International Journal of Mechanics and Energy (IJME)
Vol. 3, Issue 1,2015, ISSN: 2286-5845

A semi-implicit scheme for the stabilized finite element method for the incompressible Navier-Stokes equations

Bart Janssens[1],Tamás Bányai[2], Karim Limam[3], Walter Bosschaerts[1]

[1]*Royal Military Academy, Department of Mechanics. Avenue de Renaissance 30, 1000 Brussels, Belgium*

[2]*von Karman Institute for Fluid Dynamics, Chaussée de Waterloo 72, 1640 Rhode-St-Genèse, Belgium*

[3]*La Rochelle University, LaSIE, Avenue Michel Crépeau, 17042 La Rochelle Cedex 1, France*

Abstract: We apply a predictor-multicorrector scheme to thePressure Stabilized and Streamline Upwind Petrov-Galerkin (PSPG and SUPG) stabilized incompressible Navier-Stokes equations, solving separate systems for the velocity and pressure. The advective terms are treatedexplicitly. The algorithm is tested against the analytical solution for the Taylor-Green vortices case and shown to be of second order accuracy. We analyze the performance and compare with a fully coupled, implicit solution technique. The semi-implicit predictor-multicorrector scheme proves to be advantageous when small time steps are required due to the flow physics.

Keywords: Finite element method, Incompressible flow, Predictor–corrector scheme

1. Introduction

The Galerkin finite element method has attractive properties: it has second order accuracy in case of linear shape functions and a rigorous mathematical foundation, naturally supporting unstructured grids over a wide variety of element types. However, due to the central-difference type discretization, it is inherently unstable when applied to the Navier-Stokes equations. The Pressure Stabilized and Streamline Upwind Petrov-Galerkin (PSPG and SUPG) stabilizations overcome this problem while retaining second order accuracy and allowing equal order interpolation for the pressure and velocity [1].

In [2] these stabilizations are applied together with an extrapolation of the velocity to linearize the advection operator. With a Crank-Nicolson time discretization, this method is second order accurate in both time and space. A single linear system must be solved at each time step, yielding a discrete solution for the pressure and velocity at each node.

*** Correspondingauthor:**Bart Janssens
E-mail: bart.janssens@mil.be.

For three-dimensional problems, the number of unknowns quickly reaches into the millions, imposing the use of Krylov methods to solve the linear system. For fast convergence, a preconditioner is required. We can use algebraic multigrid [3], or one of the specialized preconditioners available in literature [4-6].The convergence rate of these algorithms often degrades to some extent as the Reynolds number increases or the mesh is refined. The cost per outer Krylov iteration also increases, since most of the specialized preconditioners rely on the solution of a Poisson-like problem and a solution for the velocity linear system. The preconditioned solution of the coupled linear system imposes no restriction on the time step, so it is effective as long as the physical timescales of the flow are relatively large.

In the present work, we explore the situation where the physical timescale of interest corresponds to imposing a Courant number near unity. In [7] a predictor-multicorrector scheme is used to solve the linear system arising from the SUPG stabilized equations in a segregated fashion, i.e. solving separate equations for the velocity and the pressure while optionally treating the diffusive and advective terms

explicitly. We apply this method to the PSPG/SUPG stabilized equations with Crank-Nicolson time stepping, but keep the advective term purely explicit. The resulting scheme remains second order accurate in time and space, but it imposes a time step restriction due to the explicit treatment of the advection. The objective of this paper is to describe the development of the scheme and compare it with the existing coupled method for some benchmark cases.

We first introduce the finite element formulation for the fully coupled approach as it was presented in [2]. Next, we describe the transformation to a segregated, semi-implicit scheme. The accuracy is then evaluated using the Taylor-Green case. Finally, we run some large-scale benchmarks to compare the performance of the coupled and segregated approach before drawing conclusions.

2. Numerical method

The governing equations for incompressible flow are:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \frac{\boldsymbol{u}(\nabla\cdot\boldsymbol{u})}{2} + \nabla p - \nu\nabla^2\boldsymbol{u} = \boldsymbol{0} \qquad (1)$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad (2)$$

Here, $\boldsymbol{u}$is the velocity, p is the kinematic pressure (i.e. the pressure divided by the density), t is the time andν is the kinematic viscosity. We use the skew symmetric formulation for the advection term in the momentum equation for improved conservation of kinetic energy [8]. To obtain the finite element formulation, we multiply the equations with a set of weighting functions, interpolate the unknowns between discrete nodes using shape functions and integrate over the domain. The weighting and shape functions are chosen to be identical, yielding a Galerkin formulation. The time derivative is approximated using the θ-method. This procedure yields a discrete system, with an unknown pressure and velocity at time level n+1 to be computed at each node in the mesh. The global shape functions are non-zero only in a node and its surrounding elements. This means that the integrals can be evaluated as a sum of integrals over these elements. This global system can thus be written as the sum of N element contributions:

$$\sum_{e=1}^{N}\left(\frac{1}{\Delta t}T_e + \theta A_e\right)(\boldsymbol{x}_e^{n+1} - \boldsymbol{x}_e^{n}) = -A_e\boldsymbol{x}_e^{n} \qquad (3)$$

The parameter θ controls the time stepping and should be set to 1 for a forward Euler method and 0.5 for the Crank-Nicolson scheme. The vector of unknowns at the element level is laid out by grouping the nodal values per unknown, i.e. for a 3D element with $m+1$ nodes:

$$\boldsymbol{x}_e^n = [p_0^n \cdots p_m^n (u_0^n)_0 \cdots (u_0^n)_m \cdots (u_2^n)_m]$$

This results in the following block structure for the matrices A_e and T_e:

$$A_e = \begin{bmatrix} A_{pp} & A_{pu} \\ A_{up} & A_{uu} \end{bmatrix} = \begin{bmatrix} A_{pp} & A_{pu_0} & A_{pu_1} & A_{pu_2} \\ A_{u_0p} & A_{u_0u_0} & A_{u_0u_1} & A_{u_0u_2} \\ A_{u_1p} & A_{u_1u_0} & A_{u_1u_1} & A_{u_1u_2} \\ A_{u_2p} & A_{u_2u_0} & A_{u_2u_1} & A_{u_2u_2} \end{bmatrix} \qquad (4)$$

We now apply the stabilized finite element method to equations (1) and (2) to obtain the following expressions for each block:

$$A_{pp} = \int_{\Omega_e} \tau_{PS}\,\nabla N_p^T \nabla N_p \, d\Omega_e$$

$$A_{pu_i} = \int_{\Omega_e} \left(\left(N_p + \frac{\tau_{PS}\tilde{\boldsymbol{u}}_{adv}\nabla N_p}{2} \right)^T (\nabla N_u)_i + \tau_{PS}\left(\nabla N_p\right)_i^T \tilde{\boldsymbol{u}}_{adv}\nabla N_u \right) d\Omega_e$$

$$A_{u_iu_j} = \int_{\Omega_e} \left(\tau_{BU}(\nabla N_u)_i + \frac{1}{2}(\tilde{\boldsymbol{u}}_{adv})_i (N_u + \tau_{SU}\tilde{\boldsymbol{u}}_{adv}\nabla N_u) \right)^T (\nabla N_u)_j d\Omega_e$$

$$A_{u_iu_i} = \int_{\Omega_e} (\nu \nabla N_u^T \nabla N_u + (N_u + \tau_{SU}\tilde{\boldsymbol{u}}_{adv}\nabla N_u)^T \tilde{\boldsymbol{u}}_{adv}\nabla N_u)\, d\Omega_e + A_{u_iu_j}$$

$$A_{u_ip} = \int_{\Omega_e} (N_u + \tau_{SU}\tilde{\boldsymbol{u}}_{adv}\nabla N_u)^T \left(\nabla N_p\right)_i d\Omega_e$$

$$T_{pu_i} = \int_{\Omega_e} \tau_{PS}\left(\nabla N_p\right)_i^T N_u d\Omega_e$$

$$T_{u_i u_i} = \int_{\Omega_e} (N_u + \tau_{SU} \tilde{\boldsymbol{u}}_{adv} \nabla N_u)^T N_u d\Omega_e \quad (5)$$

Here, N_u and N_p are the shape functions for the velocity and pressure, respectively. They are row vectors of size $m+1$ with coefficients depending on the spatial coordinates. We use mapped coordinates so the integrals can easily be evaluated numerically using Gaussian quadrature. The indices i and j iterate over the number of dimensions of the problem, indicating a single component of a vector variable or a row of a gradient matrix ∇N. The stabilization terms are multiplied with their respective stabilization coefficients τ_{PS} for the PSPG stabilization, τ_{SU} for the SUPG stabilization and τ_{BU} for the bulk viscosity term. The PSPG term allows the use of equal-order interpolation for the velocity and the pressure. It introduces a non-zero A_{pp} block, consisting of the Laplacian of the pressure. The SUPG stabilization corresponds to upwinding in the streamwise direction, i.e. the weight of upstream nodes is increased. The bulk viscosity term is necessary for flows that are strongly dominated by advection, and is sometimes called "grad-div" stabilization [9] or the "least squares on incompressibility constraint" [10].

The values for the stabilization parameters must be chosen carefully: they should be large enough to obtain the stabilizing effect, but if they are too large the scheme becomes too dissipative and accuracy suffers. We follow the definitions given in [10]:

$$\tau_{SU1} = \frac{h}{2\|\tilde{\boldsymbol{u}}_{adv}\|}$$

$$\tau_{SU2} = \frac{\Delta t}{2c_1}$$

$$\tau_{SU3} = \frac{h^2}{c_2 \nu} \quad (6)$$

$$\tau_{SU} = \left(\frac{1}{\tau_{SU1}} + \frac{1}{\tau_{SU2}} + \frac{1}{\tau_{SU3}}\right)^{-1}$$

$$\tau_{PS} = \tau_{SU}$$

$$\tau_{BU} = \tau_{SU} \|\tilde{\boldsymbol{u}}_{adv}\|^2$$

Here, h is a characteristic element length. In [11], a systematic study comparing different definitions of $h$was conducted, concluding that a length scale based on the minimal edge length of an element gives the best result in the case of high aspect ratio elements. For aspect ratios closer to one, results were comparable to other possible definitions (maximum edge length and edge length in the streamwise direction). This leads us to choose theminimum element edge length as our definition for h. The parameters c_1 and c_2 are introduced in [12] and allow further control of the stabilization. Values$c_1 = 1$ and $c_2 = 4$ correspond to the definitions in [10], while in [12] the authors choose $4 \leq c_1 \leq 16$ and $c_2 = 36$.

The advection velocity $\tilde{\boldsymbol{u}}_{adv}$ is calculated using a Taylor series expansion:

$$\tilde{\boldsymbol{u}}_{adv} = 2.1875\, \boldsymbol{u}^n - 2.1875\, \boldsymbol{u}^{n-1} + 1.3125\, \boldsymbol{u}^{n-2} - 0.3125\, \boldsymbol{u}^{n-3} \quad (7)$$

This technique allows us to linearize the equations without resorting to an iterative technique, thus solving only one linear system per time step, at the cost of storing the velocity for the previous 4 time steps for every node.

In [2], the global linear system (3) is solved by directly applying the GMRES method. It is preconditioned either with algebraic multigrid or ILU factorization. There is no stability constraint for the time step if we set $0.5 \leq \theta \leq 1$, and the scheme is second order accurate in time when setting $\theta = 0.5$. Storing the coupled system is expensive, and depending on the mesh and the flow configuration the iterative method may converge slowly.

An alternative to solving the complete system is to split it into separate linear systems for the velocity and the pressure. We follow the method proposed in [7].Introducing the velocity- and pressure differences between two time levels $\Delta \boldsymbol{u}$ and Δp, we can rewrite the momentum equation as a function of the acceleration $\boldsymbol{a} = \Delta \boldsymbol{u}/\Delta t$:

$$(T_{uu} + \theta \Delta t A_{uu})\boldsymbol{a} + \theta A_{up} \Delta p = -A_{uu} \boldsymbol{u}^n - A_{up} p^n$$

Defining $\boldsymbol{a}^* = \boldsymbol{a} + (T_{uu} + \theta \Delta t A_{uu})^{-1} \theta A_{up} \Delta p$ we obtain a linear system that can be solved for $\boldsymbol{a}^*$:

$$(T_{uu} + \theta \Delta t A_{uu})\boldsymbol{a}^* = -A_{uu} \boldsymbol{u}^n - A_{up} p^n$$

The continuity equation is:

$$\left(T_{pu} + \Delta t A_{pu}\right)\boldsymbol{a} + A_{pp}\Delta p = -A_{pu}\boldsymbol{u}^n - A_{pp}p^n$$

Using the definition of $\boldsymbol{a}^*$, we can rewrite this into a linear system for the pressure differenceΔp between two time steps:

$$\left(\left(T_{pu} + \Delta t A_{pu}\right)(T_{uu} + \theta\Delta t A_{uu})^{-1}\theta A_{up} - A_{pp}\right)\Delta p = T_{pu}\boldsymbol{a}^* + A_{pu}(\boldsymbol{u}^n + \Delta t\boldsymbol{a}^*) + A_{pp}p^n$$

Note that the system matrix for the pressure is the Schur complement of the velocity block in the original system. We can now first solve the linear system for $\boldsymbol{a}^*$, then solve the Δp system and finally get the acceleration from

$$\boldsymbol{a} = \boldsymbol{a}^* - (T_{uu} + \theta\Delta t A_{uu})^{-1}\theta A_{up}\Delta p$$

So far, we have only solved the coupled system in a different way, algebraically equivalent to a direct solution. In addition to solving a separate linear system for the velocity and the pressure, we also need the inverse $(T_{uu} + \theta\Delta t A_{uu})^{-1}$ to construct the matrix for the pressure system. Doing this directly is not possible on a large mesh, so simplification is needed to obtain an efficient method. When we simplify steps in the algorithm, the result will no longer be identical to the solution of the coupled system, so we introduce an iterative algorithm that can be executed M times each time step. The linear systems will be solved for the difference between two inner iterations m and $m+1$, i.e. $\Delta\boldsymbol{a} = \boldsymbol{a}^{m+1} - \boldsymbol{a}^m$ and $\Delta p^{m+1} = p^{m+1} - p^m$. From this, we also have $\boldsymbol{u}^m = \boldsymbol{u}^n + \Delta t\boldsymbol{a}^m$ and $p^m = p^n + \sum_{i=0}^{m}\Delta p^i$. The modified $\boldsymbol{a}^*$ is then:

$$\boldsymbol{a}^{*m} = \boldsymbol{a}^m + (T_{uu} + \theta\Delta t A_{uu})^{-1}\theta A_{up}\left(\sum_{i=0}^{m}\Delta p^i\right)$$

Filling this into the original system and using the definition of $\boldsymbol{a}^*$ yields:

$$\begin{aligned}(T_{uu} + \theta\Delta t A_{uu})\Delta\boldsymbol{a}^* &= -A_{uu}\boldsymbol{u}^m - A_{up}p^m \\ &\quad - (T_{uu} + \theta\Delta t A_{uu})\boldsymbol{a}^m + A_{uu}\Delta t\boldsymbol{a}^m \\ &\quad + (1-\theta)A_{up}\sum_{i=0}^{m}\Delta p^i\end{aligned}$$

For the continuity equation we start from:

$$\left(T_{pu} + \Delta t A_{pu}\right)\Delta\boldsymbol{a} + A_{pp}\left(\sum_{i=0}^{m+1}\Delta p^i\right) = -A_{pu}\boldsymbol{u}^n - A_{pp}p^n - \left(T_{pu} + \Delta t A_{pu}\right)\boldsymbol{a}^m$$

With $\Delta\boldsymbol{a} = \Delta\boldsymbol{a}^* - (T_{uu} + \theta\Delta t A_{uu})^{-1}\theta A_{up}\Delta p^{m+1}$ this becomes:

$$\left(\left(T_{pu} + \Delta t A_{pu}\right)(T_{uu} + \theta\Delta t A_{uu})^{-1}\theta A_{up} - A_{pp}\right)\Delta p^{m+1} = T_{pu}\Delta\boldsymbol{a}^* + A_{pu}(\boldsymbol{u}^m + \Delta t\Delta\boldsymbol{a}^*) + A_{pp}p^m + T_{pu}\boldsymbol{a}^m$$

With the problem formulated this way, we can now apply a predictor-multicorrector iterative scheme:

Set $\boldsymbol{u}^0 = \boldsymbol{u}^n$, $p^0 = p^n$ and $\boldsymbol{a}^0 = 0$
for $m = 0$ *to* $M - 1$ **do**
  Solve $\Delta\boldsymbol{a}^*$ system
  Solve Δp^{m+1} system
  Compute $\Delta\boldsymbol{a}$
  Update $\boldsymbol{u}^{m+1} = \boldsymbol{u}^m + \Delta t\Delta\boldsymbol{a}$
  Update $p^{m+1} = p^m + \Delta p^{m+1}$
end for

Without simplifications to the systems, executing the iteration once will immediately provide the correct velocity- and pressure updates.

The solution of the velocity system is difficult due to the advective terms. These terms have an important impact on the convergence rate of the iterative solvers and introduce a direct dependency of the matrix coefficients on the velocity. An easy fix is to drop the advection terms from the velocity system matrix, treating them explicitly (i.e. setting $\theta = 0$ for those terms).We assemble the simplified velocity matrices $\widetilde{A_{uu}}$ and $\widetilde{T_{uu}}$using the following expressions:

$$\widetilde{A_{u_i u_i}} = \int_{\Omega_e}(\nu + \tau_{BU})\,\nabla N_u^T\nabla N_u d\Omega_e$$

$$\widetilde{A_{u_i u_j}} = \int_{\Omega_e}\tau_{BU}\,(\nabla N_u)_i^T(\nabla N_u)_j d\Omega_e (i \neq j)$$

$$\widetilde{T_{u_i u_i}} = \int_{\Omega_e} N_u^T N_u d\Omega_e$$

The system matrix of this simplified velocity system is now symmetric and much better conditioned, due to the

removal of the advective terms. The coefficients only depend on the viscosity and τ_{BU}, so they do not vary much in time, especially if the time step is small. This allows us to reuse the same velocity matrix over a range of time steps, reducing the time required for assembly and preconditioner setup.

For the pressure system, we first need an approximation for $(T_{uu} + \theta \Delta t A_{uu})^{-1}$. As suggested in [7], the inverse of the lumped velocity mass matrix M_L is a good candidate, i.e. we sum all the elements of a row and then put that value on the diagonal, making the inverse trivial to compute. The same approximation is used in [10]. After this change, the pressure system matrix becomes:

$$\left(\left(T_{pu} + \Delta t A_{pu}\right) M_L^{-1} \theta A_{up} - A_{pp}\right)$$

If we ignore the stabilization terms and apply partial integration to the pressure gradient term in the momentum equation, we have $A_{up} = -A_{pu}^T$, hinting that the structure of the pressure matrix and the Poisson problem are similar. This leads us to simplify the pressure matrix as follows:

$$\left(\left(T_{pu} + \Delta t A_{pu}\right) M_L^{-1} \theta A_{up} - A_{pp}\right) \approx -\theta(\tau_{PS} + \Delta t) \int_{\Omega_e} \nabla N_p^T \nabla N_p d\Omega_e$$

This approximation is remarkably similar to the approximation of the Schurcomplement of the velocity block for reaction-dominated flows as described in [6]. We are indeed in the situation of reaction-dominated flows since the time term is large due to the small time steps under consideration. The validity of our approximation is further confirmed by the Taylor-Green test case. The resulting matrix is symmetric and only depends on the solution through the value of τ_{PS}.Numerical experiments show that the adjustment of τ_{PS} in the matrix has no effect on the accuracy, so we can reuse the same pressure matrix during the complete calculation. This opens up the possibility of using a direct solution method or reuse of the preconditioner. While the original matrix required a sparse matrix product, the simplification can be assembled on a per-element basis. This greatly simplifies the code and speeds up the assembly. This can be important in the case of deforming meshes or variable time steps, where the pressure matrix does change with each time step.

3. Taylor-Green vortices

As a first test case, we apply the method to the Taylor-Green periodic vortices, advected by a constant velocity field. The advantage of this test case is that it is a time dependent problem with an analytical solution in closed form. The velocity components as a function of spatial coordinates and time are:

$$u = U_a - V_s \cos\left(\frac{\pi}{D}(x - U_a t)\right) \sin\left(\frac{\pi}{D}(y - V_a t)\right) e^{-\frac{2\nu\pi^2}{D^2}t}$$

$$v = V_a + V_s \sin\left(\frac{\pi}{D}(x - U_a t)\right) \cos\left(\frac{\pi}{D}(y - V_a t)\right) e^{-\frac{2\nu\pi^2}{D^2}t}$$

The pressure is:

$$p = -\frac{V_s^2}{4}\left(\cos\left(\frac{2\pi}{D}(x - U_a t)\right) + \cos\left(\frac{2\pi}{D}(y - V_a t)\right)\right) e^{-\frac{4\nu\pi^2}{D^2}t}$$

The vorticity is given by:

$$\omega = \frac{2V_s\pi}{D} \cos\left(\frac{\pi}{D}(x - U_a t)\right) \cdot \cos\left(\frac{\pi}{D}(y - V_a t)\right) e^{-\frac{2\nu\pi^2}{D^2}t}$$

This flow field represents two-dimensional, periodic vortices with diameter D and initial maximal swirl velocity V_s, advected by the advection velocity (U_a, V_a), and dissipating due to the kinematic viscosity ν. We use the following values in our tests (based on [2]): $D = 0.5$ m, $V_s = 1$ m/s, $U_a = 0.3$ m/s, $V_a = 0.2$ m/s and $\nu = 0.001$ m^2/s.The flow field is visualized in Fig.1, using contours of the dimensionless vorticity ω/ω_0 and dimensionless time $tV_s/2D$.

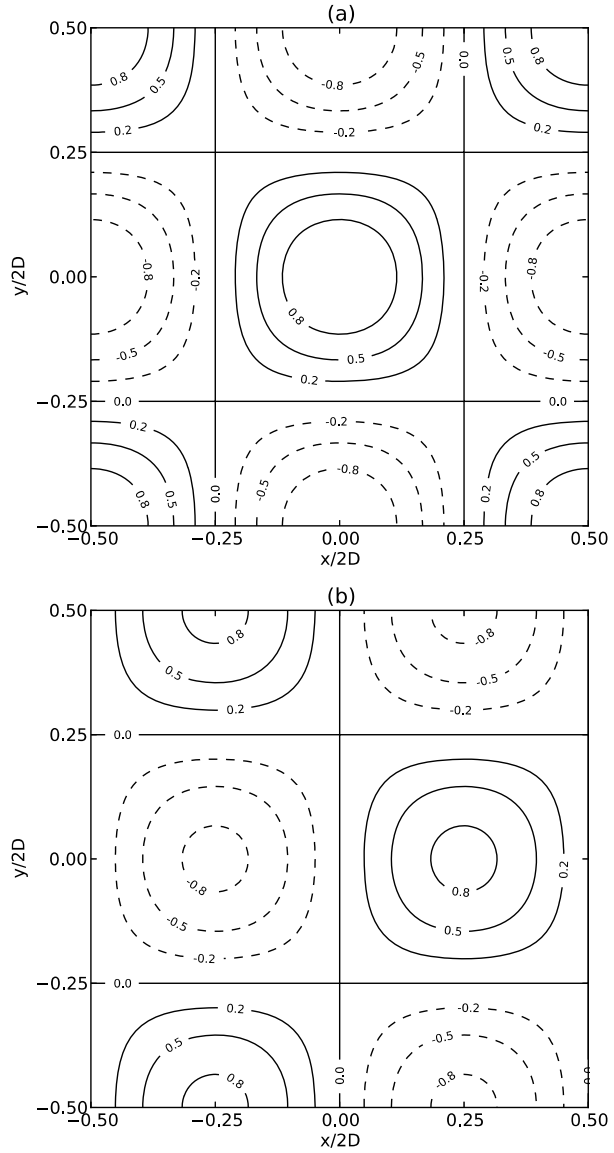

Fig. 1 Contours of dimensionless vorticity, at dimensionless time 0 (a) and 2.5 (b).

From these images, it is clear that the vortex centers move along the advection velocity vector. Viscosity redistributes the vorticity until it uniformly reaches zero everywhere in the domain, as indicated by the decrease in vorticity magnitude in the figures.

For the numerical simulations, we initialize the flow with the analytical solution at time t = 0 s and set periodic boundary conditions in both directions. Since this determines the pressure only up to a constant, we impose the pressure in the center of the domain, setting it equal to the analytical solution at every time step.

In a first test, we will determine the effect of the number of inner iterations M, using a grid of 64x64 quadrilaterals that are triangulated for the triangle element tests.In Fig. 2, the error of the segregated solution is compared to the fully coupled solution, defining the relative difference with the coupled solution as, for the x-component of the velocity:

$$\frac{\max_{\Omega}|u_c - u_s|}{\max_{\Omega}|u_c - u_{th}|}$$

Here, u_c is the solution of the coupled system of equations, $u_s$is the segregated solution and $u_{th}$is the analytical solution. For $M = 1$ and at time 6, the difference between the segregated and the coupled solution is about 10 times greater than the error between the fully coupled solution and the analytical solution, i.e. the absolute error is an order of magnitude greater. When we increase the number of iterations to two, the difference between both methods is two orders of magnitudes smaller than the absolute error, i.e. the difference is negligible and increasing the number of iterations further is not necessary. The difference flattens off after 4 iterations. At the first time step (time 0.004), the iterative technique converges towards the coupled solution as the number of iterations increases. Again, the difference with the coupled solution decreases with two orders of magnitude when using two iterations instead of one. We conclude from these observations that two inner iterations offer a good balance between computational cost and accuracy. This is no surprise: in [7] the authors point out that the term including the effect of the mass matrix on the acceleration in the right hand side of the velocity system only contributes from the second iteration onwards, since we initialize the acceleration to zero each time step.

From Fig. 2, it is clear that the difference between the two methods is greater at time 6 than after the first time step. This effect is better illustrated in Fig. 3, where we have plotted the errors$\max_{\Omega}|u_c - u_{th}|$and $\max_{\Omega}|u_s - u_{th}|$as a function of time. Both errors reach

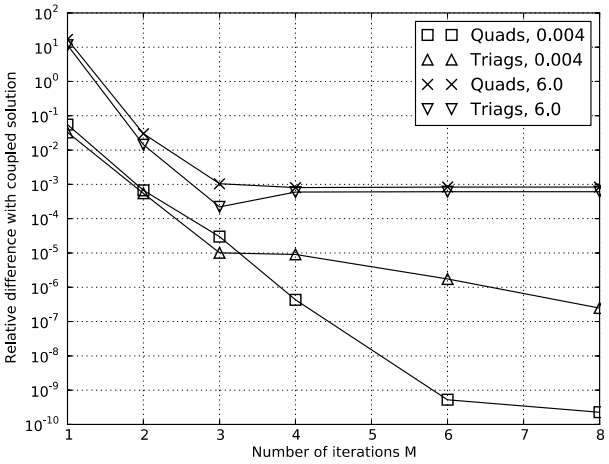

Fig. 2 Comparison between the fully coupled solution of the linear system and the current method, as a function of the number of inner iterations and at two dimensionless times 0.004 (i.e. after one time step) and 6.0.

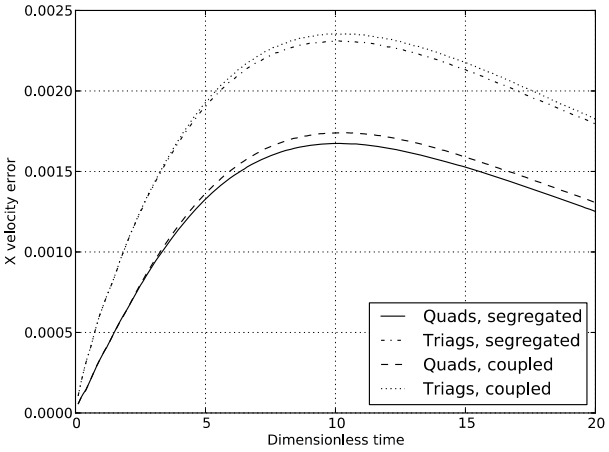

Fig. 3 Maximum error over the domain for the x-component of the velocity, for triangles and quadrilaterals and using the coupled and segregated solution method using two inner iterations.

a maximum around time 10, and so does the difference between both methods. This difference between the segregated and coupled solution remains small, varying between 2 % and 4 % of the difference with the analytical solution. This confirms that two inner iterations suffice to reproduce the results of the fully coupled solution.

Fig. 4 shows the error for a series of meshes with *N*x*N* quadrilaterals (triangulated for the triangle results). The time step was adapted to maintain a constant Courant number of 0.32 at the start of the simulation, using: $\Delta t = 0.256/N$ .We calculate the error as the maximum of the absolute value of the

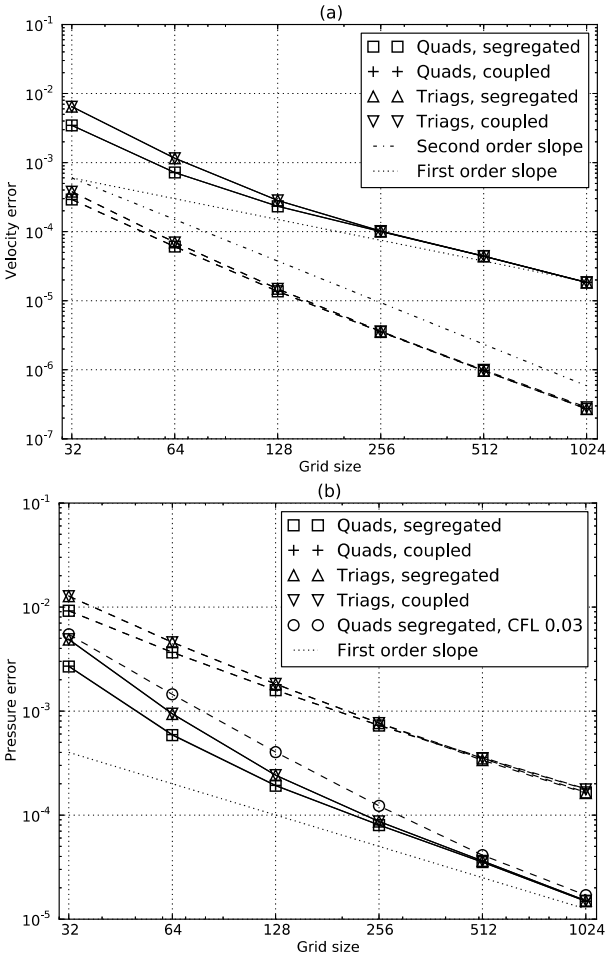

Fig.4 Maximum norm of the error for the velocity vector (a) and the pressure (b), as a function of mesh size and with a constant Courant number of 0.32. The dashed lines connect the errors after one time step, the solid lines those at time 10. Plot (b) also shows pressure error for quadrilateral elements computed at Courant number 0.03.

difference with the analytical solution over the entire domain, taking the maximum of either component for the velocity error. The velocity error after one time step (dashed line) follows the second order slope, while the pressure error follows the first order slope. The errors at time 10 (i.e. near the maximum of Fig. 3) decrease with a slope between first and second order. These effects are due to the time stepping: if we lower the Courant number to 0.03, the pressures also follow the second order law, as illustrated in Fig. 4.The errors for the coupled and the segregated method overlap, further confirming the equivalence of both methods at the time steps considered here.

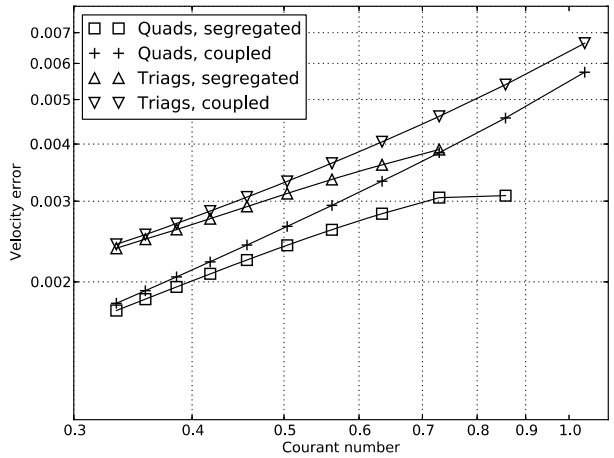

Fig. 5 Maximum norm of the error for the velocity vector as a function of Courant number on the 64x64 grid.

Fig. 5 shows the evolution of the maximum norm of the velocity error for increasing Courant numbers. The Courant number is computed here using the maximum velocity projection for all element edges. The segregated method becomes unstable when the Courant number reaches values close to 0.8, which is in line with the theoretical limits from [7]. Although we are using Crank-Nicolson time stepping, the errors in Fig. 5 do not decrease along a second order slope. Further tests with the fully coupled scheme show that second order in time is only visible at Courant number 2 and higher. In [14], a similar effect is visible in the numerical tests and this is attributed to the spatial component of the error.

4. Performance aspects

In this section we assess the performance of the segregated method. We use the direct numerical simulation of plane channel flow as a basis for the different tests, limiting the computations to 100 time steps to reduce the computational overhead. This test fits the objectives of the current method perfectly, since a DNS requires small time steps due to the physics of the flow. The meshes used in the tests are based on those from [12] and [15], using a hexahedral mesh that is refined towards the walls. We note the mesh size in the N_xxN_yxN_z format, where each N_i represents the number of nodes in the corresponding direction i(not counting periodic nodes twice). The streamwise direction corresponds to x, the wall-normal direction is y and z is the spanwise direction. Performance is measured on the following machines:

	RMA cluster	***VKI cluster***
Processor type	Xeon E5520	Opteron 6376
Cores per node	8	64
RAM per core	3 GB	4 GB
Interconnect	1 Gb ethernet	InfiniBand
Nb. nodes	30	28

The computationally expensive steps in the algorithm are the solution of the linear systems and the computation of the coefficients for the system matrices and right hand side vectors, i.e. the evaluation of the element integrals. For the fully coupled method, the matrix coefficients need to be recomputed each time step, since they depend on the advection velocity and only one linear system needs to be solved. For the segregated method, the matrix for the pressure system is constant for the whole simulation. The velocity matrix depends on the solution only through the τ_{BU} stabilization parameter. This means that the matrix coefficients are also approximately constant. Surprisingly, this results in a linear increase of the solution time per time step, as shown by the dashed line in Fig. 6. We can eliminate this effect by recomputing the coefficients every 100 time steps, resulting in a constant solution time (solid line in Fig. 6).

Since we mostly eliminated the matrix assembly from the computation, the cost for the segregated method is dominated by the solution of both linear systems and the computation of the right hand side coefficients. This work must be done every iteration, i.e. twice every time step in practice. Fig. 7 presents the scaling of the segregated method. The assembly operations follow the ideal scaling (i.e. half the time each time the number of cores is doubled) closely. This is to be expected, since this step does not depend on communication and only requires additional ghost elements as the number of mesh partitions is increased. The timing marked as “other” corresponds to some

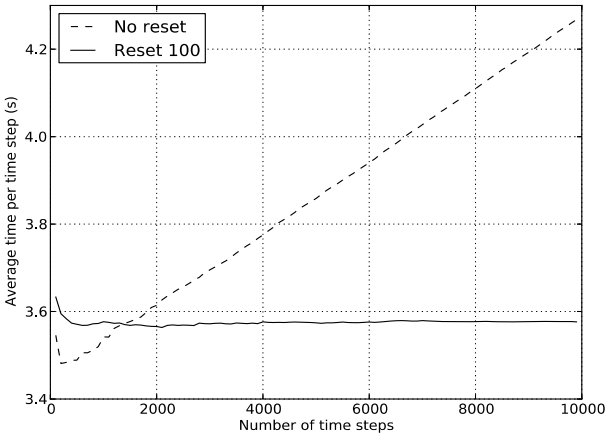

Fig. 6 Average wall clock time per time step for a 64x129x64 mesh on 64 cores on the RMA cluster. "No reset" means one single velocity matrix assembly. "Reset 100" means one velocity matrix assembly every 100 time steps.

aspects of the computation that take a negligible amount of time, such as the update of the solution and the extrapolation of the velocity for the linearization. The matrix assemblies are also included here and confirmed to be negligible in time, since they are executed at most once every 100 steps. Most of the time is spent solving the linear systems. We solve the velocity system using the Conjugate Gradient (CG) method, preconditioned using ILU factorization. The scaling is not ideal, but better than the scaling for the pressure system, which we solve using CG preconditioned with algebraic multigrid (AMG). This no longer scales when moving from 256 to 512 cores, making the solution of the pressure system the dominant factor at 512 cores.

For smaller problems, it is feasible to solve the pressure system using a direct method. Fig. 8 illustrates the effect on the timings for two different mesh sizes. The AMG timings are obtained using the same settings as before, while the MUMPS timings use the MUMPS parallel sparse direct solver [16] for the pressure system. Since the pressure matrix is constant, we only need to perform the expensive factorization once and can then apply this in all subsequent time steps. The small (32x65x32) and large (64x129x64) problems used 8 and 64 cores, respectively, thus keeping the workload per core constant. According to the ideal scaling law, the timings for the large and small

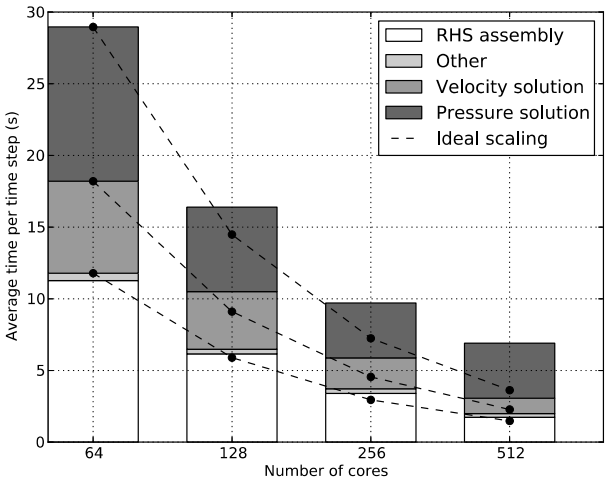

Fig. 7 Strong scaling of the average wall clock time per time step for a 128x257x128 mesh on the VKI cluster. The RHS assembly timing comprises the sum of all coefficient computations for the right hand side vectors.

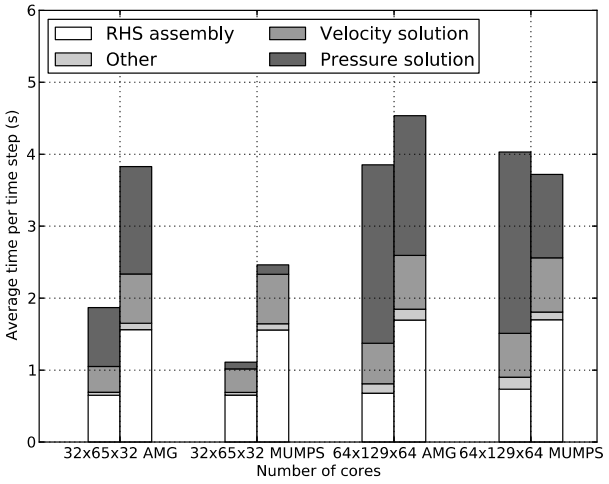

Fig. 8 Comparison between Algebraic Multigrid (AMG) and a direct solver (MUMPS) for the pressure system. Left bars for the RMA cluster, right bars for the VKI cluster. Averages for 1000 time steps.

problems should be identical, but especially on the RMA cluster the communication overhead becomes prohibitive for the large mesh. This is an effect of the 1Gb Ethernet interconnect. The assembly operations - which do not require intensive communication - do follow the ideal scaling almost perfectly.

The switch to MUMPS is very effective on the small problem: the solution time for the pressure system becomes negligible compared to the total timing, while it is the dominant factor when using AMG. The total solution time is nearly halved as a result. For the large problem, we see that the scaling for MUMPS is much

worse than AMG, resulting in very little benefit. On the RMA cluster, the initial factorization took 842s. Since we averaged the timing over 2000 iterations, there is still a contribution of 0.42s from the initial factorization in the average timing. This part will diminish as the number of iterations increases. On the VKI cluster the initial factorization only took 95s, illustrating the importance of communication in this step. As is typical for a direct method, the cost of the factorization increases non-linearly: on the small problem the timings were 5.3 s (RMA) and 6.6 s (VKI). For very large problems, the cost of the initial factorization becomes prohibitive, and the factorization itself can no longer be stored because it is much denser than the original matrix. The poor scaling of the application of the factorization is surprising, and might be due to our use of the Trilinos interface to access MUMPS. This interface also forbids the use of the symmetric solver, so better results should be possible by interfacing with MUMPS directly.

Memory usage is dominated by the storage of the linear systems. For the coupled method, the memory required to store the sparse matrix can be computed as follows, using a structured hexahedral grid where 27 nodes are adjacent to each other:

$$(27 \text{ nodes per row} * 4 \text{ variables} * 12 \text{ bytes} + 4 \text{ bytes}) * (4 \text{ equations} * \text{number of nodes})$$

We assume double precision, i.e. 8 bytes per coefficient and 32 bits for integers, i.e. 4 bytes per integer to store the coefficient index and row size. For a mesh with 10 million nodes, this yields a storage cost of 52 GB. In the case of the segregated solver, two matrices need to be stored, but because each matrix is smaller the total size is less: 3.25 GB for the pressure system and 29.25 GB for the velocity system. The savings are modest, and when using a direct solver for the pressure the segregated solver will even use more memory than the coupled method. On modern hardware, such as the clusters used in this work, problems are typically distributed over a large number of CPUs with a sufficient amount of RAM to allow either method to be chosen.

In a final test, we compare the timings per time step for the segregated and the coupled method on three different meshes, gradually finer in the wall-normal direction. The segregated solver uses the AMG method for the pressure system as described before. For the coupled method, we also use algebraic multigrid preconditioning, using the defaults optimized for advection-diffusion problems. The iterative solver is GMRES from the Belos package. Two sets of initial conditions were used: the laminar solution and a random disturbance of the laminar solution. The latter is typically used to initialize a DNS. Time steps were chosen to obtain a Courant number of around 0.15.

Table 1 summarizes the results. All tests are carried out on 8 cores on the RMA cluster, so we expect the solution time to double each time the number of mesh nodes doubles. The “mesh scaling” factor in the table lists the ratio between the current time and the time on the previous mesh, and it is always above the ideal value of 2, with significantly higher values for the coupled method. For the segregated method, the initialcondition has little impact on the solution time, but for the coupled method the added randomness appears to double the solution time. As the mesh is refined, the coupled method can take up to 48 times as long as the segregated method, so for short time steps there is a clear benefit of using the segregated approach.

Table 1 Comparison of the average time per time step for the segregated and the coupled method. Mesh scaling is the time on the current mesh divided by the time on the previous mesh. All simulations are carried out on the RMA cluster on 8 cores.

	Segregated		*Coupled*		
	Time (t_s)	**Mesh scaling**	**Time (t_c)**	**Mesh scaling**	t_c/t_s
32x65x32	1.44 s	-	7.64 s	-	5.31
32x129x32	4.78 s	3.32	35.46 s	4.64	7.42
32x257x32	12.26 s	2.57	277.96 s	7.84	22.67
32x65x32 random	1.70 s	-	15.26 s	-	8.98
32x129x32 random	3.60 s	2.12	66.00 s	4.33	18.35
32x257x32 random	11.99 s	3.33	575.30 s	8.72	47.98

5. Conclusion and future work

We have presented a segregated, semi-implicit solution method for the PSPG/SUPG stabilized incompressible Navier-Stokes equations, using a predictor-multicorrector scheme. Some simplifications to the linear systems were introduced, leading to a simpler problem at the cost of introducing a stability limit on the time step. The segregated method was shown to converge to the fully coupled solution in two inner iterations, based on tests using the Taylor-Green vortices.

We analyzed the performance on a series of meshes for plane channel flow. The solution of the pressure system was identified as the most time consuming step in the algorithm. For small problems, direct solution of the pressure system can result in an important speed up, but this method appears to scale poorly. When compared to the fully coupled solution, the segregated solution has a clear benefit if the time step needs to be small (under the CFL limit) for physical reasons.

In future work, other methods for a more efficient solution of the pressure system could be investigated. One option is to interface directly with the MUMPS solver, which would allow the use of the symmetric solver, thus halving the memory requirements. This would possibly also result in better scaling for the application of the factorization.

Acknowledgement

The software developed in this work is part of the Coolfluid 3 framework, freely available under LGPL v3 license at http://coolfluid.github.com. We thank the Coolfluid 3 development team for the many hours of work and helpful discussions, without which this work would not have been possible. In particular: Tiago Quintino for laying out the basic framework; Willem Deconinck for the work on the mesh structure; and Quentin Gasper for the work on the GUI.

References

[1] Tezduyar, T. E.; Mittal, S.; Ray, S. & Shih, R. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements, Computer Methods in Applied Mechanics and Engineering, 95(1992), 221-242

[2] Bányai, T.; VandenAbeele, D. &Deconinck, H. A fast fully-coupled solution algorithm for the unsteady incompressible Navier-Stokes equations, Conference on Modelling Fluid Flow (CMFF'06), (2006)

[3] Gee, M. W.; Siefert, C. M.; Hu, J. J.; Tuminaro, R. S. &Sala, M. G. ML 5.0 smoothed aggregation user's guide, Sandia National Laboratories, Tech. Rep. SAND2006 - 2649 (2006)

[4] Elman, H. C.; Howle, V. E.; Shadid, J. N. &Tuminaro, R. S. A parallel block multi-level preconditioner for the 3D

incompressible Navier-Stokes equations, Journal of Computational Physics, 187 (2003), 504-523

[5] urRehman, M.; Vuik, C. & Segal, G. A comparison of preconditioners for incompressible Navier-Stokes solvers, International Journal for Numerical Methods in Fluids, 57 (2008), 1731

[6] Heister, T. &Rapin, G. Efficient augmented Lagrangian-type preconditioning for the Oseen problem using Grad-Div stabilization International Journal for Numerical Methods in Fluids,71 (2013), 118-134

[7] Brooks, A. N. & Hughes, T. J. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, Computer methods in applied mechanics and engineering, 32 (1982), 199-259

[8] Zang, T. A. On the rotation and skew-symmetric forms for incompressible flow simulations Applied Numerical Mathematics, 7 (1991), 27-40

[9] Braack, M.; Burman, E.; John, V. & Lube, G. Stabilized finite element methods for the generalized Oseen problem, Computer methods in applied mechanics and engineering, 196 (2007), 853-866

[10] Tezduyar, T. &Sathe, S. Stabilization parameters in SUPG and PSPG formulations, Journal of computational and applied mechanics, 4 (2003), 71-88

[11] Mittal, S. On the performance of high aspect ratio elements for incompressible flows,Comput. Methods Appl. Mech. Engrg., 188 (2000), 269-287

[12] Trofimova, A. V.; Tejada-Martinez, A. E.; Jansen, K. E. &Lahey, R. T. Direct numerical simulation of turbulent channel flows using a stabilized finite element method, Computers & Fluids, 38 (2009), 924-938

[13] Elman, H.; Howle, V.; Shadid, J.; Shuttleworth, R. &Tuminaro, R. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier--Stokes equations, Journal of Computational Physics, 227 (2008), 1790-1808

[14] Codina, R.; Principe, J.; Guasch, O. &Badia, S. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems, Computer Methods in Applied Mechanics and Engineering, 196 (2007), 2413-2430

[15] Moser, R. D.; Kim, J. & Mansour, N. N.Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau = 590$, Physics of Fluids, 11 (1999), 943-94

[16] Amestoy, P. R.; Duff, I. S.; L'Excellent, J.-Y. &Koster, J. MUMPS: a general purpose distributed memory sparse solver Applied Parallel Computing. New Paradigms for HPC in Industry and Academia, 2001, 121-130