of processors in the Prometheus cluster. The metrics include execution time, standard parallel speed up and efficiency, as well as the results for weak scalability study. The latter was performed for approximately 160 000 DOFs per subdomain and three numbers of cluster nodes, with the last system, solved using 256 processes, having 39 943 048 DOFs.

The weak scalability study was also used to assess the numerical scalability of the algorithm. The results indicated that the convergence of the solver has not deteriorated for subsequent, refined meshes. The overall GMRES convergence rates were equal to 0.24 for G1 mesh, 0.14 for G2 mesh and 0.19 for G3 mesh. This strong convergence was obtained with the same time step length for all meshes and, hence, the growing CFL number for refined meshes. The different numbers of subdomains for each case influenced the convergence results as well. We plan to investigate these issues in forthcoming papers.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

6 Conclusions

We have shown that a proper design and tuning of parameters for algebraic multigrid method, used in block preconditioners employed to accelerate the convergence of GMRES method in finite element incompressible flow simulations, can lead to fast convergence and good scalability. The tests for a large scale example problem of 3D backward facing step produced solution times in the range of one minute for linear systems with approx. 40 million degrees of freedom and 256 processes (cores). For this size the standard direct methods or the GMRES method with standard ILU preconditioners cannot produce the results in the same order of time. We plan further investigations to show the strategies for optimal parameter selection depending on the CFL number in time discretization, that should lead to methods specifically adjusted to transient and steady-state problems.

References

1. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) Modern Software Tools in Scientific Computing. pp. 163–202. Birkhäuser Press (1997)
2. Banaś, K.: A modular design for parallel adaptive finite element computational kernels. In: Bubak, M., van Albada, G., Sloot, P., Dongarra, J. (eds.) Computational Science — ICCS 2004, 4th International Conference, Kraków, Poland, June 2004, Proceedings, Part II. Lecture Notes in Computer Science, vol. 3037, pp. 155–162. Springer (2004)
3. Banaś, K.: Scalability analysis for a multigrid linear equations solver. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) Parallel Processing and Applied Mathematics, Proceedings of VIIth International Conference, PPAM 2007, Gdansk, Poland, 2007. Lecture Notes in Computer Science, vol. 4967, pp. 1265–1274. Springer (2008)
4. Banaś, K., Chłoń, K., Cybułka, P., Michalik, K., Płaszewski, P., Siwek, A.: Adaptive finite element modelling of welding processes. In: Bubak, M., Kitowski, J., Wiatr, K. (eds.) eScience on Distributed Computing Infrastructure - Achievements of PLGrid Plus Domain-Specific Services and Tools, Lecture Notes in Computer Science, vol. 8500, pp. 391–406. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-10894-0_28
5. Brooks, A., Hughes, T.: Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with the particular emphasis on the incompressible Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering **32**, 199–259 (1982)
6. Cyr, E.C., Shadid, J.N., Tuminaro, R.S.: Stabilization and scalable block preconditioning for the Navier-Stokes equations. J. Comput. Physics **231**(2), 345–363 (2012). https://doi.org/10.1016/j.jcp.2011.09.001
7. Elman, H., Silvester, D., Wathen, A.: Finite Elements and Fast Iterative Solvers with applications in incompressible fluid dynamics. Oxford University Press (2005)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15

8. Elman, H., Howle, V., Shadid, J., Shuttleworth, R., Tuminaro, R.: A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. J. Comput. Phys. **227**(3), 1790–1808 (Jan 2008). https://doi.org/10.1016/j.jcp.2007.09.026
9. Franca, L., Frey, S.: Stabilized finite element methods II: The incompressible Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering **99**, 209–233 (1992)
10. Henson, V.E., Yang, U.M.: Boomeramg: A parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics **41**(1), 155–177 (2002). https://doi.org/https://doi.org/10.1016/S0168-9274(01)00115-5
11. Kelley, C., Keyes, D.: Convergence analysis of pseudo-transient continuation. SIAM Journal on Numerical Analysis **35**, 508–523 (1998)
12. Koric, S., Lu, Q., Guleryuz, E.: Evaluation of massively parallel linear sparse solvers on unstructured finite element meshes. Computers & Structures **141**, 19–25 (2014). https://doi.org/https://doi.org/10.1016/j.compstruc.2014.05.009
13. Michalik, K., Banaś, K., Płaszewski, P., Cybułka, P.: ModFEM – a computational framework for parallel adaptive finite element simulations. Computer Methods in Materials Science **13**(1), 3–8 (2013)
14. Pardo, D., Paszynski, M., Collier, N., Alvarez, J., Dalcin, L., Calo, V.M.: A survey on direct solvers for galerkin methods. SeMA Journal **57**, 107–134 (2012). https://doi.org/https://doi.org/10.1007/BF03322602
15. Patankar, S.V.: Numerical heat transfer and fluid flow. Series on Computational Methods in Mechanics and Thermal Science, Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group) (1980)
16. Pernice, M., Tocci, M.: A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations. Industrial and Applied Mathematics **23**, 398–418 (08 2001). https://doi.org/10.1137/S1064827500372250
17. Saad, Y.: Iterative methods for sparse linear systems. PWS Publishing, Boston (1996)
18. Segal, A., Rehman, M., Vuik, C.: Preconditioners for incompressible Navier-Stokes solvers. Numerical Mathematics – Theory, Methods and Applications **3** (08 2010). https://doi.org/10.4208/nmtma.2010.33.1
19. Stüben, K.: A review of algebraic multigrid. Journal of Computational and Applied Mathematics **128**(1), 281–309 (2001). https://doi.org/https://doi.org/10.1016/S0377-0427(00)00516-1
20. Thomas, S.J., Ananthan, S., Yellapantula, S., Hu, J.J., Lawson, M., Sprague, M.A.: A comparison of classical and aggregation-based algebraic multigrid preconditioners for high-fidelity simulation of wind turbine incompressible flows. SIAM J. Scientific Computing **41**(5), S196–S219 (2019). https://doi.org/10.1137/18M1179018

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_15