Scientific Research Publishing

Open Journal of Fluid Dynamics, 2023, 13, 133-143
https://www.scirp.org/journal/ojfd
ISSN Online: 2165-3860
ISSN Print: 2165-3852

Linear System Solutions of the Navier-Stokes Equations with Application to Flow over a Backward-Facing Step

Achraf Badahmane

LMPA, Université du Littoral Côte d'Opale, Calais Cedex, France
Email: badahmane.achraf@gmail.com

How to cite this paper: Badahmane, A. (2023) Linear System Solutions of the Navier-Stokes Equations with Application to Flow over a Backward-Facing Step. *Open Journal of Fluid Dynamics*, **13**, 133-143.
https://doi.org/10.4236/ojfd.2023.133011

Received: February 11, 2023
Accepted: June 3, 2023
Published: June 6, 2023

Abstract

Many applications in fluid mechanics require the numerical solution of sequences of linear systems typically issued from finite element discretization of the Navier-Stokes equations. The resulting matrices then exhibit a saddle point structure. To achieve this task, a Newton-based root-finding algorithm is usually employed which in turn necessitates to solve a saddle point system at every Newton iteration. The involved linear systems being large scale and ill-conditioned, effective linear solvers must be implemented. Here, we develop and test several methods for solving the saddle point systems, considering in particular the LU factorization, as direct approach, and the preconditioned generalized minimal residual ($\mathcal{P}$GMRES) solver, an iterative approach. We apply the various solvers within the root-finding algorithm for Flow over backward facing step systems. The particularity of Flow over backward facing step system is an interesting case for studying the performance and solution strategy of a turbulence model. In this case, the flow is subjected to a sudden increase of cross-sectional area, resulting in a separation of flow starting at the point of expansion, making the system of differential equations particularly stiff. We assess the performance of the direct and iterative solvers in terms of computational time, numbers of Newton iterations and time steps.

Keywords

Navier-Stokes Equation, $\mathcal{P}$GMRES, Direct Solver, Schur Approach, Preconditioner

1. Introduction

Numerical solution of the Navier-Stokes equations is a crucial problem in engi-

neering and physical sciences. We consider the solution of the incompressible Navier-Stokes equations governing the flow of viscous Newtonian fluids whose form reads

$$\begin{cases} -\boldsymbol{\nu}\nabla^2 u + u\cdot\nabla u + \nabla p = f & \text{dans } \Omega \\ \nabla\cdot u = 0 & \text{dans } \Omega, \end{cases} \tag{1}$$

where u is the velocity vector, p the pressure and f the field of external forces. The constant $\nu > 0$ is the kinematic viscosity. The first equation models the conservation of momentum fluid. The second equation models the conservation of mass. We consider the problem posed on a domain Ω of dimension 2 or 3 with boundary conditions defined by

$$\begin{aligned} u &= w && \text{sur } \partial\Omega_D, \\ \nu\frac{\partial u}{\partial n} - \boldsymbol{n}p &= 0 && \text{sur } \partial\Omega_N, \end{aligned} \tag{2}$$

where $\boldsymbol{n}$ denote normal vector and $\partial\Omega = \partial\Omega_D \cup \Omega_N$. We consider the following notations

$$L_2(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,|\, \int_\Omega u^2 < \infty \right\},$$

and Sobolev space is defined as follow:

$$H^1(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,|\, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L_2(\Omega) \right\}.$$

We define solution and test spaces:

$$H_E^1(\Omega) := \left\{ u \in H^1(\Omega)^2 \,|\, u = w \text{ sur } \partial\Omega_D \right\},$$

$$H_{E_0}^1 := \left\{ v \in H^1(\Omega)^2 \,|\, v = 0 \text{ sur } \partial\Omega_D \right\}.$$

The variational formulation of such problems is the following (1) find $u \in H_E^1(\Omega)$ and $p \in L_2(\Omega)$ such that:

$$\begin{aligned} &\boldsymbol{\nu}\int_\Omega \nabla u : \nabla v + \int_\Omega (u\cdot\nabla u)v - \int_\Omega p(\nabla\cdot v) = \int_\Omega f\cdot v \quad \forall v \in H_{E_0}^1, \\ &\int_\Omega q(\nabla\cdot u) = 0 \quad \forall q \in L_2(\Omega). \end{aligned} \tag{3}$$

Let $u = u_k + \delta u_k$ et $p = p_k + \delta p_k$. The linearized form of (3) is given as follow find $\delta u_k \in H_{E_0}^1$ and $\delta p_k \in L_2(\Omega)$, such that:

$$\begin{aligned} &D(u_k, \delta u_k, v) + \boldsymbol{\nu}\int_\Omega \nabla\delta u_k : \nabla v - \int_\Omega \delta p_k(\nabla\cdot v) = R_k(v), \\ &\int_\Omega q(\nabla\cdot\delta u_k) = r_k(q). \end{aligned} \tag{4}$$

where

$$D(u_k, \delta u_k, v) = \int_\Omega (\delta u_k\cdot\nabla\delta u_k)\cdot v + \int_\Omega (\delta u_k\cdot\nabla u_k)\cdot v + \int_\Omega (u_k\cdot\nabla\delta u_k)\cdot v.$$

$$R_k(v) = \int_\Omega f\cdot v - \int_\Omega (u_k\cdot\nabla u_k)v - \boldsymbol{\nu}\int_\Omega \nabla u_k : \nabla v + \int_\Omega p_k(\nabla\cdot v).$$

$$r_k(q) = -\int_\Omega q(\nabla\cdot u_k).$$

$\forall v \in H_{E_0}^1$ et $\forall q \in L_2(\Omega)$. We use finite dimensional spaces. Let $V^h \subset H_{E_0}^1$

et $Q^h \subset L_2(\Omega)$, the discret form of (4) is defined as follow: find $u_h \in V^h$ and $p \in Q^h$

$$\begin{aligned} &D(u_h, \delta u_h, v_h) + \nu \int_\Omega \nabla \delta u_h \nabla v_h - \int_\Omega \delta p_h \cdot \nabla v_h = R_k(v_h), \\ &\int_\Omega q_h \cdot \nabla \delta u_h = r_k(q_h), \end{aligned} \tag{5}$$

for all $v_h \in V^h$ and $q_h \in Q^h$.

Then we use (5) as in function of the basis V^h and Q^h, to find the following saddle point system

$$\begin{pmatrix} \nu A + D_k & B^{\mathrm{T}} \\ B & 0 \end{pmatrix} \begin{pmatrix} \Delta u^{(k)} \\ \Delta p^{(k)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_k \\ \boldsymbol{r}_k \end{pmatrix}, \tag{6}$$

where $\Delta u^{(k)} = u^{(k+1)} - u^{(k)}$, $\Delta p^{(k)} = p^{(k+1)} - p^{(k)}$, A denote the laplacian matrix

$$A = \left[a_{i,j} \right], \quad a_{i,j} = \int_\Omega \nabla \phi_i : \nabla \psi_j, \quad i, j = 1, \cdots, n_u.$$

B denote the divergence matrix

$$B = \left[b_{k,j} \right], \quad b_{k,j} = -\int_\Omega \psi_k \cdot \nabla \phi_j, \quad j = 1, \cdots, n_u; k = 1, \cdots, n_p.$$

D is the nonlinear matrix

$$D = \left[d_{i,j} \right], \quad d_{i,j} = \int_\Omega \left(u_h \cdot \nabla \phi_j \right) \cdot \phi_i + \int_\Omega \left(\phi_j \cdot \nabla u_h \right) \cdot \phi_i + \int_\Omega \left(\phi_j \cdot \nabla \phi_j \right) \cdot \phi_i,$$

where $i = 1, \cdots, n_u$, $j = 1, \cdots, n_u$ et $k = 1, \cdots, n_p$. The right-hand side is defined as follow

$$\boldsymbol{R} = \left[R_i \right], \quad R_i = \int_\Omega f \cdot \phi_i - \int_\Omega \left(u_h \cdot \nabla u_h \right) \cdot \phi_i - \nu \int_\Omega \nabla u_h \nabla \phi_i + \int_\Omega p_h \cdot \nabla \phi_i,$$

$$\boldsymbol{r} = \left[r_i \right], \quad r_k = \int_\Omega \psi_k \cdot \nabla u_h.$$

For finding the root, the Newton method is often implemented. It is an iterative algorithm consisting in assigning the root of the linearized equation at the current iterate to the next iterate to refine the solution until a convergence criterion is satisfied. Implementing the Newton method requires evaluating the Jacobian matrix. In the Newton method, algebraic linear systems (6) involving the Newton matrices must thus be solved at each iterate. We are thus interested in the following linear system:

$$\mathcal{S}X = b, \tag{7}$$

where matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$ is generally nonsymmetric and vector $b \in \mathbb{R}^N$ is the right-hand side at the current iterate. This linear system can be solved directly by decomposing the Newton matrix into the product of a lower triangular and upper triangular factor (LU factorization) and by solving the two resulting triangular systems through forward and backward substitutions. Whenever it is prohibitively costly to assemble Newton matrix $\mathcal{S}$, the linear system (1) is to be solved using a matrix free method. It is indeed possible to resort to iterative linear methods, for instance a Krylov subspace projection technique. Implementing any iterative method in this context would require to repeatedly compute the application of the Newton matrix on the current iterate. Numerical differentiation via the difference quotient being prone to truncation and round-off errors,

this inexact way of proceeding is usually referred to as the inexact Newton method. More general approximations of the Newton matrix can be used to perform the Newton iterations. Proceeding this way precisely amounts to implementing a modified Newton method. This means that an approximate inverse of the Newton matrix is employed to indicate the tangent direction. The exact Newton method resorting to the exact Newton matrix $\mathcal{S}$, as opposed to the inexact or modified Newton methods, is seldom used because assembling the exact Newton matrix at each iteration is computationally too expensive in most practical problems of interest. In this work, we investigate three alternative strategies to perform the Newton iterations based on the factorization of an approximate Newton matrix.

The first strategy consists in implementing a modified Newton method and solving the linear systems (7) appearing at the Newton iterations using lower-upper factorization (LU). Is an efficient solver to perform the LU factorization of a sparse matrix and the triangular resolutions. Factoring the Newton matrix being computationally intensive, the task is done occasionally, not at every Newton iteration nor even time step. The previously updated Newton matrix is recycled several times. The Newton matrix is computed again when the number of Newton iterations to satisfy the converge criterion exceeds a certain parameter value. This is the default way of proceeding of the IFISS software [1] used in numerous incompressible flow simulation [2]. The second strategy is to solve the linear systems occurring during the Newton iterations using an iterative method. Any iterative solver consists in generating a sequence of improving approximate solutions by repeatedly applying the current approximate onto the matrix until a convergence criterion is satisfied. It can possibly be implemented within both the modified and inexact Newton methods, depending on whether the matrix-vector product involves an approximate Newton matrix or is computed inexactly from a different quotient approximating the exact Newton matrix. We apply the generalized minimum residual (GMRES) method [3] [4] [5]. This iterative method is based on the standard Arnoldi algorithm [6], is a particular Krylov subspace projection method and is well adapted to nonsymmetric indefinite linear systems of equations, unlike the conjugate gradient method that rather deals with symmetric definite positive matrices. Krylov subspace projection methods are suited to solve linear systems in which the involved matrix is large and sparse. An important number of iterations are however necessary to obtain a reliable approximate of the solution when the condition number of the matrix is large. For ill-conditioned matrices, it is preferable to apply a preconditioner to accelerate the convergence of the associated GMRES method. In practice, the preconditionning matrix $\mathcal{P}$ should be chosen close to the inverse of the Newton matrix. Here, we consider the Newton matrix obtained at a previous time stem and compute its LU factorization [7] or an incomplete LU factorization [8]. Preconditioning then amounts to applying the inverted matrix on the current approximate by solving the two associated triangular linear systems by forward and backward substitutions.

The third strategy consists in solving the linear systems using a Schur approach. The goal is to take advantage of the particular structure of the saddle point matrix. N corresponds to the number of differential equations. The saddle point system to solve can be rewritten as follows:

$$\mathcal{S}X = \begin{pmatrix} \tilde{A} & B^{\mathrm{T}} \\ B & O \end{pmatrix} \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} f \\ g \end{pmatrix}}_{b}, \tag{8}$$

where: $\tilde{A} = (\nu A + D) \in \mathbb{R}^{n \times n}$ is a sparse and nonsymmetric matrix, $B \in \mathbb{R}^{n \times m}$ maybe a rank deficient matrix with $\operatorname{rank}(B) < m$, $D \in \mathbb{R}^{m \times m}$ is a dense matrix, $f \in \mathbb{R}^{n}$ and $g \in \mathbb{R}^{m}$ are given vectors. Besides, we have $N = n + m$ with $n \gg m$. The Schur approach consists in computing the Schur complement of the partitioned Newton matrix, in which the linear system associated with the large sparse block is handled by LU factorization. The paper is organized as follows. We next introduce the direct approach (Section 2.1), iterative approach (Section 2.2) and Schur approach (Section 2.3) used for solving the saddle point systems arising within the modified or inexact Newton algorithms. We present simulation results for the lid driven cavity and chanel flow systems obtained using the various solvers in Section 3. We finally conclude in Section 4 and give some recommendation on which approach to implement depending on the nature of the problem.

2. Implementation of Methods

With the exact Newton method, the saddle point matrix must be assembled at every iteration, which represents a costly task. A modified Newton method that is often computationally more efficient in practice consists in reusing the Newton matrix several times. This way of proceeding usually necessitates more Newton iterations, but these ones are performed faster. In all implementations of Newton methods, we allow at most 10 Newton iterations, but this limit can be changed by the user. The input constants used by the main solver (maximum number of Newton) as well as their meaning are described in detail in Ref [1].

2.1. Direct Approach

Linear Equation (8) is solved directly by resorting to lower-upper factorization implemented [9]. Once the saddle point matrix has been factorized into $\mathcal{S} = LU$, the linear system (8) is solved via forward and backward substitution. This amounts to successively solving the two following triangular systems of equations

$$LY = b, \tag{9}$$

$$UX = Y. \tag{10}$$

The solution Y of system (9) as obtained via forward substitution is to be injected into system (10), yielding X via backward substitution. Solving the two triangular linear systems amounts to performing a modified Newton iteration whose cost is much smaller than that of factoring the Newton matrix.

2.2. Preconditioned Iterative Approach

Iterative methods are ideally suited to solve high-dimensional sparse linear systems of equations of the form (7). Their advantage over direct methods is that they don't require to factorize saddle point matrix $\mathcal{S}$ nor even evaluate it. This is the case for Krylov subspace projection methods and, among them the GMRES method implemented in the following. The only request is the ability to compute the application of the matrix on any vector v. In the present context, this task can be done without evaluating the saddle point matrix. Hence, vector $\mathcal{S}v$ can be obtained through a finite difference along direction v.

The drawback of iterative methods is that they are inexact and may require a large number of iterations so that the iterate satisfies the tolerance condition. The remedy to this technical drawback is called preconditioning. Here, the preconditioner $\mathcal{P}$ applies a linear transformation to system (7) so as to reduce the condition number of the transformed matrix $\mathcal{P}^{-1}\mathcal{S}$. The preconditioned linear system to solve writes:

$$\mathcal{P}^{-1}\mathcal{S}X = \mathcal{P}^{-1}b. \tag{11}$$

If the condition number of the transformed matrix $\mathcal{P}^{-1}S$ is smaller than that of matrix $\mathcal{S}$ then the number of iterations is generally reduced. Several ways of implementing the preconditioned iterative approach can be designed depending on the choice of the preconditioning linear transformation $\mathcal{P}$ and Newton matrix $\mathcal{S}$. We furthermore consider the preconditioner built upon the incomplete lower-upper (ILU) factorization of the last-updated Newton matrix. Its potential advantage resides in the smaller amount of memory necessary to perform an incomplete factorization. This feature is useful when the model system is very large and the Newton matrix cannot be factored due to memory limitations. The three preconditioners are denoted by $\mathcal{P}_D$, $\mathcal{P}_T$ and $\mathcal{P}_R$ respectively, while notations $\mathcal{P}_D$GMRES, $\mathcal{P}_T$GMRES and $\mathcal{P}_R$GMRES stand for the corresponding preconditioned GMRES methods. When solving Equation (11) using any of the preconditioned GMRES methods described above, the iterations are stopped as soon as the Euclidean norm of the current residue is lower than a tolerance threshold ($j \in \{D,T,R\}$):

$$\frac{\left\| \mathcal{P}_j^{-1}b - \mathcal{P}_j^{-1}\mathcal{S}X^{(k)} \right\|_2}{\left\| \mathcal{P}_j^{-1}b \right\|_2} < \varepsilon, \tag{12}$$

where $X^{(k)}$ denotes the current iterate and $\varepsilon = 10^{-6}$ is the threshold value. The maximum number of iteration of the $\mathcal{P}$GMRES is 200.

2.3. Schur Approach

The Schur approach is an alternative direct method aiming at solving linear system (8). The goal is to take advantage of the structure of the saddle point matrix $\mathcal{S}$. The preliminary steps for implementing the Schur approach consists in computing the Schur complement matrix S_{schur} associated with matrix $\mathcal{S}$ as illustrated in **Figure 1** and listed below:

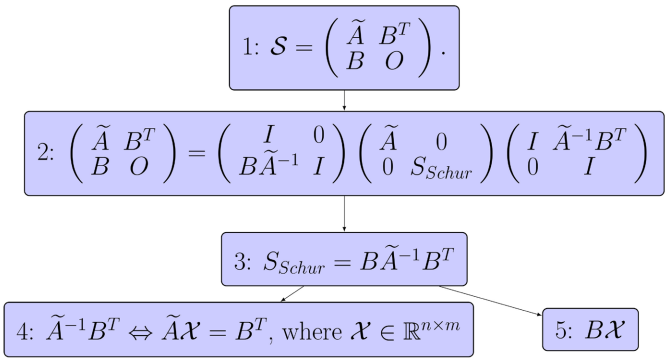

Figure 1. Schematic diagram illustrating the block LDU decomposition of S and the computation of the Schur complement matrix.

1) Perform the block decomposition of $\mathcal{S}$ into four block matrices such that the diagonal block $\tilde{A}$ and matrix D.

2) Formally perform the lower-diagonal-upper block decomposition (LDU) of S_{schur}.

3) Formally define the Schur complement matrix S_{schur}.

4) Solve the systems $\tilde{A}\mathcal{X} = B^{\mathrm{T}}$, with multiple right-hand sides $B^{\mathrm{T}} \in \mathbb{R}^{n\times m}$ using LU factorization.

5) Evaluate the Schur complement S_{Schur} by computing the product of B^{T} and $\mathcal{X}$.

The next steps of the Schur approach consist in solving the two triangular systems of linear block equations using the block LDU factorization so as to obtain the solution of overall linear system (8). These additional steps are depicted in Figure 2 and detailed below:

1) The matrix, unknown and right-hand side of linear system (8) are first partitioned.

2) The solution part y is computed through forward block substitution; the dense system involving the previously computed Schur matrix $\mathcal{S}$ is solved through LU factorization.

3) The solution part x is computed through backward substitution and the sublinear system involving A is solved directly using LU.

4) The solution X of block linear system (8) is eventually obtained by concatenating partial solutions x and y.

2.4. Modified Newton Method

In the following applications, the root-finding algorithm will be a modified Newton method when used in combination with preconditioned iterative solver, also Schur approach and direct approach. In this situation, the Newton matrix $\mathcal{S}$ is computed at a previous time step and is fixed throughout the nonlinear iterations. When used in combination with an iterative approach, *i.e.* the preconditioned GMRES method, the root-finding algorithm is either a modified Newton method, depending on whether the matrix-vector products $\mathcal{S}v$ is computed using a partially updated Newton matrix.

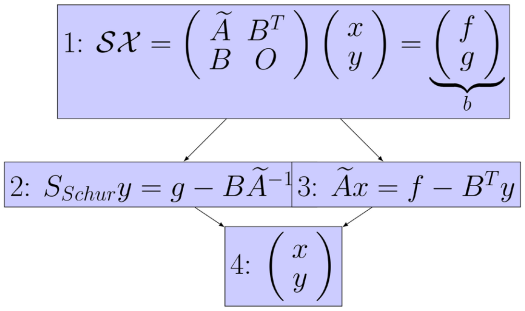

Figure 2. Schematic diagram of forward and backward block substitution in Schur approach. The main costs at each modified Newton iteration consist in solving the two sublinear systems involving the sparse high-dimensional matrix Ae. The small and dense linear system involving matrix S is solved using LU factorization since the dimension of the matrix is too small.

3. Numerical Results

In this section, we carry out two numerical example simulations, to assess the relative efficiencies of the various algebraic numerical methods described in Sections 2.1, 2.2 and 2.3. For the direct, iterative and Schur approaches, In the meanwhile the feasibility and effectiveness of the approaches, from the point of view of the elapsed Central Process Units time for the solution of saddle point problem (8) (denoted by CPU) as well as the number of iterations of $\mathcal{P}$GMRES. The parameters introduced in the lid driven cavity and flow over backward facing step are listed in **Table 1**, for more details see [1]. We use the IFISS software package developed by Elman *et al.* [1] to generate the linear systems corresponding to $l=4$, $l=5$ and $l=6$. The IFISS software provides the matrices A, B, and the right-hand side f and g. Generic information of the test problems, including n and m, are provided in **Table 1**.

In practice, the preconditioners used for solving (6) are $\mathcal{P}_D$, $\mathcal{P}_T$ and $\mathcal{P}_R$, where $\mathcal{P}_D$, $\mathcal{P}_T$ and $\mathcal{P}_R$ are given as follows:

$$\mathcal{P}_R = \begin{bmatrix} A & B^{\mathrm{T}} \\ B & \alpha Q \end{bmatrix}, \quad \mathcal{P}_T = \begin{bmatrix} A & O \\ B & Q \end{bmatrix} \text{ and } \mathcal{P}_D = \begin{bmatrix} A & O \\ O & Q \end{bmatrix}. \tag{13}$$

Here S is a sparse approximation of the pressure Schur complement $S_{Schur} = BA^{-1}B^{\mathrm{T}}$, and Q is one of the matrices I or (S). The parameter of the $\mathcal{P}_R$ preconditioner is chosen as to implement the regularized preconditioner effeciently, we need to choose the parameters α appropriately since the analytic determination of the parameters which results in the fastest convergence of the preconditioned GMRES iteration appears to be quite a difficult problem. In all Tables, to implement the regularized preconditioner efficiently, we need to choose the parameters α appropriately since the analytic determination of the parameters which results in the fastest convergence of the preconditioned GMRES iteration appears to be quite a difficult problem. In the regularized preconditioner, the parameter α is taken as $\alpha = \left(\left\|B^{\mathrm{T}}\right\|_2 \|B\|_2\right) / \left(\|A\|_2 \|S\|_2\right)$, which balances the matrices A and $B^{\mathrm{T}} S^{-1} B$ in the Euclidean norm.

Example: Flow over backward facing step

We consider the Flow over backward facing step. The simulation parameters used for Lid driven cavity model are summarized in Table 1, for more details see [1]. Numerical results for all approaches are presented in Tables 2-4.

In all Tables indicates that the $\mathcal{P}_R$ preconditioner with $Q = I$ leads to much better numerical results than the $\mathcal{P}_D$ and $\mathcal{P}_T$ preconditioned GMRES methods less CPU times in all trials and iteration. the preconditioned $\mathcal{P}_R$GMRES method with the proper parameter α has a better performance than the preconditioned $\mathcal{P}_D$GMRES and the preconditioned $\mathcal{P}_T$GMRES methods in terms of the iterations and CPU times. In the following figures, we display a streamline plot for the velocity solution, and a plot of the pressure solution of the Flow in a symmetric step channel, over a plate and over a backward facing step.

Figure 3 illustrates that the flow and pressure rendering in a rectangular duct containing a sudden expansion. We set the conditions of Dirichlet at the entry of the rectangular conduit and the conditions of Neumann at the exit of the duct. The vertical speed is zero, which means that the lines of the fluid flow propagate in a parallel way to the rectangular duct walls. Rendering pressure becomes closer to zero at the exit of the duct.

Table 1. The size of the matrices A and B on $2^l \times 2^l$.

Flow over backward facing step[a]				
l	n	m	size of A	size of B
4	578	192	578 × 578	578 × 192
5	2178	768	2178 × 2178	2178 × 766
6	8450	3070	8450 × 8450	8450 × 3070
7	33,282	12,288	33,282 × 33,282	33,282 × 12,288

[a]Flow over backward facing step.

Table 2. The numerical results of direct, Iterative and schur approaches.

$l = 4$	Direct solver[a]	Iterative solver[b]			Schur solver[c]	
	LU	$\mathcal{P}_D$GMRES	$\mathcal{P}_T$GMRES	$\mathcal{P}_R$GMRES	Direct	Iterative
CPU	†	0.73	0.75	0.07	†	†
ITER	†	282	217	4	†	†

[a]Direct solver. [b]Iterative solver. [c]Schur solver.

Table 3. The numerical results of direct, iterative and schur approaches.

$l = 5$	Direct solver[a]	Iterative solver[b]			Schur solver[c]	
	LU	$\mathcal{P}_D$GMRES	$\mathcal{P}_T$GMRES	$\mathcal{P}_R$GMRES	Direct	Iterative
CPU	†	4.73	3.96	0.90	†	†
ITER	†	296	254	4		

[a]Direct solver. [b]Iterative solver. [c]Schur solver.

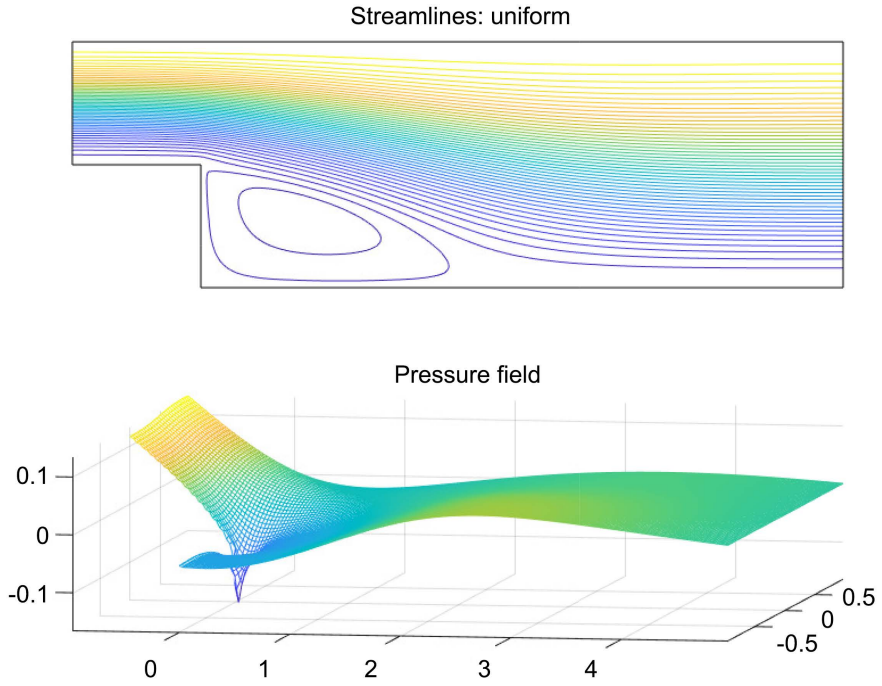

Figure 3. The velocity and pressure solutions of flow over backward facing step.

Table 4. The numerical results of direct, iterative and schur approaches.

$l=6$	Direct solver[a]	Iterative solver[b]			Schur solver[c]	
	LU	$\mathcal{P}_D$GMRES	$\mathcal{P}_T$GMRES	$\mathcal{P}_R$GMRES	Direct	Iterative
CPU	†	46.54	38.72	17.42	†	†
ITER	†	322	271	4		

[a]Direct solver. [b]Iterative solver. [c]Schur solver.

4. Conclusion

The numerical results reported in Section 3 show that the preconditioned GMRES methods are more efficient than LU and Schur approaches. The two latter approaches are more expensive computationally in term of CPU times outperforms the other methods. For the test presently illustrated, the speed-up factors increases with the size of the Jacobian matrix and reaches a value of three. The Flow over backward facing systems for which the $\mathcal{P}$GMRES method preconditioning exhibits speed-ups that are even more important compared with the implementation of Schur and direct approaches. As a result, the system of ordinary differential equations is extremely stiff. The stiffness implies that the saddle point matrix has a large condition number, *i.e.* is ill-conditioned. In this situation, Modified-Newton methods based on the inverse of the approximate Jacobian matrix require an important number of iterations. With $\mathcal{P}_R$GMRES, the iterations are based on the exact Jacobian inverse and for this reason the method reverts to the standard Newton method that converges in fewer iterations. Future developments will focus on the deterministic/Navier-Stokes that is enabled

in [1] code when the simulated system is too large. When this situation occurs, the linear solver being very sensitive to solve the linear system, other direct linear solver will be tested. Future research should be devoted to the construction of parallel iterative solvers robust with respect to higher order virtual element discretizations and non-symmetric problems, such as saddle point systems deriving from discretizations of the Navier-Stokes equations.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Elman, H.C., Silvester, D.J. and Wathen, A.J. (2011) IFISS: A Computational Laboratory for Investigating Incompressible Flow Problems. *SIAM Review*, **56**, 261-273.

[2] Elman, H.C., Silvester, D.J. and Wathen, A.J. (2011) Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics. Oxford University Press, Oxford.

[3] Saad, Y. and Schultz, M.H. (1986) The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem. *SIAM Journal on Scientific and Statistical Computing*, **7**, 856-869.

[4] Saad, Y. (2003) Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898718003

[5] Badahmane, A. (2020) Regularized Preconditioned GMRES and the Regularized Iteration Method. *Applied Numerical Mathematics*, **152**, 159-168. https://doi.org/10.1016/j.apnum.2020.01.001

[6] Arnoldi, W.E. (1951) The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem. *Quarterly of Applied Mathematics*, **9**, 17-29, https://ci.nii.ac.jp/naid/10020869898/en/ https://doi.org/10.1090/qam/42792

[7] Gilbert, J.R. and Peierls, T. (2016) Sparse Partial Pivoting in Time Proportional to Arithmetic Operations. *SIAM Journal on Scientific and Statistical Computing*, **9**, 862-874. https://epubs.siam.org/doi/10.1137/0909058 https://doi.org/10.1137/0909058

[8] Saad, Y. (1996) Preconditioning Techniques. Chap. 10 in Iterative Methods for Sparse Linear Systems. *The PWS Series in Computer Science*, **9**, 862-874.

[9] Amestoy, P.R., Duff, I.S., L'Excellent, J.-Y. and Koster, J. (2001) A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. *SIAM Journal on Matrix Analysis and Applications*, **23**, 15-41. https://doi.org/10.1137/S0895479899358194