ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

ALTERNATIVE SOLUTION ALGORITHMS FOR PRIMAL AND ADJOINT INCOMPRESSIBLE NAVIER-STOKES

Mattia Oriani[1,2], **Guillaume Pierrot**[1]

[1]ESI Group
99 Rue des Solets, Rungis 94513, France
{mattia.oriani,guillaume.pierrot}@esi-group.com

[2] Queen Mary, University of London
Mile End Road, London E1 4NS, UK
m.oriani@qmul.ac.uk

Keywords: Oseen Preconditioners, Adjoint Navier-Stokes, Block-Coupled, SIMPLEC, V-Coupled, Augmented Lagrangian.

Abstract. *Regardless of the specific discretisation framework, the discrete incompressible Navier-Stokes equations present themselves in the form of a non-linear, saddle-point Oseen-type system. Traditional CFD codes typically solve the system via the well-known SIMPLE-like algorithms, which are essentially block preconditioners based on Schur complement theory. Due to their "segregated" nature, which reduces to iteratively solving a sequence of linear systems smaller than the full Oseen and better conditioned, traditional SIMPLE-like algorithms have long been considered as the only viable strategy.*

However, recent progress in computational power and linear solver capabilities has led researchers to develop, for Oseen-type systems (and discrete Navier-Stokes in particular), a number of alternative preconditioners and solution schemes, found to be more efficient than SIMPLE-like strategies but previously deemed practically unfeasible in industrial contexts.

The improved efficiency of novel preconditioners entails a) faster, more stable convergence and b) the possibility of driving residuals below more strict tolerances, which is sometimes difficult with SIMPLE due to stagnating behaviour. The second aspect in particular is extremely relevant in the context of adjoint-based optimisation, as evidence suggests that an adjoint system may be affected by convergence issues when the primal flow solution is not well converged.

In this work, we present some solution schemes (both traditional and novel) implemented for the Mixed Hybrid Finite Volumes Navier-Stokes solver we introduced in our previous work. Performance, in terms of robustness and convergence properties, is assessed on a series of benchmark test cases. We also turn our attention to the discrete adjoint Navier-Stokes problem itself, which in essence requires solving a linear system similar to the original Oseen and therefore may benefit from the same preconditioning techniques. We show how the primal algorithms are adapted to the adjoint system, and we run a series of adjoint test cases to compare performance of various solution schemes.

1 INTRODUCTION

In numerical optimisation, gradient-based methods are defined as a way of actively searching for a local minimum of a given cost function J via minimisation algorithms that make use of its sensitivity, i.e. its gradient with respect to a chosen set of design parameters α. Typical examples related to fluid mechanics include: reducing the drag force exerted on an aircraft wing by seeking the optimal airfoil profile; minimising the total pressure loss of a flow through a duct by modifying the shape of the duct's cross section.

Performing numerical optimisation in an industrial context can be prohibitively costly for two reasons. Firstly, each iteration of the optimisation loop requires an evaluation of J, and therefore a CFD solve. Secondly, extra computations are required in order to get the sensitivity $\frac{dJ}{d\alpha}$. The second issue in particular is a blocking factor in an industrial context: gradient computation via traditional methods - finite differencing (FD) or forward-mode Algorithmic Differentiation (AD) [11] - demands as many extra CFD solves as the number of design variables n_α, which may be in the order of millions. This is notably the case for shape optimisation processes where we want to compute the *surface sensitivity*, i.e. the gradient of J w.r.t. the coordinates of each mesh node lying on the surface to be optimised.

The *adjoint method* [12, 13, 22] provides a workaround to the problem, as it allows to compute the sensitivity of J at a cost that is essentially independent of n_α. Our work is based on the *discrete adjoint* approach, which consists in applying adjoint theory directly to the set of discrete equations describing the original (or *primal*) problem. In short, a discrete adjoint boils down to solving a linear system in the form:

$$\left(\frac{\partial \mathbf{r}}{\partial \mathbf{u}}\right)^T \mathbf{u}^* = -\frac{\partial J}{\partial \mathbf{u}} \tag{1}$$

where $\mathbf{r}$ is the primal residual vector and $\mathbf{u}^*$ the discrete *adjoint field*.

The adjoint method, powerful as it is, suffers from stability and convergence issues. Solving linear system (1) is not always a trivial task, and it was shown [20] that, if the primal solution is not properly converged, then the adjoint might not converge at all. Evidence suggests that such lack of robustness is to be blamed in part on the quality of the CFD solver itself, which in turns depends on two key factors:

- the level of consistency and robustness of the discrete operators used in the primal;
- the efficiency of the algorithm used to converge the primal flow field.

In other words, even though a conventional CFD solver is able to produce results that are "good enough" for mere engineering purposes, its discrete adjoint counterpart may be unstable or too inaccurate to be of any use, since its quality is highly sensitive to how the primal governing equations were discretised, as well as the tolerance down to which they were solved.

In our previous work [17, 18, 19] we addressed the task of finding robust discretisation schemes for the Navier-Stokes equations, which we summarise in the next section. In this paper we will turn our attention to developing improved solution algorithms for both primal and adjoint systems.

1.1 Previous work: discretisation schemes

Traditional CFD solvers often rely on classical Finite Volumes (FV) schemes to discretise the incompressible Navier-Stokes equations, which seem to produce discrete adjoints that some-

what lack robustness. In our recent work we attempted to tackle such issue, focusing our research on non-standard discretisation schemes. We looked in particular into a class of methods formerly known as Mimetic Finite Differences (MFD), now renamed Virtual Elements Methods (VE) [5], originally developed for pure anisotropic diffusion problems [3, 4] and subsequently extended to convection-diffusion operators [7, 26] and 1st-order accurate Navier-Stokes [8]. Compared to FV, VE operators sport a series of attractive features, most notably:

- they "mimic" at a discrete level certain key properties of their continuous counterparts (e.g. they satisfy the discrete Gauss-Green formula);
- they are perfectly consistent up to a set order of accuracy, which can be increased to an arbitrary level by attaching extra degrees of freedom to each mesh element;
- they are fully implicit and free of numerical artefacts (e.g. the Non-Orthogonal Correctors typically found in FV schemes);
- their requirements in terms of mesh quality are minimal, i.e. they can deal with highly skewed, non-orthogonal or non-convex elements.

We refer the reader to our previous publications for details on our specific implementation of a VE diffusion operator [17], the addition of a convective term [18], the extension to stabilised 2nd-order accuracy and our implementation of a Mixed Hybrid Finite Volumes (MHFV) Navier-Stokes solver [19]. Here we limit ourselves to providing a very simplified outline of our framework.

The main degrees of freedom for a generic mesh cell (Figure 1) are:

- face-averaged velocities $\vec{U}_F$, with components u_F, v_F, w_F;
- cell-averaged pressure p_C.

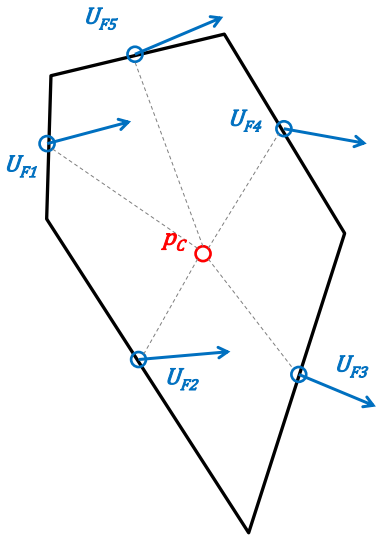


Figure 1: Placement of main MHFV flow variables on a generic 2D cell.

Considering now the incompressible, steady-state Navier-Stokes equations

$$\begin{cases} \vec{U} \cdot \nabla \vec{U} - \nabla \cdot \nu \nabla \vec{U} = -\nabla p + \vec{g} \\ \nabla \cdot \vec{U} = 0 \end{cases}, \tag{2}$$

when discretised via our MHFV operators they yield an Oseen-type saddle-point system

$$\begin{bmatrix} \mathcal{F}_{\Phi,x} & 0 & 0 & \mathcal{G}_x \\ 0 & \mathcal{F}_{\Phi,y} & 0 & \mathcal{G}_y \\ 0 & 0 & \mathcal{F}_{\Phi,z} & \mathcal{G}_z \\ \mathcal{D}_x & \mathcal{D}_y & \mathcal{D}_z & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}_F \\ \mathbf{v}_F \\ \mathbf{w}_F \\ \mathbf{p}_C \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{F,u} \\ \mathbf{g}_{F,v} \\ \mathbf{g}_{F,w} \\ \mathbf{0} \end{pmatrix} \tag{3}$$

or, in a more compact form:

$$\begin{bmatrix} \mathcal{F}_\Phi & \mathcal{G} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{U}_F \\ \mathbf{p}_C \end{pmatrix} = \begin{pmatrix} \mathbf{g}_F \\ \mathbf{0} \end{pmatrix}. \tag{4}$$

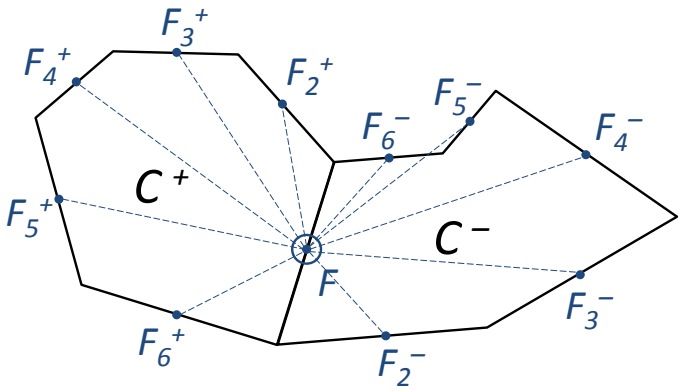


Figure 2: Face-to-face stencil on a generic 2D mesh.

- $\mathcal{F}_{\Phi,i}$ stands for any version of the MHFV hybrid convection-diffusion operator described in [19], acting on the i^{th} velocity component. It is identical in all 3 spatial directions (boundary conditions aside), and it is of course non-linear, since it depends on the convective flux Φ_F which is a function of $\vec{U}_F$. However, when linearised w.r.t. Φ_F, $\mathcal{F}_\Phi$ is evidently block-diagonal, i.e. it only acts on the velocity component in its own direction.

  The operator is derived by defining a *face flux* Ψ_{FC}, through face F outward w.r.t. cell C, catering for both convective and diffusive fluxes and acting on its respective component of $\vec{U}_{F'}$ defined on all faces F' belonging to C. For instance, for u_F:

$$\Psi^u_{FC} = \sum_{F' \in C} n^C_{FF'} u_{F'}. \tag{5}$$

  Derivation of the $n^C_{FF'}$ coefficients, omitted here, is the core of the VE/MHFV scheme. We then impose flux conservation across each face, i.e.

$$\Psi^u_{FC^+} + \Psi^u_{FC^-} = 0 \tag{6}$$

where C^+ and C^- denote the two cells connected by face F, which yields the above mentioned hybrid convection-diffusion operator. $\mathcal{F}_\Phi$ therefore operates on a face-to-face stencil, as the one shown in Figure 2;

- the divergence operator $\mathcal{D}$ is simply derived from the Gauss divergence theorem applied over each cell, i.e.

$$\mathcal{D}_C\left(\vec{U}_F\right) = \sum_{F\in\partial C} \vec{U}_F \cdot \vec{F} \tag{7}$$

where $\vec{F} = \{F_x, F_y, F_z\}$ is the area vector of face F, outward w.r.t. cell C;

- the gradient operator $\mathcal{G}$ acting on the pressure is, for a 1st-order scheme, the transpose of the divergence operator, i.e.

$$\mathcal{G}_F\left(p_C\right) = \vec{F}\left(p_{C^-} - p_{C^+}\right). \tag{8}$$

For a 2nd-order scheme, the operator also includes least-squares approximations of pressure face values:

$$\mathcal{G}_F\left(p_C\right) = \vec{F}\left\{\left[p_{C^-} + \nabla^{LSQ}_{C^-} p \cdot (\vec{x}_F - \vec{x}_{C^-})\right] - \left[p_{C^+} + \nabla^{LSQ}_{C^+} p \cdot (\vec{x}_F - \vec{x}_{C^+})\right]\right\}. \tag{9}$$

1.2 MHFV discrete adjoint

Let us now turn our attention to the discrete adjoint Navier-Stokes problem. In order to assemble the adjoint system (1) we make use of the Equational Differentiation (ED) methodology which we describe in [21]. In a nutshell, ED stands as a clear formulation of the fact that we differentiate the primal (4) itself, and not the solution algorithm used to solve it (which is what AD does). In other words, ED requires assembling (1) explicitly and independently of the primal solving algorithm. Assembly of the Jacobian matrix $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$ and the adjoint right-hand side $-\frac{\partial J}{\partial \mathbf{u}}$ may be done via any viable method, such as hand-differentiation, FD or AD tools. In our case, we make use of FD techniques combined with colouring algorithms in order to make the assembly cost independent of the problem size. One of the main advantages of such approach is that it allows to assemble the system in an automatic, non-intrusive and black-box-like fashion. However, since one of our goals here is to devise preconditioning techniques for the adjoint system, it is paramount that we analyse its structure.

Let us consider the Jacobian $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$ of the MHFV Navier-Stokes problem, i.e. the matrix of partial derivatives of the residual vector of system (4) with respect to each flow variable. The gradient and divergence operators, $\mathcal{G}$ and $\mathcal{D}$, are linear with respect to all variables, and therefore their corresponding entries in the Jacobian correspond to the operators themselves. The convection-diffusion operator $\mathcal{F}_\Phi$, however, is non-linear with respect to $\mathbf{U}_F$: this is mainly due to its dependency on the convective flux Φ_F, although further non-linearities may be also be present if the operator includes certain solution-dependent discretisation schemes (flux limiters, weighting coefficients...), which is notably the case for our Upwind Least Squares (ULSQR) stabilisation technique, often used in our tests. The full Jacobian $\mathcal{A}$ takes the form:

$$\mathcal{A} = \begin{bmatrix} \widetilde{\mathcal{F}_{xx}} & \widetilde{\mathcal{F}_{xy}} & \widetilde{\mathcal{F}_{xz}} & \mathcal{G}_x \\ \widetilde{\mathcal{F}_{yx}} & \widetilde{\mathcal{F}_{yy}} & \widetilde{\mathcal{F}_{yz}} & \mathcal{G}_y \\ \widetilde{\mathcal{F}_{zx}} & \widetilde{\mathcal{F}_{zy}} & \widetilde{\mathcal{F}_{zz}} & \mathcal{G}_z \\ \mathcal{D}_x & \mathcal{D}_y & \mathcal{D}_z & 0 \end{bmatrix} \tag{10}$$

or, in compact notation:

$$\mathcal{A} = \begin{bmatrix} \widetilde{\mathcal{F}} & \mathcal{G} \\ \mathcal{D} & 0 \end{bmatrix}. \tag{11}$$

Unlike its primal counterpart, the adjoint momentum block $\widetilde{\mathcal{F}}$ is not block-diagonal, due to the fact that convective fluxes Φ_F depend on all components of $\vec{U}_F$, and therefore so does the momentum residual, regardless of its specific direction. Each entry $\widetilde{\mathcal{F}_{ij}}$ in (10) corresponds to the partial derivative of the momentum residual in the i^{th} direction with respect to the j^{th} velocity component. Defining now $\mathbf{g}_F^* = -\frac{\partial J}{\partial \mathbf{U}_F}$ and $\mathbf{g}_C^* = -\frac{\partial J}{\partial \mathbf{p}_C}$, J being the cost function, we can assemble the full adjoint system:

$$\begin{bmatrix} \widetilde{\mathcal{F}}^T & \mathcal{D}^T \\ \mathcal{G}^T & 0 \end{bmatrix} \begin{pmatrix} \mathbf{U}_F^* \\ \mathbf{p}_C^* \end{pmatrix} = \begin{pmatrix} \mathbf{g}_F^* \\ \mathbf{g}_C^* \end{pmatrix}, \tag{12}$$

where $\mathbf{U}_F^*$ and $\mathbf{p}_C^*$ are the adjoint velocity and pressure fields, respectively. The system is evidently an Oseen-type saddle-point problem, much like the primal it is derived from.

2 PRECONDITIONING OF OSEEN-TYPE PROBLEMS

In the past few decades, constant increases in computing power have driven the advances in algorithm development, specifically towards block preconditioners for Oseen-type systems more efficient than the traditional SIMPLE-like strategies. Research has successfully produced a number of alternative algorithms, although mostly restricted so far to the FE community: variants of classical ILU preconditioners [28]; those based on the so-called *approximate commutators* [9] - in particular the *Pressure Convection-Diffusion* (PCD) commutator [15]; the *Augmented Lagrangian* approach [1]. Several interesting comparisons amongst various Navier-Stokes preconditioners have also been published [23, 24].

We showed above how our discrete Navier-Stokes problem, both primal and adjoint, indeed takes the form of a saddle-point Oseen-type problem. The main difference is that the primal is non-linear due to the contributions of convective terms to $\mathcal{F}_\Phi$, which depend on $\mathbf{U}_F$, and therefore requires an outer iterative procedure (*Picard iteration*) on top of the inner one that solves the linearised Oseen problem itself. However, since there is no interest in obtaining the exact Oseen solution at each Picard iteration, the two are typically performed at the same time in a one-shot fashion: this gives rise to the well-known CFD solution algorithms, which are in fact the mere combination of preconditioners for linearised Oseen-type systems with outer non-linear iterations. The adjoint is simply a linear saddle-point problem; it is therefore perfectly legitimate to devise generic algorithms that can be adapted to both primal and adjoint, provided that a few subtle differences are taken care of. For this reason, in the following sections we will first describe preconditioners for a generic Oseen system written in the following notation:

$$\begin{bmatrix} \mathbb{F} & \mathbb{G} \\ \mathbb{D} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{U} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_u \\ \mathbf{g}_p \end{pmatrix}, \tag{13}$$

then adapt them to the MHFV framework and highlight the differences between primal and adjoint.

Notice that we set the lower right matrix block in (13) to zero, which is the case for our Navier-Stokes scheme (4) and for its adjoint (12). This is typically true for FE schemes satisfying the so-called inf-sup stability condition, but false in general - most notably for classical FV using Rhie-Chow interpolation, as well as for many common stabilised FE schemes.

In the following sections we cover some of the algorithms we implemented so far, which are mostly MHFV adaptations of preconditioners developed for traditional FV or FE schemes. It is worth mentioning that, although our main goals are a) to achieve a better converged primal in order to produce a more robust discrete adjoint and b) to devise efficient ways of solving the adjoint itself, the potential benefits of our investigation go beyond the scope of adjoint-based optimisation, since the ability to solve the discrete Navier-Stokes to higher accuracy and in as few iterations as possible is highly desirable in the industry, regardless of adjoint computation.

2.1 SIMPLEC

Classical FV often make use of the ever-popular SIMPLE-like solution algorithms (see e.g. [25]). The efficiency of SIMPLE-like preconditioners is debatable at best: they are somewhat stable but they exhibit a rather poor convergence rate, they are prone to stagnation and their performance is affected by mesh refinement. They mostly owe their popularity to legacy reasons, being amongst the first devised working methods, and to their segregated nature, as they require solving linear systems that are relatively small and better conditioned in comparison to the full Oseen.

Despite traditionally being presented as *segregated algorithms*, highlighting the fact that they solve separately for velocity and pressure, SIMPLE-like strategies can in fact be seen as a way of preconditioning the discrete Oseen problem [24]. Notice that the generic Oseen matrix in (13) can be factorized as

$$\begin{bmatrix} \mathbb{F} & \mathbb{G} \\ \mathbb{D} & 0 \end{bmatrix} = \begin{bmatrix} \mathbb{F} & 0 \\ \mathbb{D} & -\mathbb{S} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbb{F}^{-1}\mathbb{G} \\ 0 & \mathbb{I} \end{bmatrix}, \tag{14}$$

where $\mathbb{I}$ is the identity matrix and $\mathbb{S}$ is known as *Schur complement*:

$$\mathbb{S} = \mathbb{D}\mathbb{F}^{-1}\mathbb{G}. \tag{15}$$

This suggests a potentially very efficient way of preconditioning the Oseen system; in fact, an exact Schur complement would provide an exact preconditioner, i.e. it would allow us to solve the linearised (13) in one iteration only. However, this would require inverting operator $\mathbb{F}$, which would be computationally extremely expensive in real-life engineering applications. Hence, practicality dictates that we compute an approximate Schur complement instead:

$$\widehat{\mathbb{S}} = \mathbb{D}\widehat{\mathbb{F}^{-1}}\mathbb{G} \tag{16}$$

and solve iteratively (relaxing if necessary) as follows:

1. solve $\mathbb{F}\mathbf{U}^{n+1/2} = \mathbf{g}_u - \mathbb{G}\mathbf{p}^n$ (*predictor step* for velocity);
2. solve $\widehat{\mathbb{S}}\delta\mathbf{p} = \mathbb{D}\mathbf{U}^{n+1/2} - \mathbf{g}_p$ (*pseudo-Laplacian* for pressure correction);
3. update pressure: $\mathbf{p}^{n+1} = \mathbf{p}^n + \delta\mathbf{p}$;
4. update velocity: $\mathbf{U}^{n+1} = \mathbf{U}^{n+1/2} - \widehat{\mathbb{F}^{-1}}\mathbb{G}\delta\mathbf{p}$ (*corrector step*);
5. if non-linear, update operator $\mathbb{F}$ (Picard step).

If $\mathbb{F}$ happens to be block-diagonal, as is the case for the primal Navier-Stokes system, then step 1 can be split into three separate linear solves, each corresponding to the momentum equation in its respective spatial dimension.

In its most basic implementation, SIMPLE approximates the inverse of $\mathbb{F}$ with the inverse of its main diagonal:

$$\mathbb{F}^{-1} \approx \widehat{\mathbb{F}^{-1}} = \left(\text{DIAG}\left(\mathbb{F}\right)\right)^{-1}. \tag{17}$$

In this paper, we focus on the variant of SIMPLE known as SIMPLEC. It operates by adding to the momentum equations in (13) some form of implicit relaxation by factor α:

$$\mathbb{F}_{\alpha} = \mathbb{F} + \alpha \text{DIAG}\left(\mathbb{F}\right) \tag{18}$$

and subsequently approximating the inverse of $\mathbb{F}$ as:

$$\mathbb{F}^{-1} \approx \widehat{\mathbb{F}^{-1}} = \frac{1}{\alpha}\left(\text{DIAG}\left(\mathbb{F}\right)\right)^{-1}. \tag{19}$$

It has been observed [25] that SIMPLEC, in the steady-state case, yields a pseudo-Laplacian pressure equation aimed at correcting the velocity increment $\left(\mathbf{U}^{n+1/2} - \mathbf{U}^{n}\right)$ rather than the velocity itself, and since it acts on relaxed velocity increments, it does not require relaxing the pressure correction step.

We previously adapted SIMPLEC to the MHFV framework. The main difference with respect to classical FV is that, when relaxing our operator $\mathcal{F}_{\Phi}$, instead of using its diagonal coefficients as in (18), we apply *inertial relaxation* in the form:

$$\mathcal{F}_{\Phi,\alpha} = \mathcal{F}_{\Phi} + \alpha\left[\text{diag}\left(\beta_F\right)\right] \tag{20}$$

where β_F is a suitable scaling factor defined for each face, related to the hybridisation procedure (i.e. the elimination of cell-averaged velocity components in the convection-diffusion operator) and the flux conservation (6) we impose at each face. Our inertial relaxation is proportional to the local Reynolds number Re_F, meaning that stronger relaxation is applied in areas where convection-to-diffusion ratio is higher. The explicit definition of β_F, omitted here, can be found in [19].

For the primal, the overall iterative algorithm is analogous to the generic one outlined above, the only difference being that, since we are dealing with incompressible flow, there is no source term for the continuity equation.

Primal SIMPLEC:

1. solve relaxed momentum:

$$\mathcal{F}_{\Phi^n,\alpha}\mathbf{U}_F^{n+1/2} = \mathbf{g}_F - \mathcal{G}\mathbf{p}_C^n + \alpha\left[\text{diag}\left(\beta_F\right)\right]\mathbf{U}_F^n; \tag{21}$$

2. solve pseudo-Laplacian:

$$\mathcal{D}\left[\text{diag}\left(\frac{1}{\alpha\beta_F}\right)\right]\mathcal{D}^T\delta\mathbf{p}_C = \mathcal{D}\mathbf{U}_F^{n+1/2}; \tag{22}$$

3. update pressure:

$$\mathbf{p}_C^{n+1} = \mathbf{p}_C^n + \delta\mathbf{p}_C; \tag{23}$$

4. update velocity:

$$\mathbf{U}_F^{n+1} = \mathbf{U}_F^{n+1/2} - \left[\text{diag}\left(\frac{1}{\alpha\beta_F}\right)\right]\mathcal{D}^T\delta\mathbf{p}_C; \tag{24}$$

5. update convective fluxes and assemble new operator $\mathcal{F}_{\Phi^{n+1},\alpha}$.

Notice that here, in the Schur complement appearing in step 2, we use the transpose of the divergence operator $\mathcal{D}^T$ rather than the gradient operator $\mathcal{G}$. As mentioned in Section 1.1, the two are identical - barring boundary conditions - when the pressure gradient term is discretised via the 1st-order scheme (8). For the 2nd-order scheme (9) that is no longer the case, meaning that by using $\mathcal{D}^T$ we introduce a further level of approximation in the Schur complement. We do so in order not to degrade the sparsity pattern of the pseudo-Laplacian, which is already rather challenging for standard linear solvers even with 1st-order connectivity. Since this approximation only affects the pressure correction step, the overall algorithm still converges.

SIMPLEC can be easily adapted to the adjoint system. The main difference is that the adjoint momentum operator $\widetilde{\mathcal{F}}^T$ is not block-diagonal; since the ability to solve separately for each velocity component is arguably one of the most attractive features of SIMPLE-like algorithms, we aim to to maintain such de-coupled nature in the adjoint version. We therefore need to find a suitable block-diagonal approximation to $\widetilde{\mathcal{F}}^T$ to be used as system matrix for the velocity prediction step, whilst treating all extra-diagonal blocks explicitly. Intuitively, we choose to use the transpose of operator $\mathcal{F}_\Phi$ itself, assembled using convective flux values Φ_F taken at the last primal iteration and therefore, presumably, converged.

As we did for the primal, we apply inertial relaxation in the form (20) to the adjoint momentum equation, which is necessary for steady-state SIMPLEC. We choose for simplicity to recycle the same scaling factor β_F from the primal, but we reserve the possibility to set a relaxation factor α independently. As for the adjoint pressure correction step, again we replace $\mathcal{G}^T$ with $\mathcal{D}$, regardless of the order of accuracy of $\mathcal{G}$, in order not to deteriorate the sparsity pattern of the Schur complement.

Adjoint SIMPLEC:

1. solve relaxed adjoint momentum:

$$\mathcal{F}_{\Phi,\alpha}^T\mathbf{U}_F^{*\,n+1/2} = \mathbf{g}_F^* - \mathcal{D}^T\mathbf{p}_C^{*\,n} - \left(\widetilde{\mathcal{F}}^T - \mathcal{F}_\Phi^T\right)\mathbf{U}_F^{*\,n} + \alpha\left[\text{diag}\left(\beta_F\right)\right]\mathbf{U}_\mathbf{F}^{*\,n}. \tag{25}$$

2. solve pseudo-Laplacian:

$$\mathcal{D}\left[\text{diag}\left(\frac{1}{\alpha\beta_F}\right)\right]\mathcal{D}^T\delta\mathbf{p}_C^* = \mathcal{G}^T\mathbf{U}_F^{*\,n+1/2} - \mathbf{g}_C^*. \tag{26}$$

3. update adjoint pressure:

$$\mathbf{p}_C^{*\,n+1} = \mathbf{p}_C^{*\,n} + \delta\mathbf{p}_C^*; \tag{27}$$

4. update adjoint velocity:

$$\mathbf{U}_F^{*\,n+1} = \mathbf{U}_F^{*\,n+1/2} - \left[\text{diag}\left(\frac{1}{\alpha\beta_F}\right)\right]\mathcal{D}^T\delta\mathbf{p}_C^*. \tag{28}$$

The term $\left(\widetilde{\mathcal{F}}^T - \mathcal{F}_\Phi^T\right) \mathbf{U}_F^{*\,n}$ on the right-hand side of (25) is the contribution due to the coupling of adjoint velocity components that we treat explicitly, and in adjoint jargon it is known as *Adjoint Transposed Convection* (ATC), a name borrowed from continuous adjoint theory.

It is also worth noticing that our adjoint SIMPLEC algorithm is very similar to what we would obtain if we were to apply AD to our solver via the so-called *Christianson's method*. The method, described in [6], operates by 1) applying reverse-mode AD to the last CFD iteration only, and 2) iterating until the adjoint states are converged. This can be interpreted as transposing the primal algorithm by using $\mathcal{F}_\Phi^T$ with a converged Φ_F for the momentum linear solves, with all off-diagonal blocks of the Jacobian - the ATC - relegated to the right-hand side.

2.2 Block-Coupled

The Block-Coupled (BCPL) solution strategy is arguably the most intuitive and straightforward way of solving a system like (13). In our framework, for the primal, it simply requires solving (4) after linearisation, i.e. with a frozen convective flux Φ_F, then update the value of Φ_F with the newly computed $\mathbf{U}_F$, re-assemble operator $\mathcal{F}_\Phi$, and iterate. In other words, BCPL proceeds from one Picard iteration to the next, without any inner Oseen iterations; it does not in general require any relaxation.

As for the adjoint system, since it is linear, a BCPL approach reduces to one linear solve with no iterations at all except for those that may be performed internally by the linear solver in use (which may or may not be iterative).

The BCPL approach entails two major drawbacks:

- it has been observed [27] that the saddle-point nature of system (4) poses a challenge for standard linear solvers, as it is difficult to precondition and, for certain types of solvers, even difficult to factorise/solve via direct methods due to the so-called *zero-block* on the main diagonal in the discrete continuity equation;

- while more traditional algorithms (such as SIMPLEC) require at each iteration solving multiple smaller linear systems, the BCPL approach requires solving the full linearised Oseen system (4) which can indeed be extremely large for industrial cases - even more so in our MHFV framework, since the number of velocity unknowns scales with the number of faces in the mesh, and not cells like in FV.

2.3 V-Coupled

We mentioned in Section 2.1 how the adjoint version of SIMPLEC solves for $\mathbf{u}_F^*$, $\mathbf{v}_F^*$ and $\mathbf{w}_F^*$ in a segregated fashion, moving the ATC to the right-hand side. Although we did not record any issues in our experiments, the ATC is known to be a troublesome term especially when treated explicitly, causing severe instabilities to the point where some researchers, such as [14, 20], resort to arbitrarily damping it, or even eliminating it completely in sensitive areas, in the hope that the final sensitivities won't be excessively affected qualitatively.

The issue is particularly evident in continuous adjoints, where the ATC appears explicitly at a PDE level, thus posing the challenge of finding a suitable discretisation as well as a stable way of treating it in the solution algorithm. However, it is reasonable to expect robustness issues in a discrete adjoint as well, especially at high Re, since high values on the right-hand side of (25) might cause instabilities, leading to divergence unless heavy relaxation is applied.

The obvious solution is to treat the ATC implicitly; we do so in our *V-Coupled* (VCPL) algorithm, which is simply a version of the adjoint SIMPLEC from Section 2.1 in which, in the

predictor step, we couple all velocity components by keeping them on the left-hand side:

$$\widetilde{\mathcal{F}}_{\alpha}^{T}\mathbf{U}_{F}^{*\,n+1/2} = \mathbf{g}_{F}^{*} - \mathcal{D}^{T}\mathbf{p}_{C}^{*\,n} + \alpha\left[\mathrm{diag}\left(\beta_{F}\right)\right]\mathbf{U}_{\mathbf{F}}^{*\,n}; \tag{29}$$

the pressure correction step (26), on the other hand, remains unchanged.

Besides tackling the above mentioned stability issues, by treating the ATC implicitly we can also expect an overall reduction on the iteration count. On the downside, the VCPL approach (29) comes with the obvious drawback of having to store larger matrices and solve larger linear systems in comparison with SIMPLEC.

Of course the VCPL approach is only relevant to the adjoint system; applying VCPL to the primal would be exactly equivalent to applying SIMPLEC, the only difference being that we would be solving at each iteration a single, larger linear system rather than breaking it down into three smaller ones.

2.4 Augmented Lagrangian

The Augmented Lagrangian (AL) preconditioning scheme for Oseen-type problems was first presented by [1], and further developed with several variants by e.g. [2, 16]. AL-based preconditioners have been so far investigated mostly within FE frameworks, and have been proven to be theoretically almost optimal [2] in terms of mesh and Re-dependency. Despite its drawbacks, we therefore deem it worth investigating the AL methodology and attempting to adapt it to our MHFV scheme.

The AL core idea is to re-write system (13) as

$$\begin{bmatrix} \mathbb{F}_{\gamma} & \mathbb{G} \\ \mathbb{D} & 0 \end{bmatrix}\begin{pmatrix} \mathbf{U} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{u,\gamma} \\ \mathbf{g}_{p} \end{pmatrix}, \tag{30}$$

where

$$\mathbb{F}_{\gamma} = \mathbb{F} + \gamma\mathbb{G}\mathbb{W}^{-1}\mathbb{D} \tag{31}$$

and

$$\mathbf{g}_{u,\gamma} = \mathbf{g}_{u} + \gamma\mathbb{G}\mathbb{W}^{-1}\mathbf{g}_{p}. \tag{32}$$

Systems (30) and (13) are clearly equivalent, since (31) and (32) simply add to the velocity blocks a term proportional to residual of the continuity equation. In other words, the AL method corresponds to adding to the momentum equations a penalisation term which is driven to zero for a converged solution. In (31) and (32), γ is a positive augmentation factor, and $\mathbb{W}$ an arbitrary symmetric positive definite (SPD) matrix.

As observed by [2], an advantage of using an AL formulation (besides theoretical near-optimality of the preconditioner itself) is that the issue of finding a good approximation for the Schur complement (15) is circumvented. If the augmentation factor is large enough, then the penalisation term will prevail on the operator $\mathbb{F}$ itself, thus justifying the approximation

$$\mathbb{F}_{\gamma}^{-1} \approx \widehat{\mathbb{F}_{\gamma}^{-1}} = \left(\gamma\mathbb{G}\mathbb{W}^{-1}\mathbb{D}\right)^{-1} \tag{33}$$

which yields an approximate Schur complement in the form:

$$\widehat{\mathbb{S}} = \mathbb{D}\left(\gamma\mathbb{G}\mathbb{W}^{-1}\mathbb{D}\right)^{-1}\mathbb{G} = \frac{1}{\gamma}\mathbb{W}. \tag{34}$$

The generic AL iterative procedure is outlined as follows:

1. solve $\mathbb{F}_\gamma \mathbf{U}^{n+1} = \mathbf{g}_{u,\gamma} - \mathbb{G}\mathbf{p}^n$ (penalised momentum equation);
2. solve $\frac{1}{\gamma}\mathbb{W}\delta\mathbf{p} = \mathbb{D}\mathbf{U}^{n+1} - \mathbf{g}_p$ (pressure correction);
3. update pressure: $\mathbf{p}^{n+1} = \mathbf{p}^n + \delta\mathbf{p}$;
4. if non-linear, assemble new augmented momentum operator $\mathbf{F}_\gamma$.

In FE, matrix $\mathbb{W}$ is often chosen to be the so-called pressure mass matrix or, for practical reasons, a diagonal approximation of it (usually a lumped mass matrix, or else simply its main diagonal); in that case, the pressure correction step 2 involving the approximate Schur complement (34) simply requires inverting a diagonal matrix, i.e. it doesn't actually involve a linear solve. The main drawbacks of AL-based preconditioners are:

- the augmented momentum operator $\mathbb{F}_\gamma$ is no longer block-diagonal, because each of the velocity components contributes to the penalisation term of the momentum equations in all spatial dimensions. Therefore, step 1 in the procedure above entails a single coupled linear solve for u, v and w at the same time and, unlike with SIMPLEC, it cannot be de-coupled into segregated smaller systems, at least for the basic AL formulation;
- as shown by [1], too large values of γ cause the penalised block $\mathbb{F}_\gamma$ to become increasingly ill-conditioned since $\mathbb{G}\mathbb{W}^{-1}\mathbb{D}$ is a singular matrix, and therefore increasingly challenging for linear solvers. On the other hand, the approximated Schur complement (34) is only close to the exact one if γ is large enough, hence if γ is chosen too small, the overall algorithm may underperform or even diverge. A trade-off between these two extremes is thus required, possibly combined with inexact solves of the augmented momentum equations.

Developing an AL preconditioner for our primal MHFV solver is fairly straight-forward. We already described in Section 1.1 the divergence operator $\mathcal{D}$ (7); we proceed by adding a penalisation term in the form (31) to our MHFV momentum operator as follows:

$$\mathcal{F}_{\Phi,\gamma} = \mathcal{F}_\Phi - \gamma\mu\mathcal{D}^T\left[\text{diag}\left(\frac{1}{|C|}\right)\right]\mathcal{D}, \tag{35}$$

whereas the right-hand side remains unchanged since there is no source term for the continuity equation. A few observations are in order:

- as we did for the SIMPLEC Schur complement, here we use $\mathcal{D}^T$ rather than $\mathcal{G}$ in the AL augmentation term. We already highlighted how the two are identical in the case of 1st-order pressure gradient scheme (8), and since a 2nd-order accurate $\mathcal{G}$ as defined in (9) would further increase the complexity of the already challenging augmented operator (35), we choose to stick with the 1st-order operator $\mathcal{D}^T$. It is important to stress that such choice does not in any way affect the order of accuracy of the solution itself: the AL algorithm ultimately drives $\mathcal{D}\mathbf{U}_F$, and therefore the penalisation term, to zero, and the final flow field satisfies the original non-augmented Navier-Stokes problem (4);
- $|C|$ is the volume of cell C, and the diagonal matrix diag $(|C|)$ plays in (35) the role of $\mathbb{W}$ in the generic formulation (31). In a FV-like framework, such as ours, this is indeed the equivalent of the FE pressure mass matrix, and since it is diagonal it can be inverted with no further approximation;

- besides the augmentation parameter γ, we also multiply the penalisation term by a scaling factor μ, in order to maintain γ within a range of values that work reasonably well regardless of the specific mesh size and problem physics. Following suggestions from [1], we set μ to scale with the velocity: $\mu = \max|\vec{U}_C|$, where $|\vec{U}_C|$ the cell-averaged velocity magnitude.

We can now outline our primal AL iterative procedure.

Primal Augmented Lagrangian:

1. solve augmented momentum:

$$\mathcal{F}_{\Phi^n,\gamma}\mathbf{U}_F^{n+1} = \mathbf{g}_F - \mathcal{G}\mathbf{p}_C^n; \tag{36}$$

2. compute pressure correction:

$$\delta\mathbf{p}_C = -\gamma\mu\left[\text{diag}\left(\frac{1}{|C|}\right)\right]\mathcal{D}\mathbf{U}_F^{n+1}; \tag{37}$$

3. update pressure:

$$\mathbf{p}_C^{n+1} = \mathbf{p}_C^n + \delta\mathbf{p}_C; \tag{38}$$

4. update convective fluxes and assemble new augmented operator $\mathcal{F}_{\Phi^{n+1},\gamma}$.

Lastly, we propose here an adaptation of the AL preconditioner to the adjoint Navier-Stokes system. Following the generic methodology above, the penalised adjoint momentum operator becomes

$$\widetilde{\mathcal{F}}_\gamma^T = \widetilde{\mathcal{F}}^T - \gamma\mu\mathcal{D}^T\left[\text{diag}\left(\frac{1}{|C|}\right)\right]\mathcal{D}, \tag{39}$$

and its right-hand side:

$$\mathbf{g}_{F,\gamma}^* = \mathbf{g}_F^* - \gamma\mu\mathcal{D}^T\left[\text{diag}\left(\frac{1}{|C|}\right)\right]\mathbf{g}_C^*. \tag{40}$$

The scaling factor μ is taken from the primal, while the penalisation coefficient γ is defined independently.

Notice that, as we did for the primal, in (39) we replaced $\mathcal{G}^T$ with $\mathcal{D}$ in the adjoint augmentation term in order to avoid excessively complex connectivities. For the adjoint, however, we need to take a few extra measures: the adjoint continuity equation dictates that we penalise by a quantity proportional to $\left(\mathcal{G}^T\mathbf{U}_F^* - \mathbf{g}_C^*\right)$ exactly; therefore, if $\mathcal{G}^T \neq \mathcal{D}$, penalising for $(\mathcal{D}\mathbf{U}_F^* - \mathbf{g}_C^*)$ as we do in (39) and (40) may never converge, as it would be the equivalent to solving the adjoint of a 1st-order accurate primal, which would be inconsistent with the actual Jacobian. A way around the issue is to treat all the (presumably small) 2nd-order contributions explicitly, as shown in the algorithm below.

Adjoint Augmented Lagrangian:

1. solve adjoint augmented momentum:

$$\widetilde{\mathcal{F}}_\gamma^T\mathbf{U}_F^{*\,n+1} = \mathbf{g}_{F,\gamma}^* - \mathcal{D}^T\mathbf{p}_C^{*\,n} - \gamma\mu\mathcal{D}^T\left[\text{diag}\left(\frac{1}{|C|}\right)\right]\left(\mathcal{G}^T - \mathcal{D}\right)\mathbf{U}_F^{*\,n}; \tag{41}$$

2. compute adjoint pressure correction:

$$\delta \mathbf{p}_C^* = -\gamma\mu \left[\text{diag}\left(\frac{1}{|C|} \right) \right] \left(\mathcal{G}^T \mathbf{U}_{\mathbf{F}}^{*\,n+1} - \mathbf{g}_C^* \right); \tag{42}$$

3. update adjoint pressure:

$$\mathbf{p}_C^{*\,n+1} = \mathbf{p}_C^{*\,n} + \delta \mathbf{p}_C^*. \tag{43}$$

3 NUMERICAL RESULTS

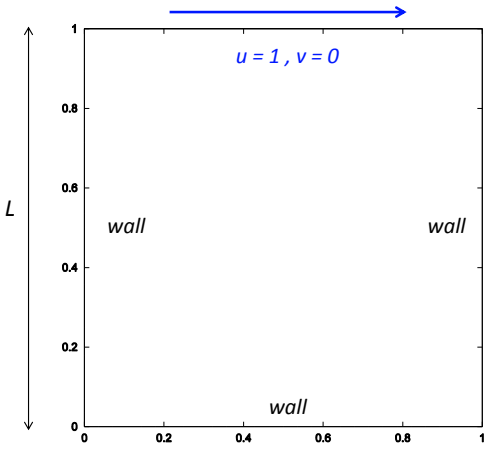


Figure 3: Lid-driven cavity test case.

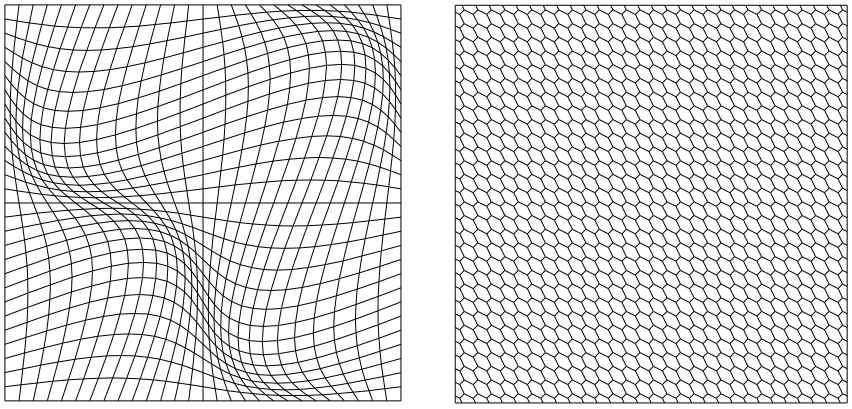


Figure 4: Mesh types used for the lid-driven cavity test case.

For a first evaluation of the performance of our (primal) algorithms, we test them on the well-known 2D lid-driven cavity benchmark test case, set-up as shown in Figure 3. We run it at two different Reynolds numbers, $Re = 10^2$ and $Re = 10^3$, on the two different mesh types shown

in Figure 4, with Type A being a quadrilateral, highly distorted/non-orthogonal mesh, and Type B a more regular, polygonal honeycomb-like mesh. We test on a series of progressively refined meshes of both types in order to assess h-dependency.

As for the discretisation schemes, we use 2^{nd}-order MHFV operators for both velocity and pressure, the first stabilised via our ULSQR technique described in [19], the second as in (9). All simulations are run down to a tolerance of 10^{-4} on scaled residuals.

3.1 Block-Coupled

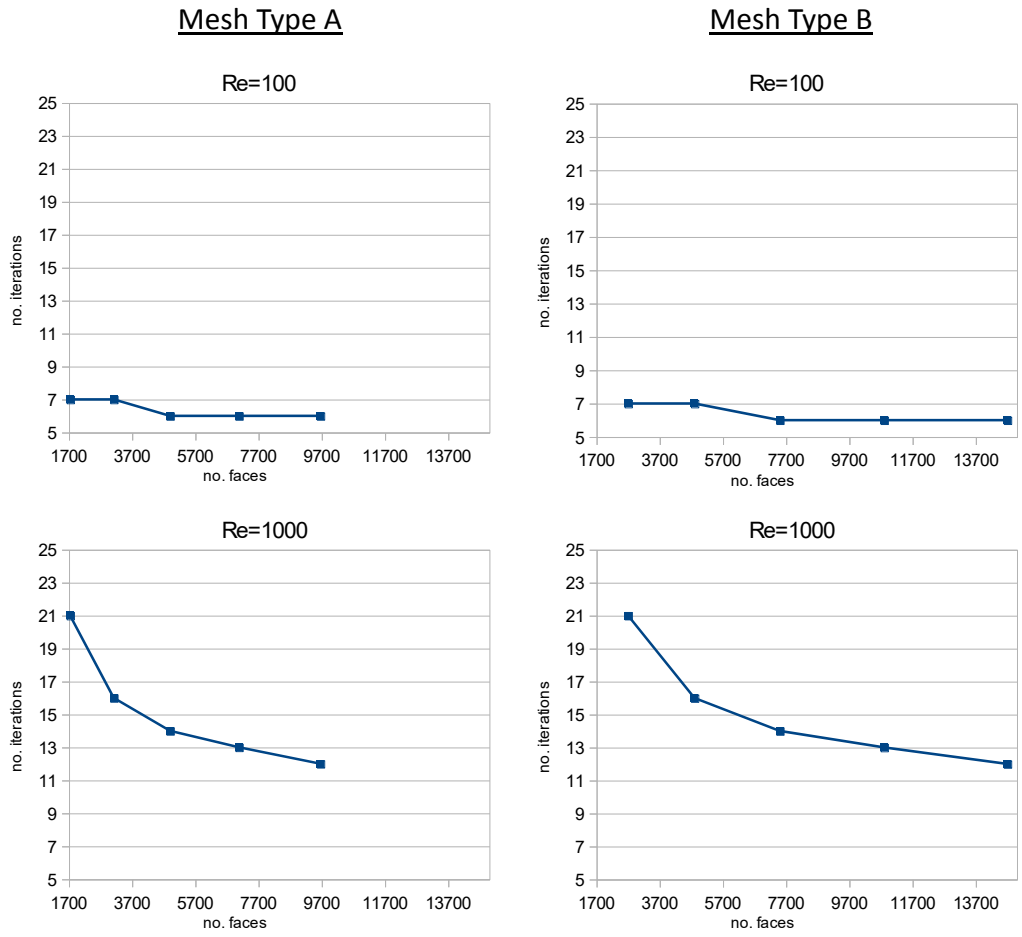


Figure 5: MHFV Block-Coupled. Lid-driven cavity test case. Iteration count in function of mesh type, mesh size and Re.

Results for BCPL are shown in Figure 5 where we plot, for each mesh type and each Re, the iteration count n_{iter} against the number of mesh faces, i.e. against mesh refinement.

The figures are rather positive: at $Re = 10^2$ the iteration count is consistently small (below 10), and does not appear to be affected by mesh size or quality; at $Re = 10^3$ we record an asymptotic behaviour of n_{iter} with respect to mesh size, namely the iteration count is higher on coarser meshes and tends to settle around a value in the order of 10 as we refine the mesh; such behaviour is identical over both mesh types. Evidence thus suggests that our BCPL algorithm is independent of mesh quality and mesh size (barring coarse mesh cases, where it underperforms

slightly), and only marginally affected by an increase of Re.

3.2 SIMPLEC

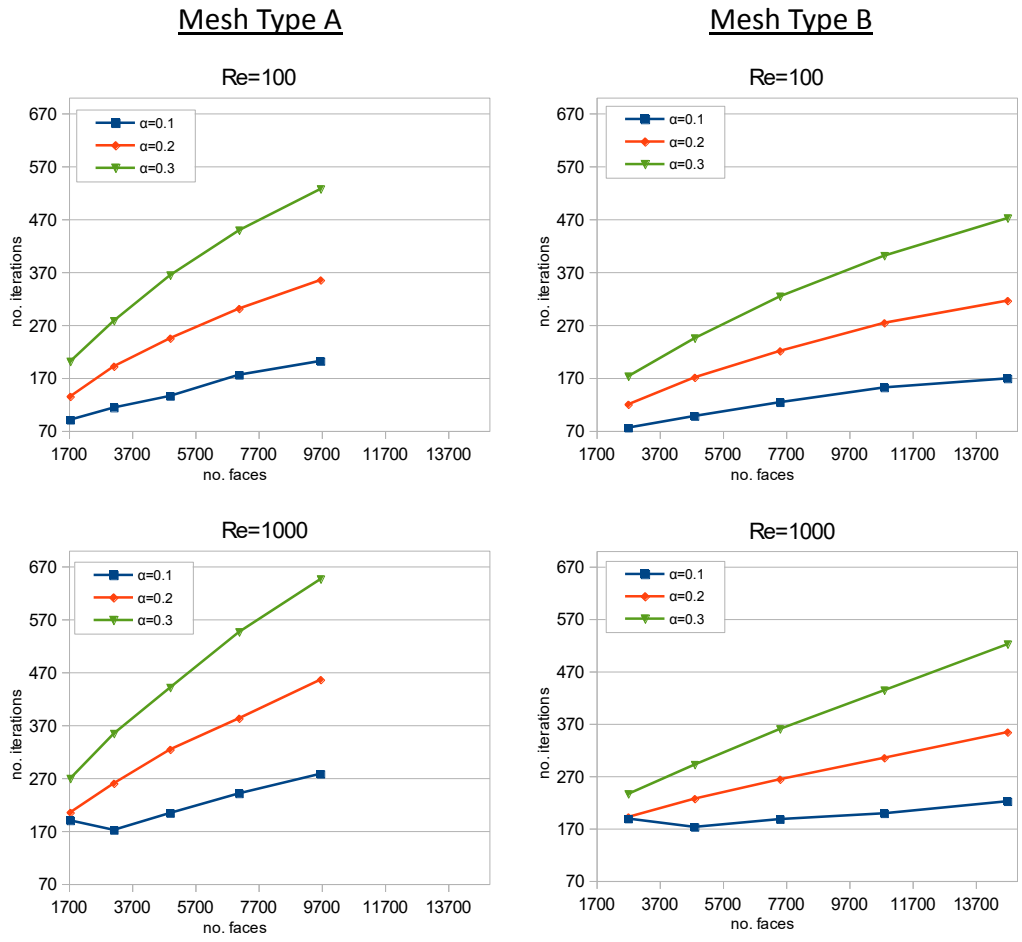


Figure 6: MHFV SIMPLEC. Lid-driven cavity test case. Iteration count in function of mesh type, mesh size, Re and relaxation factor α.

We perform on our SIMPLEC scheme the same tests presented above for BCPL. For SIMPLEC we also try out different values of relaxation factor α, in order to verify whether the optimal value depends on the other parameters.

The resulting graphs are reported in Figure 6. It appears that, for both mesh types and both Re, setting $\alpha = 0.1$ gives the best performance in terms of both iteration count and algorithm scaling against problem size. Results are encouraging in that sense, as that they suggest that the optimal relaxation factor does not depend on mesh quality, and it is only slightly influenced by grid coarseness and problem physics.

On the other hand, these results also highlight two severe limitations of SIMPLEC itself:

- the iteration count scales with the problem size, i.e. refining the mesh causes n_{iter} to grow. The iteration count itself is in general very high, roughly an order of magnitude greater than with the BCPL approach;

- the algorithm heavily depends on numerical parameters, and in particular on mesh type. Notice how n_{iter} is in general much higher, and grows more rapidly with the problem size, for mesh Type A. This can be attributed to the irregularity of Type A, which features several strongly distorted cells as well as a wide range of cell volumes/face areas - whereas Type B is only slightly non-orthogonal, and perfectly regular in terms of element size. This might be causing SIMPLEC to underperform on Type A compared to Type B, despite the considerably smaller problem size.

3.3 Augmented Lagrangian

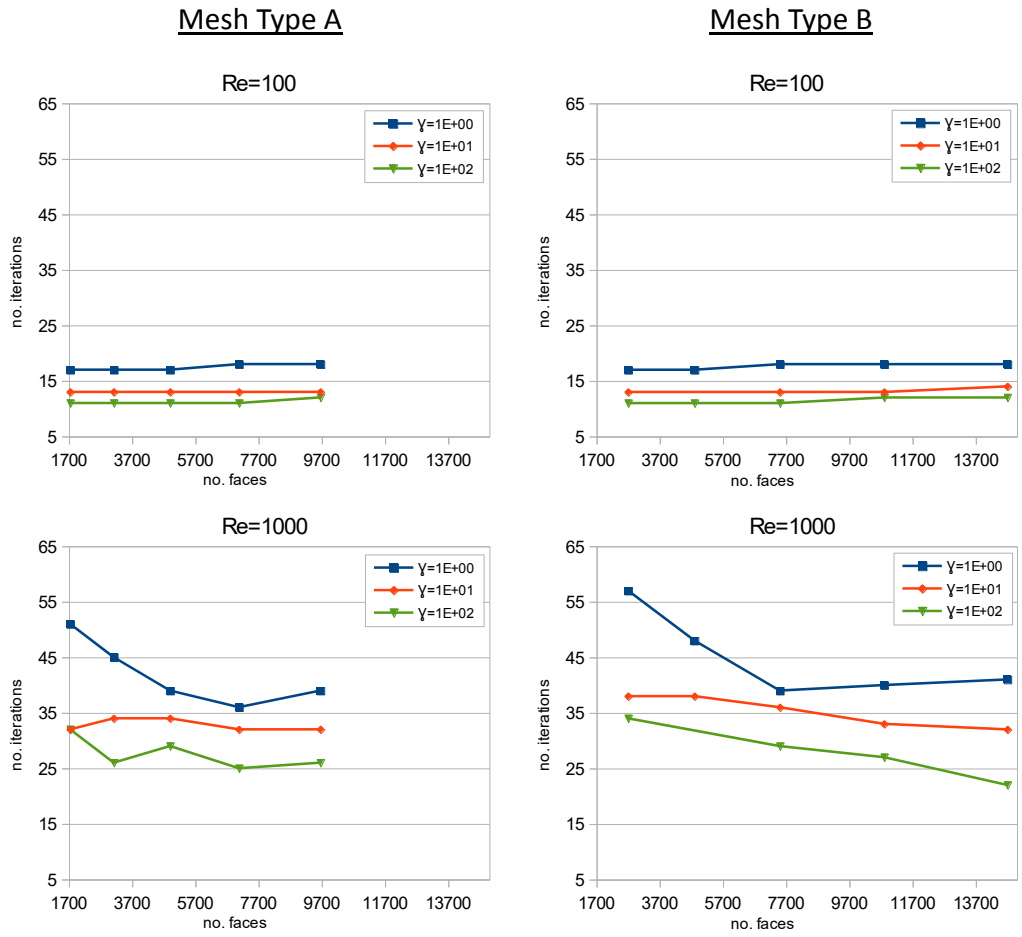


Figure 7: MHFV Augmented Lagrangian. Lid-driven cavity test case. Iteration count in function of mesh type, mesh size, Re and augmentation factor γ.

Once again we run the lid-driven test case to validate the performance of AL in function of Re, mesh quality and mesh size; we also test for different values of penalisation factor γ. Results are reported in Figure 7. The curves show that there is no definite correlation between n_{iter} and grid size nor grid quality, hence our AL implementation is completely mesh independent. Similarly to the other preconditioners, we observe however a slight increase in iteration count for higher Re values. The value of n_{iter} itself remains in the order of 10, hence much lower than that of SIMPLEC and fairly close to BCPL results, thus confirming the near-optimality of

the AL algorithm.

Results also highlight how higher values of γ consistently reduce the total number of iterations; this is expected because, when γ is high, the convection-diffusion operator $\mathcal{F}_\Phi$ in (35) becomes negligible compared to the penalisation term, and therefore the diagonal approximate Schur complement used in (37) is close to the exact one. However, as observed in Section 2.4, too high values of γ will cause the augmented operator $\mathcal{F}_{\Phi,\gamma}$ to be nearly singular, thus hindering the linear solve for velocity prediction. Keeping γ in the order of 1 appears to be a resonable choice; in order to run tests at higher γ values, we circumvent the linear solver issue by using direct solvers, which is however not a viable option for real industrial cases.

3.4 Comparison of primal Navier-Stokes solution algorithms

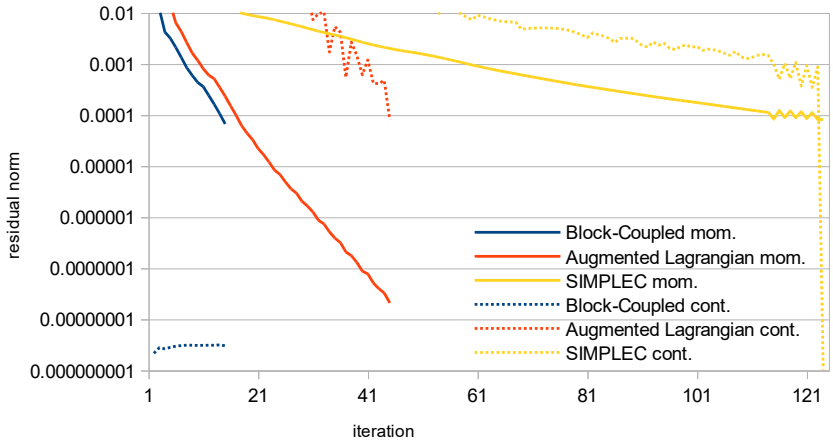
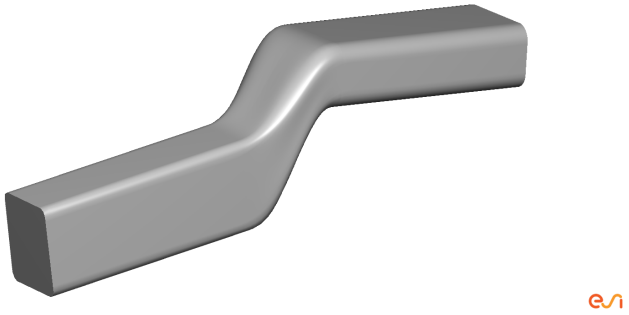


Figure 8: S-bend test case. Convergence history of the x-momentum and continuity scaled residual norms for different solution algorithms.

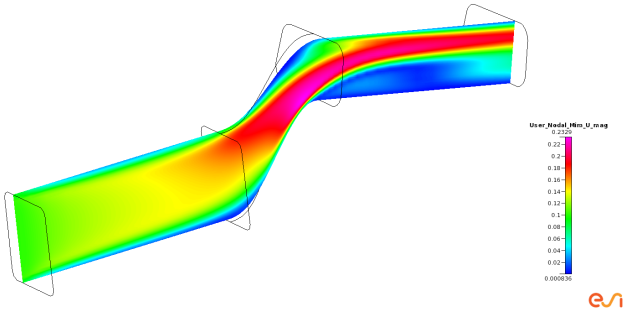
We run a further comparison of the primal BCPL, SIMPLEC and AL algorithms on a 3D flow solve: the S-bend, a popular benchmark test case involving an internal flow simulation through an S-shaped air duct. We run the S-bend at a fairly low $Re \approx 400$ and over a relatively coarse, hexahedral mesh (approximately 41k elements and 126k faces). We show in Figure 9 the S-bend geometry and some solution contours.

For the current test, we set the following algorithm parameters: no relaxation for BCPL; $\alpha = 0.1$ for SIMPLEC (experimentally determined to be close to optimal); $\gamma = 5$ for AL (which we found to produce good overall convergence properties whilst maintaining the augmented system relatively well-conditioned). Tolerance is set to 10^{-4} for both momentum and continuity equations. As for the numerical scheme, we set our MHFV solver to 2^{nd}-order accuracy for both pressure and velocity, the latter stabilised via the previously mentioned ULSQR strategy.

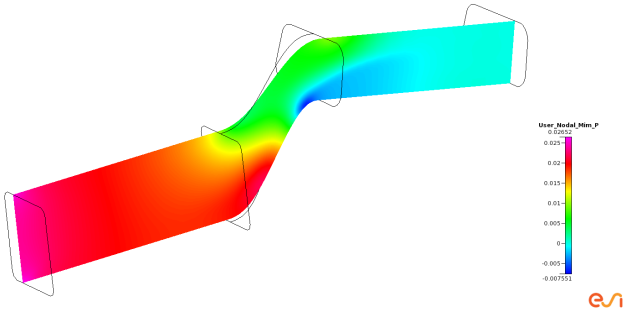
A comparison is shown in Figure 8, where we report the history of the x-momentum and continuity scaled residuals for each preconditioner. The graph demonstrates once again the definite superiority of both BCPL and AL over SIMPLEC in terms of convergence rate. The reader may also notice a stagnation-like behaviour of SIMPLEC at the last few iterations, also noticeable in other graphs throughout this paper. This is due to the fact that our CFD solver only performs a linear solve if the initial residual norm for the specific equation being solved is above the global tolerance; not all segregated equations reach convergence at the exact same



(a)



(b)



(c)

Figure 9: S-bend test case. Domain geometry (a), velocity magnitude (b), pressure field (c).

iteration, hence a synchronisation phase happens at the end of the process, essentially causing the equations to be solved alternately until all residual norms align just below tolerance.

It is remarkable how BCPL and AL exhibit very similar convergence properties. The difference is that AL runs several extra iterations, due to the fact that the continuity equation does not converge at the same rate for the two: BCPL solves it exactly at each step, while AL iteratively reduces the continuity residual via the penalisation mechanism. The extra AL iterations evidently result in a better converged momentum equation, at the expense of continuity being solved only down the set global tolerance, whilst BCPL solves it exactly - or almost, depending on the linear solver in use.

3.5 Comparison of adjoint Navier-Stokes solution algorithms

To test and compare performances of our adjoint preconditioners, we first run a simulation on the simple 2D square box test case represented in Figure 10, a popular benchmark case

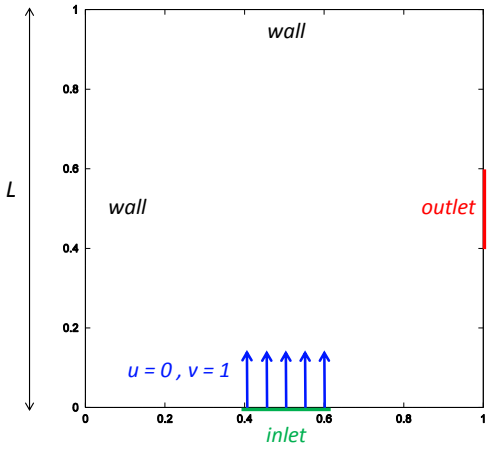


Figure 10: Square box test case.

often used to illustrate adjoint-based optimisation techniques; the cost function J, necessary to compute the adjoint right-hand side, is defined as the total pressure drop across the box's boundary $\partial\Omega$:

$$J = -\int_{\partial\Omega} \left(p + \frac{1}{2}|\vec{U}|^2 \right) \tag{44}$$

We run the primal at $Re \approx 10^3$ on a regular polygonal mesh (Type B in Figure 4); MHFV operators are all 2$^{\text{nd}}$-order accurate. We converge the primal down to a normalised resudual of 10^{-6} in order to secure a robust discrete adjoint; for the adjoint itself, we set a tolerance of 10^{-4}. We relax by $\alpha = 0.3$ both SIMPLEC and VCPL, while for AL we use $\gamma = 10$. We report

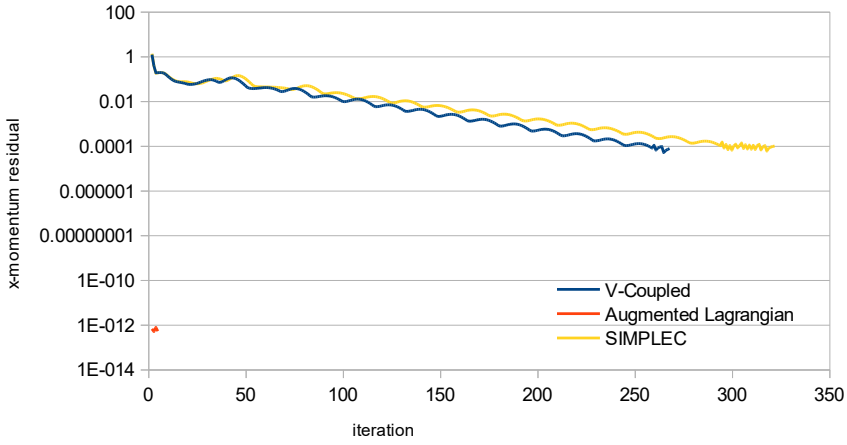


Figure 11: Square box test case. Convergence history of the adjoint x-momentum scaled residual norm for different solution algorithms.

in Figure 11 the convergence history for the adjoint x-momentum. SIMPLEC and VCPL both exhibit a somewhat oscillatory behaviour, and both converge roughly at the same rate. The gain in iteration count obtained by opting for VCPL over SIMPLEC is moderate at best; a series of other test cases, not reported here, also confirm that the reduction on n_{iter}, when present,

is limited to a maximum of about 10%, at the considerable cost of having to solve a velocity coupled system at each iteration. It should also be mentioned, however, that we mostly test on fairly low-Re; we therefore maintain the argument that, for higher Re simulations, treating the ATC implicitly may bring about significant advantages in terms of solver stability.

AL, on the other hand, outperforms by far the SIMPLE-like preconditioners. For this particular test case the adjoint momentum residual falls well below tolerance - in fact, down to machine precision - from the very first iteration, whilst the adjoint continuity equation, not shown here, converges to the given tolerance in 4 iterations only. Therefore, as for the primal, AL displays a behaviour very close to the optimal BCPL approach - the adjoint system is linear, hence a fully implicit adjoint BCPL approach does not need any outer iterations.

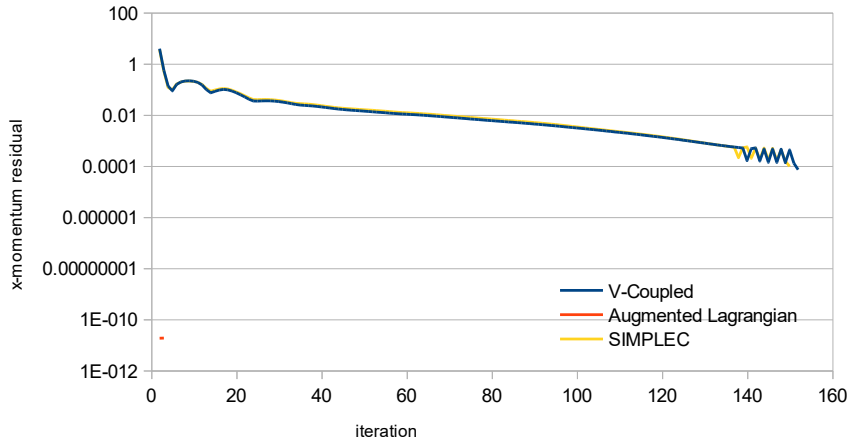
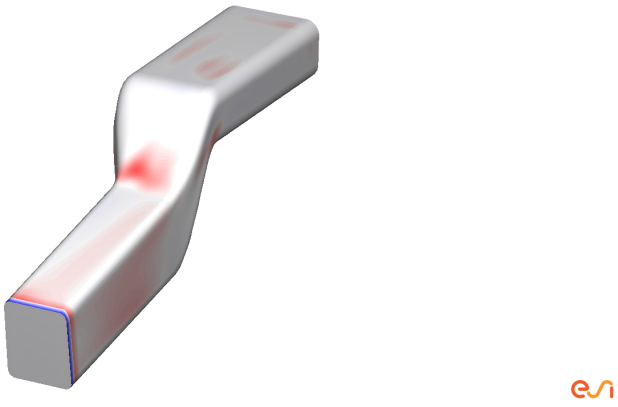


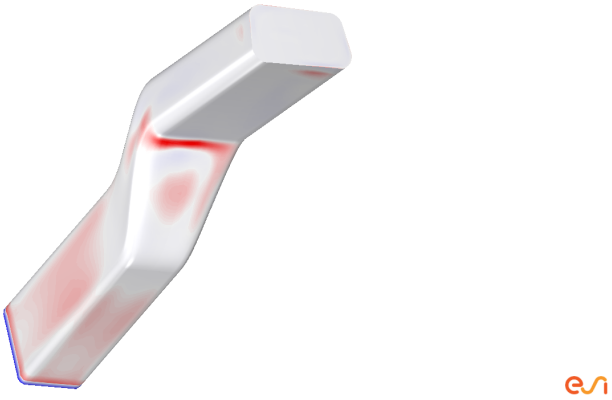
Figure 12: S-bend test case. Convergence history of the adjoint x-momentum scaled residual norm for different solution algorithms.

To further validate, we also assemble and solve the discrete adjoint of the 3D S-bend test case whose primal we described in section 3.4; again we define as cost function the total pressure drop (44). We use an optimal relaxation factor, found to be $\alpha = 0.28$, for both SIMPLEC and VCPL, and an augmentation factor $\gamma = 10$ for AL. Results are reported in Figure 12; we also show, for the sake of generating reader interest, the computed surface sensitivity field in Figure 13.

Results on the S-bend confirm the observations previously made on the 2D square box: SIMPLEC and VCPL perform comparably (in fact, to be precise, for the S-bend VCPL even takes a few more iterations than SIMPLEC due to the final residual synchronisation process explained in Section 3.4), whilst AL drops the adjoint momentum residual down to near-machine precision from the very start, and only takes 3 iterations to converge the adjoint continuity below tolerance. It should be mentioned, however, that all of the drawbacks highlighted for the primal AL also affect the adjoint AL, namely: on one hand, the difficulties in solving the augmented momentum other than via direct methods for too large values of γ; on the other hand, the failure of the overall AL algorithm for a γ too small.



(a)



(b)

Figure 13: S-bend test case. Surface sensitivity field: red areas indicate zones where pulling the surface outwards results in total pressure loss reduction.

4 CONCLUSIONS

Despite our MHFV discretisation scheme being an alternative, non-standard form of the discrete Navier-Stokes equations, numerical results shown in this paper reveal how the adaptation of preconditioning techniques developed for classical Finite Volumes or Finite Elements schemes leads to algorithms exhibiting behaviours similar to those described in traditional literature. SIMPLE-type schemes suffer from poor convergence rates and excessive dependency on mesh quality and size; alternative schemes, such as BCPL and AL, although fully mesh-independent and theoretically near-optimal, are affected by several practical issues, notably caused by the size and/or complexity of the linear solves involved. The same applies when such algorithms are run for the adjoint Navier-Stokes system.

Convergence properties and robustness aside, there is one other aspect, paramount in industrial contexts, which we haven't yet delved into: the cost in terms of CPU time. Our tests show that SIMPLEC benefits from a lower CPU time per iteration, as it involves solving smaller, better conditioned systems. On the other hand, the CPU time per iteration is found to scale similarly for all algorithms, and since the BCPL/AL iteration count is an order of magnitude lower than for SIMPLEC, the former two remain by far the best choice.

Interestingly enough, an AL iteration appears to be roughly twice as expensive as a BCPL one, despite AL solving for smaller matrices; this is easily explained by the fact that AL matri-

ces, although smaller, are also less sparse: the face-to-face connectivity (Figure 2) repeated on the off-diagonal blocks of the augmented momentum operator easily gives rise to AL matrices with more non-zeroes than the full Oseen matrix itself. Of course the issue - which is scheme-dependent and may be less aggravating for traditional FV methods - could be circumvented via a more efficient preconditioning of the AL linear solves but, as we mentioned, this can prove to be a very challenging task and beyond the scope of our current research.

In the near future we therefore plan to implement a MHFV version of the *Modified Augmented Lagrangian* (MAL) described in [2], which is indeed designed to alleviate some of the issues discussed above whilst still benefiting from most advantages brought about by AL. We also plan to conduct further research related to Navier-Stokes preconditioning in general, namely by investigating approximate commutators such as the *Pressure Convection-Diffusion* (PCD) [15] and the *Least Squares approximate commutator* [10].

ACKNOWLEDGEMENTS

This work was funded by the EU through the FP7-PEOPLE-2012-ITN "AboutFlow" Grant agreement number 317006.

REFERENCES

[1] M. Benzi, M. A. Olshanskii, An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, **28**, 2095–2113, 2006.

[2] M. Benzi, M. A. Olshanskii, Z. Wang, Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, **66**(4), 486–508, 2011.

[3] F. Brezzi, K. Lipnikov, M. Shashkov, A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, **15**(10), 1533–1551, 2005.

[4] F. Brezzi, K. Lipnikov, M. Shashkov, Convergence of the Mimetic Finite Difference method for diffusion problems on polyhedral meshes. *SIAM Journal on Numerical Analysis*, **43**(5), 1872–1896, 2005.

[5] F. Brezzi, R.S. Falk, L.D. Marini, Basic principles of mixed virtual element method. *Mathematical Modeling and Numerical Analysis*, **48**(4), 1227–1240, 2014.

[6] B. Christianson, Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, **3**, 311–326, 1994.

[7] J. Droniou, Remarks on discretizations of convection terms in hybrid mimetic mixed methods. *Networks and Heterogeneous Media*, **5**(3), 545–563.

[8] J. Droniou, R. Eymard, Study of the mixed finite volume method for Stokes and Navier-Stokes equations. *Numerical Methods for Partial Differential Equations*, **25**(1)., 137–171, 2008.

[9] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, Block preconditioners based on approximate commutators. *SIAM Journal of Scientific Computing*, **27**(5), 1651–1668, 2006.

[10] H. Elman, V. E. Howle, J. Shadid, D. Silvester, R. Tuminaro, Least squares preconditioners for stabilized discretizations of the NavierStokes equations. *SIAM Journal of Scientific Computing*, **30**(1), 290–311, 2007.

[11] A. Griewank, A. Walther, Evaluating derivatives: principles and techniques of Algorithmic Differentiation. *SIAM, second edition*, 2008.

[12] A. Jameson, Aerodynamic design via control theory. *Journal of Scientific Computing*, **3**, 233–260, 1988.

[13] A. Jameson, Optimum aerodynamic design using CFD and control theory. *AIAA Paper* **95-1729**, 1995.

[14] G. Karpouzas, E.M. Papoutis-Kiachagias, T. Schumacher, E. de Villiers, K.C. Giannakoglou, C. Othmer, Adjoint optimization for vehicle external aerodynamics. *International Journal of Automotive Engineering*, **7**, 2016.

[15] D. Kay, D. Loghin, A. Wathen, A Preconditioner for the Steady-State Navier-Stokes Equations. *SIAM Journal of Scientific Computing*, **24**(1), 237–256, 2002.

[16] M. A. Olshanskii, M. Benzi, An augmented Lagrangian approach to linearized problems in hydrodynamic stability. *SIAM Journal on Scientific Computing*, **30**, 1459–1473, 2008.

[17] M. Oriani, G. Pierrot, Alleviating adjoint solver robustness issues via mimetic CFD discretization scheme. *OPT-I: International Conference on Engineering and Applied Sciences Optimization*, Kos (Greece), 4-6 June, 2014.

[18] M. Oriani, G. Pierrot, Towards converged adjoint state for large industrial cases by improving the discretization schemes. *WCCM XIECCM VECFD VIIACM-ECCOMAS*, Barcelona (Spain), July 20-25, 2014.

[19] M. Oriani, G. Pierrot, A Mixed Hybrid Finite Volume scheme for incompressible Navier-Stokes. *NAFEMS World Congress*, San Diego (California), 21-24 June, 2015.

[20] C. Othmer, Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, **4**(6), 2014.

[21] G. Pierrot, Equational Differentiation of incompressible flow solvers as a middle ground between continuous and discrete adjoint methodologies. *WCCM XIECCM VECFD VIIACM-ECCOMAS*, Barcelona (Spain), July 20-25, 2014.

[22] O. Pironneau, On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, **64**, 97–110, 1974.

[23] M. Rehman, C. Vuik, G. Segal, A comparison of preconditioners for incompressible Navier-Stokes solvers. *International Journal for Numerical Methods in Fluids*, **57**(12), 1731–1751, 2008.

[24] A. Segal, M. Rehman, C. Vuik, Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics: Theory, Methods and Applications*, **3**(3), 245–275, 2010.

[25] H. Veersteg, W. Malalasekera, An introduction to Computational Fluid Dynamics: the Finite Volume method. *Pearson Education, Harlow*, 2007.

[26] L.B. Da Veiga, J. Droniou, G. Manzini, A unified approach to handle convection terms in Finite Volumes and Mimetic Discretization Methods for elliptic problems. *IMA Journal of Numerical Analysis*, **31**(4), 1357–1401, 2011.

[27] C. Vuik, A. Segal, Solution of the coupled Navier-Stokes equations. *TU Delft internal report* **95-28**, 1995.

[28] S. Wille, O. Staff, A.F.D. Loula, Efficient a priori pivoting schemes for a sparse direct Gaussian equation solver for the mixed finite element formulation of the Navier-Stokes equations. *Applied Mathematical Modelling*, **28**(7), 607–616, 2004.