Implementation of a stabilized finite element formulation for the incompressible Navier–Stokes equations based on a pressure gradient projection

Ramon Codina[a,*,1], Jordi Blasco[a,2], Gustavo C. Buscaglia[b,3] and Antonio Huerta[a,4]

[a] *Universitat Politècnica de Catalunya, Barcelona, Spain*
[b] *Centro Atómico de Bariloche and Instituto Balseiro, Bariloche, Argentina*

SUMMARY

We discuss in this paper some implementation aspects of a finite element formulation for the incompressible Navier–Stokes equations which allows the use of equal order velocity–pressure interpolations. The method consists in introducing the projection of the pressure gradient and adding the difference between the pressure Laplacian and the divergence of this new field to the incompressibility equation, both multiplied by suitable algorithmic parameters. The main purpose of this paper is to discuss how to deal with the new variable in the implementation of the algorithm. Obviously, it could be treated as one extra unknown, either explicitly or as a condensed variable. However, we take for granted that the only way for the algorithm to be efficient is to uncouple it from the velocity–pressure calculation in one way or another. Here we discuss some iterative schemes to perform this uncoupling of the pressure gradient projection (PGP) from the calculation of the velocity and the pressure, both for the stationary and the transient Navier–Stokes equations. In the first case, the strategies analyzed refer to the interaction of the linearization loop and the iterative segregation of the PGP, whereas in the second the main dilemma concerns the explicit or implicit treatment of the PGP.

KEY WORDS: finite element; incompressible flow; Navier–Stokes equations; pressure gradient projection

1. INTRODUCTION

Finite element formulations for incompressible flows allowing to circumvent the inf-sup stability condition for the velocity and pressure interpolations are the subject of active research. These may fall basically into two categories, namely, methods that allow the use of equal interpolations (and therefore continuous pressures) and techniques to stabilize simple

* Correspondence to: Universitat Politècnica de Catalunya, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain.
[1] E-mail: ramon.codina@upc.es
[2] E-mail: blasco@ma1.upc.es
[3] E-mail: gustavo@cab.cnea.gov.ar
[4] E-mail: antonio.huerta@upc.es

elements, such as the Q_1/P_0 pair (multilinear velocity, piecewise constant pressure). Examples of the first group are the methods in [1,2], the Galerkin/least-squares (GLS) technique [3–5] and least-squares methods for first-order systems as those in [6], whereas examples of the second are those in [7,8], among others.

The practical numerical implementation of such stabilized formulations is sometimes not addressed completely. For instance, methods involving pressure jumps, integrals over boundaries, the introduction of many auxiliary variables or macroelement filtering strategies may have attractive convergence results any may be theoretically appealing. However, they may involve a high computational complexity that makes these methods hardly applicable in real-life problems, where the weight of efficiency in the accuracy–efficiency balance is higher the larger the problem is. Maybe one of the reasons for the success of the Galerkin/least-squares technique is that its implementation is straightforward, at least for linear and multilinear elements for which elementwise second derivatives of finite element functions are either zero or can be neglected.

In this paper we discuss the implementation of a pressure stabilized finite element method for the incompressible Navier–Stokes equations whose motivation, development and theoretical foundations have been presented in [9–11]. This method can be considered of the same type as the GLS method, in the sense that it intends to allow the use of equal velocity–pressure interpolations. This is possible due to a stabilization technique based on the introduction as unknown of the discrete problem of the pressure gradient projection (PGP) onto the finite element space of continuous vector fields. The divergence of the difference between these two vectors (pressure gradient and its projection) is introduced in the continuity equation multiplied by suitable algorithmic parameters. This method turns out to be formally the same as that proposed independently in [12]. We shall refer to the resulting formulation as the *stabilized by pressure gradient projection* (SPGP) method.

The presence of the PGP as a new unknown in the problem makes the applicability of the method questionable, since the number of nodal unknowns is increased. The main purpose of this paper is to show that it is not necessary to solve in a fully coupled manner for the velocity, the pressure and the PGP. For stationary problems, this can be done by using iterative techniques, but for transient problems the natural way to uncouple the resolution is by treating explicitly the PGP. We show that this is not only feasible, but also very convenient in general situations. Nevertheless, in some cases the iterative coupling is also efficient.

Note also that the only purpose of the stabilization technique presented here is to stabilize the pressure. The instabilities due to the convective term when the viscosity is very small are not considered in our formulation and thus they have to be treated by other stabilization mechanisms. This point is also addressed later on.

We have organized the paper in two main sections. In the next one, the pressure stabilized method is fully described and their stability and convergence properties summarized. Its performance will be compared against the algebraic subgrid scale (ASGS) method, which is also presented at the end of Section 2. This method is based on the ideas of [13], whereas its extension to the Navier–Stokes equations and its analysis is presented in (Codina R. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, submitted). In Section 3 the implementation aspects of the SPGP formulation are treated, starting with the matrix formulation of the

method, a brief discussion on the possibility of dealing with the full system, and the basic description of iterative schemes for the non-linear problem and the PGP treatment in transient problems. Numerical results are presented in Section 4. As is shown there, even though the computational cost relative to the ASGS method is problem dependent, the accuracy is usually higher, in particular in what concerns the pressure approximation near boundaries. The summary of the most salient conclusions is finally presented in Section 5.

2. DESCRIPTION OF THE METHOD

2.1. Problem statement

Let us consider the transient Navier–Stokes equations for an incompressible fluid. Let Ω be an open, bounded and polyhedral domain of $\mathbb{R}^{n_{sd}}$, where $n_{sd}=2$ or 3 is the number of space dimensions, $\Gamma=\partial\Omega$ its boundary and $[0, T]$ the time interval of analysis. The Navier–Stokes problem consists in finding a velocity $\boldsymbol{u}$ and a pressure p such that

$$\partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \quad \text{in } \Omega, \quad t\in(0, T) \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega, \quad t\in(0, T) \tag{2}$$

$$\boldsymbol{u} = 0 \quad \text{on } \Gamma, \quad t\in(0, T) \tag{3}$$

$$\boldsymbol{u} = \boldsymbol{u}^0 \quad \text{in } \Omega, \quad t=0 \tag{4}$$

where ν is the kinematic viscosity, $\boldsymbol{f}$ is the force vector and $\boldsymbol{u}^0$ is the velocity initial condition. We have considered the homogeneous Dirichlet boundary condition (3) for simplicity.

To write the weak form of problems (1)–(4) we need to introduce some notation. We denote by $H^1(\Omega)$ the Sobolev space of functions whose first derivatives belong to $L^2(\Omega)$, and by $H^1_0(\Omega)$ the subspace of $H^1(\Omega)$ of functions with zero trace on Γ. A bold character is used for the vector counterpart of these spaces. The L^2 scalar product in a set ω is denoted by $(\cdot,\cdot)_\omega$, and the L^2 norm by $\|\cdot\|_\omega$. The subscript ω is omitted when it coincides with Ω. To pose the problem, we also need the functional spaces $V_{st}=\boldsymbol{H}^1_0(\Omega)^{n_{sd}}$, and $Q_{st}=\{q\in L^2(\Omega)\,|\int_\Omega q=0\}$ as well as $V=\boldsymbol{L}^2(0, T; V_{st})$ and $Q=L^2(0, T; Q_{st})$ for the transient problem.

Assuming for simplicity the force vector to be square integrable, the weak form of problems (1)–(4) consists in finding $(\boldsymbol{u}, p)\in V\times Q$ such that

$$(\partial_t\boldsymbol{u}, \boldsymbol{v}) + (\boldsymbol{u}\cdot\nabla\boldsymbol{u}, \boldsymbol{v}) + \nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v}) - (p, \nabla\cdot\boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) \tag{5}$$

$$(q, \nabla\cdot\boldsymbol{u}) = 0 \tag{6}$$

for all $(\boldsymbol{v}, q)\in V_{st}\times Q_{st}$, and satisfying the initial condition in a weak sense.

2.2. Time discretization

Any time integration of (5)–(6) is in principle possible. However, we shall concentrate on the monolithic trapezoidal rule (solving for the velocity and the pressure at the same time). The time discretized version of (5)–(6) in this case consists in solving the following problem: from known $\boldsymbol{u}^n$, find $\boldsymbol{u}^{n+1} \in V_{\mathrm{st}}$ and $p^{n+1} Q_{\mathrm{st}}$ such that

$$(\delta_t^n \boldsymbol{u}, \boldsymbol{v}) + (\boldsymbol{u}^{n+\theta} \cdot \nabla \boldsymbol{u}^{n+\theta}, \boldsymbol{v}) + \nu(\nabla \boldsymbol{u}^{n+\theta}, \nabla \boldsymbol{v}) - (p^{n+1}, \nabla \cdot \boldsymbol{v}) = (\bar{\boldsymbol{f}}^{n+\theta}, \boldsymbol{v}) \tag{7}$$

$$(q, \nabla \cdot \boldsymbol{u}^{n+1}) = 0 \tag{8}$$

for all $(\boldsymbol{v}, q) \in V_{\mathrm{st}} \times Q_{\mathrm{st}}$, where δt is the time step size, superscript m refers to the time step level $t^m = m\delta t$, $\theta \in [0, 1]$ and we use the notation

$$\boldsymbol{u}^{n+\theta} := \theta \boldsymbol{u}^{n+1} + (1-\theta)\boldsymbol{u}^n, \quad \delta \boldsymbol{u}^n := \boldsymbol{u}^{n+1} - \boldsymbol{u}^n \quad \text{and} \quad \delta_t^n \boldsymbol{u} := \frac{\delta \boldsymbol{u}^n}{\delta t}$$

The force term $\bar{\boldsymbol{f}}^{n+\theta}$ in (7) and below has to be understood as the time average of the force in the interval $[t^n, t^{n+1}]$, even though we use a superscript $n+\theta$ to characterize it. The pressure value computed here has been identified as the pressure evaluated at t^{n+1}, although this is irrelevant for the velocity approximation. The values of interest of θ are $\theta = 1/2$, corresponding to the second-order Crank–Nicolson scheme, and $\theta = 1$, which corresponds to the backward Euler method. In this case, the convective term in (7) can be replaced by $(\boldsymbol{u}^n \cdot \nabla \boldsymbol{u}^{n+1}, \boldsymbol{v})$, since it also leads to a first-order unconditionally stable scheme, well suited for the long-term time integration.

2.3. Pressure stabilized finite element discretization

Let $\mathcal{T}_h$ denote a finite element partition of the domain Ω of diameter h, from which we construct the finite element spaces Q_h, V_h and $V_{h,0}$, approximations to Q_{st}, $\boldsymbol{H}^1(\Omega)^{n_{\mathrm{sd}}}$ and V_{st} respectively. The former is made up with continuous functions of degree k_{q} and the other two with continuous vector functions of degree k_{v}, the latter verifying the homogeneous Dirichlet boundary conditions.

The Galerkin finite element approximation of problem (7)–(8) is standard. It is well known that the finite element spaces used to interpolate the velocity and pressure need to satisfy the discrete inf-sup condition (see [14] for a complete discussion and analysis of this problem when $\theta = 1/2$). In order to avoid this, a pressure stabilized finite element formulation was proposed in [9] for the stationary Stokes problem, extended to the stationary Navier–Stokes equations in [11] and to the transient case in [10]. The method consists in adding to the incompressibility equation the divergence of the difference between the pressure gradient and its projection onto V_h, both multiplied by algorithmic parameters defined elementwise. Similarly to the ASGS method (see Section 2.5), we take these parameters as

$$\tau_K := \left[c_1 \frac{\nu}{h_K^2} + c_2 \frac{|\boldsymbol{u}_h|_{\infty, K}}{h_K} \right]^{-1} \tag{9}$$

for $K \in \mathscr{T}_h$, where h_K is the diameter of K, $|\boldsymbol{u}_h|_{\infty,K}$ the supremum of the norm of $\boldsymbol{u}_h$ in K and c_1 and c_2 are algorithmic constants, that we take as $c_1 = 4$ and $c_2 = 2$ for linear elements and $c_1 = 16$ and $c_2 = 4$ for quadratics.

Having introduced these parameters, the discrete version of problem (7)–(8) can be defined as: find finite the element approximations $(\boldsymbol{u}_h^{n+\theta}, p_h^{n+1})$ to $(\boldsymbol{u}^{n+\theta}, p^{n+1})$ and also $\boldsymbol{\xi}_h^{n+1}$ such that

$$(\delta_t^n \boldsymbol{u}_h, \boldsymbol{v}_h) + (\boldsymbol{u}_h^{n+\theta} \cdot \nabla \boldsymbol{u}_h^{n+\theta}, \boldsymbol{v}_h) + \nu(\nabla \boldsymbol{u}_h^{n+\theta}, \nabla \boldsymbol{v}_h) - (p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h) = (\bar{\boldsymbol{f}}^{n+\theta}, \boldsymbol{v}_h) \tag{10}$$

$$\sum_K \tau_K (\nabla q_h, \nabla p_h^{n+1} - \boldsymbol{\xi}_h^{n+\beta})_K + (q_h, \nabla \cdot \boldsymbol{u}_h^{n+1}) = 0 \tag{11}$$

$$(\nabla p_h^{n+1}, \boldsymbol{\eta}_h) - (\boldsymbol{\xi}_h^{n+1}, \boldsymbol{\eta}_h) = 0 \tag{12}$$

for all $(\boldsymbol{v}_h, q_h, \boldsymbol{\eta}_h) \in \boldsymbol{V}_{h,0} \times Q_h \times \boldsymbol{V}_h$.

In these equations we have introduced the parameter β, whose values of interest are $\beta = 0$ and $\beta = 1$. In the first case, the pressure gradient projection is treated explicitly, whereas in the second it is treated implicitly.

2.4. *Stability and convergence*

In this section we summarize the main results proved in [9–11] concerning the stability and convergence properties of both (10)–(12) and its stationary version. All these results were obtained taking $c_2 = 0$ in (9), which leads to stability and convergence estimates whose 'constants' depend on the inverse of ν. Therefore, from the numerical point of view they are only meaningful for large values of ν. When this parameter is small, convection needs to be stabilized (see Section 2.5). For the sake of simplicity, we consider here the case of quasi-uniform meshes, with equal algorithmic parameters for all the elements (although the numerical examples presented in Section 4 have been solved using expression (9) for each element).

Let us consider first the stationary problem (obtained by setting $\delta t = \infty$ in (10)). It can be shown that the solution $(\boldsymbol{u}_h, p_h)$ is stable when continuous pressure interpolations are used. More precisely, the following stability estimate holds

$$\nu \|\nabla \boldsymbol{u}_h\|^2 + \tau \|\nabla p_h\|^2 \leq C \|\boldsymbol{f}\|^2 \tag{13}$$

Here and below, C, possibly with subscripts, denotes a constant independent of h and δt, but possibly depending on ν.

Convergence can also be proven if the solution $(\boldsymbol{u}, p)$ of the continuous problem is smooth enough. The convergence estimate associated to the previous stability result is

$$\nu \|\nabla \boldsymbol{u} - \nabla \boldsymbol{u}_h\|^2 + \tau \|\nabla p - \nabla p_h\|^2 \leq C h^{2k}, \quad k = \min\{k_{\mathbf{v}}, k_{\mathbf{q}} + 1\} \tag{14}$$

Let us move now to the transient case. The stability estimate for the velocity is

$$\|\boldsymbol{u}_h^{N+1}\|^2 + \sum_{n=0}^{N} \delta t \nu \|\nabla \boldsymbol{u}_h^{n+1}\|^2 \leq C_1 \sum_{n=0}^{N} \delta t \|\boldsymbol{f}^{n+1}\|^2 + C_2 \tag{15}$$

whereas for the pressure it is

$$\sum_{n=0}^{N} \delta t (\tau \|\nabla p_h^{n+1}\|^2)^s \leq C_1 \sum_{n=0}^{N} \delta t \|\boldsymbol{f}^{n+1}\|^2 + C_2 \tag{16}$$

where $s = 1/2$ for the three-dimensional Navier–Stokes equations and $s = 1$ if either the convective term is dropped (Stokes problem) or $n_{sd} = 2$ (recall that C_1 and C_2 can depend on ν in these estimates).

Concerning convergence for the transient problem, when $\theta = 1$ and the solution of the continuous problem is smooth enough it can be shown that the error in the norms defined by the left-hand-sides of (15) and (16) is of order $O(\delta t + h^k)$, where k is the same as in (14). We have not performed yet the analysis of the scheme with $\theta = 1/2$, but we presume that results similar to those obtained for the Galerkin method in [14] will also hold in our case.

2.5. Algebraic subgrid scale stabilized method

Perhaps the simplest finite element formulation that allows equal velocity–pressure interpolations is the GLS method (see [15] for its application to the Navier–Stokes equations). In the numerical examples, we shall compare the performance of the SPGP and the ASGS method, which is very similar to the GLS method (identical for linear elements) but easier to extend to more general settings.

The basic idea of the ASGS method can be found in [13] and its application to the Navier–Stokes equations in Ref. [26]. We shall use here a slightly simplified version of the method presented in this reference. Calling

$$\boldsymbol{m}_{h,K}^{n+\theta} := \delta_t^n \boldsymbol{u}_h + \boldsymbol{u}_h^{n+\theta} \cdot \nabla \boldsymbol{u}_h^{n+\theta} - \nu \Delta \boldsymbol{u}_h^{n+\theta} + \nabla p_h^{n+1} \quad \text{in } K \in \mathcal{T}_h$$

the momentum rate within each element, the method consists in finding $(\boldsymbol{u}_h^{n+\theta}, p_h^{n+1}) \in V_{h,0} \times Q_h$ such that

$$(\delta_t^n \boldsymbol{u}_h, \boldsymbol{v}_h) + (\boldsymbol{u}_h^{n+\theta} \cdot \nabla \boldsymbol{u}_h^{n+\theta}, \boldsymbol{v}_h) + \nu (\nabla \boldsymbol{u}_h^{n+\theta}, \nabla \boldsymbol{v}_h) - (p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h) - (\boldsymbol{f}^{n+1}, \boldsymbol{v}_h)$$
$$+ \sum_K \tau_K (\boldsymbol{u}_h^{n+\theta} \cdot \nabla \boldsymbol{u}_h + \nu \Delta \boldsymbol{v}_h, \boldsymbol{m}_{h,K}^{n+\theta} - \boldsymbol{f}^{n+1})_K = 0 \tag{17}$$

$$\sum_K \tau_K (\nabla q_h, \boldsymbol{m}_{h,K}^{n+\theta} - \boldsymbol{f}^{n+1})_K + (q_h, \nabla \cdot \boldsymbol{u}_h^{n+1}) = 0 \tag{18}$$

for all $(\boldsymbol{v}_h, q_h) \in V_{h,0} \times Q_h$. There is also the possibility of adding a term involving a least-squares form of the velocity divergence, that we have omitted for simplicity. As shown in Ref. [26], if the parameter τ_K is taken as indicated in (9), optimal convergence can be proved for the linearized problem. Observe also that this scheme is identical to the GLS method except for the $+$ sign of the viscous operator applied to the velocity test function $\boldsymbol{v}_h$.

At this point it is important to highlight the differences in the implementation of the ASGS and the SPGP methods:

1. The presence of the term $\boldsymbol{u}_h^{n+\theta}\cdot\nabla\boldsymbol{v}_h$ multiplying the element residual in (17) introduces a streamline diffusion that stabilizes convection. For the SPGP method, this term, or a similar one, must be introduced also when convection dominates. This would imply that the SPGP method is combined with the SUPG method.
2. The *whole* element residual needs to be computed within each element when the ASGS is employed. In particular, second-order derivatives of the shape functions have to be computed and stored for higher-order elements, a cumbersome and time consuming process.
3. No variables other than the velocity and the pressure need to be dealt with when the ASGS is used. In particular, no projections need to be performed.

Clearly, from a computational point of view item 2 is favorable to the SPGP method and item 3 to the ASGS method. The issue is to decide whether they make one method preferable to the other, once combined with accuracy considerations. Our purpose in this paper is to give hints in this direction, although we anticipate that no definite conclusion is to be expected.

The stabilization of convection is a point that we shall omit in our discussion. Obviously, if a SUPG strategy needs to be employed combined with the SPGP, the whole element residual needs to be computed also in this case. Nevertheless, it is shown in [16] that the SPGP method can be consistently extended to stabilize also convection.

3. IMPLEMENTATION ASPECTS

3.1. Matrix form

For our discussion concerning the implementation of the SPGP method given by (10)–(12), it is convenient to introduce the matrix version of this problem.

Let N^a be the standard (Lagrangian) shape function associated to node a of the finite element mesh. If equal interpolation is used for all the variables (velocity, pressure and PGP) the matrix form of the problem is

$$\mathbf{M}_0\delta_t\mathbf{U}^n + \mathbf{K}_{\mathrm{conv}}(\mathbf{U}^{n+\theta})\mathbf{U}^{n+\theta} + \mathbf{K}_{\mathrm{visc}}\mathbf{U}^{n+\theta} + \mathbf{G}_0\mathbf{P}^{n+1} = \mathbf{F}^{n+1} \tag{19}$$

$$-\mathbf{L}_\tau\mathbf{P}^{n+1} + \mathbf{D}_\tau\boldsymbol{\Xi}^{n+\beta} + \mathbf{D}\mathbf{U}^{n+1} = \mathbf{0} \tag{20}$$

$$\mathbf{G}\mathbf{P}^{n+1} - \mathbf{M}\boldsymbol{\Xi}^{n+1} = \mathbf{0} \tag{21}$$

If we denote the node indexes with superscripts a, b and the space indexes with subscripts i, j, the components of the arrays involved in these equations are:

$$\mathbf{M}_{ij}^{ab} = (N^a, N^b)\delta_{ij}$$

$$\mathbf{K}_{\mathrm{conv}}(\mathbf{U})_{ij}^{ab} = (N^a, \boldsymbol{u}_h \cdot \nabla N^b)\delta_{ij}$$

$$\mathbf{K}_{\mathrm{visc}}{}_{ij}^{ab} = \nu(\nabla N^a, \nabla N^b)\delta_{ij}$$

$$\mathbf{G}_i^{ab} = (N^a, \partial_i N^b)$$

$$\mathbf{F}_i^a = (N^a, f_i)$$

$$\mathbf{L}_\tau^{ab} = -\sum_K \tau_K (\nabla N^a, \nabla N^b)_K$$

$$\mathbf{D}_{\tau j}^{ab} = -\sum_K \tau_K (\partial_j N^a, N^b)_K$$

$$\mathbf{D}_j^{ab} = (N^a, \partial_j N^b)$$

Matrices $\mathbf{M}_0$ and $\mathbf{G}_0$ have the same components as $\mathbf{M}$ and $\mathbf{G}$ respectively, with index a running only over the set of interior nodes. Also, it is understood that all the arrays are matrices (except $\mathbf{F}$, which is a vector) whose components are obtained by grouping together the left indexes in the previous expressions (a and possibly i) and the right indexes (b and possibly j). Finally, note that $\mathbf{D}_j^{ab} = \mathbf{G}_j^{ab}$.

3.2. *Two-step Laplacian matrix*

If the nodal unknowns of the PGP are eliminated from (21) it is found that

$$\Xi^{n+1} = \mathbf{M}^{-1}\mathbf{G}\mathbf{P}^{n+1}$$

Inserting the expression of $\Xi^{n+\beta}$ in (20) yields

$$\mathbf{D}_\tau \mathbf{M}^{-1}\mathbf{G}\mathbf{P}^{n+\beta} - \mathbf{L}_\tau \mathbf{P}^{n+1} + \mathbf{D}\mathbf{U}^{n+1} = \mathbf{0}$$

If $\beta = 0$ the first term can be moved to the right-hand-side of the resulting algebraic system. Let us concentrate on the case $\beta = 1$. Introducing the notation

$$\tilde{\mathbf{L}}_\tau := \mathbf{D}_\tau \mathbf{M}^{-1}\mathbf{G}$$

the matrix of the system to be solved is of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{G}_0 \\ \mathbf{D} & \tilde{\mathbf{L}}_\tau - \mathbf{L}_\tau \end{bmatrix} \tag{22}$$

where $\mathbf{K}$ is the matrix multiplying the vector of nodal velocities, which takes into account the temporal, convective and viscous contributions.

Except for the dependence of $\tilde{\mathbf{L}}_\tau$ on τ, this can be understood as the matrix resulting from the discrete divergence applied to the projection of the discrete gradient. Therefore, it is a Laplacian computed in two steps. It is shown in [17] that the difference $\tilde{\mathbf{L}}_\tau - \mathbf{L}_\tau$ is positive semi-definite, thus explaining from an algebraic point of view why the terms depending on τ in (19)–(21) enhance the stability with respect to the standard Galerkin approach.

If an algebraic system with matrix (22) is to be solved exactly, there are basically two possibilities to deal with $\tilde{\mathbf{L}}_\tau$, depending on the method of solution:

3.2.1. Direct solution. In this case, $\tilde{\mathbf{L}}_\tau$ needs to be stored. The only feasible way to do this is by approximating the mass matrix $\mathbf{M}$ by the (lumped) diagonal mass matrix $\mathbf{M}_\ell$ since otherwise $\mathbf{M}^{-1}$ is a full matrix. This approach has been used in fractional step methods which do not use the continuous pressure Laplacian equation (see e.g. [18,19]), since matrix $\tilde{\mathbf{L}} := \mathbf{D}\mathbf{M}^{-1}\mathbf{G}$ appears in the projection step. However, this is computationally expensive because the non-zero entries of this matrix are not only those of the mesh graph, but of its square (using a finite difference expression, it has a twice larger stencil).

3.2.2. Iterative solution. In this case, the problem is to compute matrix–vector products of the form $(\tilde{\mathbf{L}}_\tau - \mathbf{L}_\tau)\mathbf{P}$. This can be efficiently done in three steps as follows:

1. Compute $\mathbf{Y} = \mathbf{G}\mathbf{P}$.
2. Solve $\mathbf{M}\mathbf{Z} = \mathbf{Y}$. This is trivial if $\mathbf{M}$ is approximated by $\mathbf{M}_\ell$. Otherwise, the simple Jacobi iteration

$$\mathbf{Z}_0 = \mathbf{0}; \qquad \mathbf{Z}_{k+1} = \mathbf{Z}_k + \mathbf{M}_\ell^{-1}(\mathbf{Y} - \mathbf{M}\mathbf{Z}_k) \quad k = 0, 1, 2, \ldots \tag{23}$$

   is extremely efficient. We shall use it later on.
3. Compute $\mathbf{D}_\tau\mathbf{Z} - \mathbf{L}_\tau\mathbf{P}$.

This algorithm is similar to what is commonly used for the dual problem for the Lagrange multiplier in conjugate gradient methods for optimization problems with restrictions. The matrix of the system in this case is $\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^t$, $\mathbf{A}$ being the matrix for the primal problem and $\mathbf{B}$ the matrix of the restrictions.

In order to avoid the numerical difficulties present in both direct and iterative solution methods, we advocate for the iterative elimination of the PGP. This is the subject of the following sections and the strategy employed in the numerical results.

3.3. Non-linear iterative schemes

In this section we consider three different possibilities of uncoupling the PGP from the velocity–pressure calculation. We assume that $\theta \geq 1/2$ in (19) and therefore the convective term of the Navier–Stokes equations needs to be linearized, either with the simple Picard iteration or with the Newton–Raphson scheme. For the sake of simplicity, only the first

method is considered here, although our results apply to the Newton–Raphson linearization as well.

Using the same notation as in (22), the non-linear algebraic system to be solved within each time step is of the form

$$\mathbf{K}(\mathbf{U})\mathbf{U} + \mathbf{G}_0\mathbf{P} = \mathbf{F} \tag{24}$$

$$-\mathbf{L}_\tau\mathbf{P} + \mathbf{D}_\tau\boldsymbol{\Xi} + \mathbf{D}\mathbf{U} = \mathbf{0} \tag{25}$$

$$\mathbf{G}\mathbf{P} - \mathbf{M}\boldsymbol{\Xi} = \mathbf{0} \tag{26}$$

This system may arise either from the transient problems (19)–(21) or from the direct discretization of the stationary equations.

3.3.1. Coupled PGP update-non-linearity scheme. The first scheme that we considered consists of a single loop in which the non-linearity of the problem is accounted for together with the updating of the PGP. This coupling is solved by a block Gauss–Seidel type scheme, in which a new value of the velocity and the pressure is obtained from a known value of the PGP, which is then updated from the new pressure. The scheme reads as indicated in Box 1, where subscripts refer to iteration counters.

There are several remarks to be made about this algorithm:

1. If the external forces (or part of them) are potential, the PGP could be initialized to its projection onto the finite element space. Observe also that setting $\mathbf{U}_0 = \mathbf{0}$ implies that the first effective velocity will be the solution of a Stokes problem. These comments are also valid for the following algorithms.
2. The updating of the PGP implies the solution of an algebraic system with a mass matrix. As discussed before, it can be approximated by the diagonal matrix $\mathbf{M}_\ell$, which turns out to be the most efficient choice and does not upset accuracy, as will be shown in a numerical test. However, if the 'consistent' mass matrix is to be used, the Jacobi iteration (23) provides an inexpensive way to update the PGP.
3. Most of the computational effort of each iteration is concentrated in the solution of the velocity–pressure problem, whose matrix must be rebuilt at each iteration. Thus, if the presence of the PGP increases the number of iterations to converge, it can be anticipated that the CPU time will be substantially increased.
4. In all our numerical examples, we have checked convergence in the relative discrete Euclidian norm of the arrays of unknowns. For this particular scheme there is no way to discern between the lack of convergence due to the non-linearity and that due to the PGP update.
5. In this and the following algorithms, convergence can be checked for all the variables (unless otherwise specified) or for those that may be considered relevant in a certain problem.

Box 1: Coupled PGP update-non-linearity

- Set $i=0$, $\mathbf{U}_0=\mathbf{0}$, $\Xi_0=\mathbf{0}$
  WHILE (not converged) DO:
  - Build up matrices $\mathbf{K}(\mathbf{U}_i)$, $\mathbf{G}_0$, $\mathbf{D}$, $\mathbf{L}_\tau$
  - Solve for the velocity and the pressure:
    $\mathbf{K}(\mathbf{U}_i)\mathbf{U}_{i+1}+\mathbf{G}_0\mathbf{P}_{i+1}=\mathbf{F}$
    $\mathbf{D}\mathbf{U}_{i+1}-\mathbf{L}_\tau\mathbf{P}_{i+1}=-\mathbf{D}_\tau\Xi_i$
  - Solve for the PGP:
    $\mathbf{M}\Xi_{i+1}=\mathbf{G}\mathbf{P}_{i+1}$
  - Set $i\leftarrow i+1$ and check convergence
    END

3.3.2. Nested PGP update-non-linearity scheme. This second scheme consists of a pair of nested loops: an outer loop for the PGP update and an inner loop for the non-linearity. The algorithm is presented in Box 2, where subscript i refers to the iteration counter for the non-linearity and j for the PGP update. Unknowns converged in the innermost loop have been identified by a subscript c.

This method has the disadvantage that at each of the inner iterations, a number of linear systems has to be solved with different system matrices, which have to be computed every time. Nevertheless, we hoped that as the outer iteration scheme proceeds, the number of inner iterations needed to solve the non-linearity would decrease, since the initial values for the non-linear solver progressively approach the solution. This fact could, in principle, make this method competitive with the previous one, but that was not the case.

Box 2: Nested PGP update-non-linearity

- Set $j=0$, $\mathbf{U}_{c,0}=\mathbf{0}$, $\Xi_0=\mathbf{0}$
  WHILE (not converged in j) DO:
  - Set $i=0$, $\mathbf{U}_{0,j+1}=\mathbf{U}_{c,j}$
    WHILE (not converged in i) DO:
    - Build up matrices $\mathbf{K}(\mathbf{U}_{i,j+1})$, $\mathbf{G}_0$, $\mathbf{D}$, $\mathbf{L}_\tau$
    - Solve for the velocity and the pressure:
      $\mathbf{K}(\mathbf{U}_{i,j+1})\mathbf{U}_{i+1,j+1}+\mathbf{G}_0\mathbf{P}_{i+1,j+1}=\mathbf{F}$
      $\mathbf{D}\mathbf{U}_{i+1,j+1}-\mathbf{L}_\tau\mathbf{P}_{i+1,j+1}=-\mathbf{D}_\tau\Xi_j$
    - Set $i\leftarrow i+1$ and check convergence for $\mathbf{U}$, $\mathbf{P}$ in i
      END
  - Set $\mathbf{U}_{c,j+1}=\mathbf{U}_{i,j+1}$, $\mathbf{P}_{c,j+1}=\mathbf{P}_{i,j+1}$
  - Solve for the PGP:
    $\mathbf{M}\Xi_{j+1}=\mathbf{G}\mathbf{P}_{c,j+1}$
  - Set $j\leftarrow j+1$ and check convergence in j
    END

3.3.3. Nested non-linearity-PGP update scheme. Finally, we considered a method in which the inner and outer iteration loops of the previous scheme are performed in reversed order. That is, an outer iterative method is considered to solve the non-linearity of the full problem, within which an inner iteration scheme is used for the PGP update. The resulting algorithm is shown in Box 3, where the subscript notation is the same as in Box 2.

The remarks to be made about this algorithm are the following:

1. As will be demonstrated in the numerical examples, this method turns out to be superior to any of the schemes previously considered. At each of the outer iterations, a single system matrix needs to be computed, which is the same for all the inner iterations (since it does not depend on j). Moreover, the number of inner iterations required to solve the formulation decreases as the outer iteration scheme advances, becoming very small in the last stages, as the initial approximation approaches the solution.
2. The computational cost due to the PGP update in this case is given by the cost of solving linear systems of equations with a given matrix. When direct solution techniques are employed, this is very low (of order the number of unknowns), since the matrix of the system needs to be factored only once and only forward and backward substitutions need to be performed for each j iteration. When iterative methods are used, the initial guess for this iterative method becomes closer to the converged solution as the inner loop advances, and thus less iterations of the linear system solver are required. Moreover, in this case the cost of solving the algebraic system relative to that of the whole calculation is smaller.
3. Since the innermost iteration can be viewed as a block Gauss–Seidel scheme for a linear problem, we have tested the performance of overrelaxation strategies, consisting in redefining the unknowns as

$$\mathbf{U}_{i+1,j+1} \leftarrow \omega_1 \mathbf{U}_{i+1,j+1} + (1-\omega_1)\mathbf{U}_{i+1,j}$$

$$\mathbf{P}_{i+1,j+1} \leftarrow \omega_2 \mathbf{P}_{i+1,j+1} + (1-\omega_2)\mathbf{P}_{i+1,j}$$

$$\boldsymbol{\Xi}_{i+1,j+1} \leftarrow \omega_3 \boldsymbol{\Xi}_{i+1,j+1} + (1-\omega_3)\boldsymbol{\Xi}_{i+1,j}$$

right after solving for the PGP. In these expressions, ω_i, $i = 1, 2, 3$, are numerically determined overrelaxation parameters. Even though for the linear (Stokes) problem we found a certain improvement in the number of iterations required to converge (with values of ω_i close to 1.3), this turned out to be insignificant for the non-linear problem (meaning that $\omega_i = 1$ was found to be the best choice), and we did not pursue the investigation of this possibility any further.

3.4. Transient scheme: explicit pressure gradient projection

In the previous section we have tacitly assumed that if a transient problem is being analyzed, β in (20) is taken as 1, and therefore at each time step a non-linear algebraic problem of the form (24)–(25) needs to be solved. However, it is also possible to take $\beta = 0$, the case in which the update of the PGP can be performed at the end of each time step. It is important to note that the scheme thus obtained is *unconditionally stable*. The stability estimates (15)–(16) hold

both for $\beta = 0$ and $\beta = 1$ (in the former case, there is a technical condition on δt to obtain (16); see [10]).

Box 3: Nested non-linearity-PGP update

- Set $i = 0$, $\mathbf{U}_{0,c} = 0$, $\mathbf{\Xi}_{0,c} = \mathbf{0}$
  WHILE (not converged in i) DO:
  - Build up matrices $\mathbf{K}(\mathbf{U}_{i,c})$, $\mathbf{G}_0$, $\mathbf{D}$, $\mathbf{L}_\tau$
  - Set $j = 0$, $\mathbf{U}_{i+1,0} = \mathbf{U}_{i,c}$, $\mathbf{\Xi}_{i+1,0} = \mathbf{\Xi}_{i,c}$
    WHILE (not converged in j) DO:
    - Solve for the velocity and the pressure:
      $\mathbf{K}(\mathbf{U}_{i,c})\mathbf{U}_{i+1,j+1} + \mathbf{G}_0\mathbf{P}_{i+1,j+1} = \mathbf{F}$
      $\mathbf{D}\mathbf{U}_{i+1,j+1} - \mathbf{L}_\tau\mathbf{P}_{i+1,j+1} = -\mathbf{D}_\tau\mathbf{\Xi}_{i+1,j}$
    - Solve for the PGP:
      $\mathbf{M}\mathbf{\Xi}_{i+1,j+1} = \mathbf{G}\mathbf{P}_{i+1,j+1}$
    - Set $j \leftarrow j+1$ and check convergence in j
      END
  - Set $\mathbf{U}_{i+1,c} = \mathbf{U}_{i+1,j}$, $\mathbf{P}_{i+1,c} = \mathbf{P}_{i+1,j}$, $\mathbf{\Xi}_{i+1,c} = \mathbf{\Xi}_{i+1,j}$
  - Set $i \leftarrow i+1$ and check convergence in i
    END

For the sake of completeness, we have presented in Box 4 the transient algorithm treating explicitly the PGP. The time step counter has been denoted by n and the total number of time steps by N.

Box 4: Transient scheme with explicit PGP

- Read $\mathbf{U}^0$ and set $\mathbf{\Xi}^0 = \mathbf{0}$
  FOR $n = 0, 1, 2, \ldots, N-1$ DO:
  - Solve the non-linear problem:
    $\mathbf{M}_0\delta_t\mathbf{U}^n + \mathbf{K}_{\text{conv}}(\mathbf{U}^{n+\theta})\mathbf{U}^{n+\theta} + \mathbf{K}_{\text{visc}}\mathbf{U}^{n+\theta} + \mathbf{G}_0\mathbf{P}^{n+1} = \mathbf{F}^{n+1}$
    $\mathbf{D}\mathbf{U}^{n+1} - \mathbf{L}_\tau\mathbf{P}^{n+1} = -\mathbf{D}_\tau\mathbf{\Xi}^n$
  - Project the pressure gradient:
    $\mathbf{M}\mathbf{\Xi}^{n+1} = \mathbf{U}\mathbf{P}^{n+1}$
    END

It is important to remark that this scheme has proved to be very efficient and accurate in general transient problems, but there are two small misbehaviors that need to be reported. First, its low numerical damping, even when $\theta = 1$, causes that steady state solutions are reached after a larger number of time steps than with other schemes, such as mixed div-stable velocity–pressure interpolations or the ASGS formulation. For long-term time integrations, its performance turns out to be similar to that of classical projections schemes. This point is demonstrated in a numerical example.

The second problem encountered is that the explicit treatment of the PGP produces a spurious initial pressure wave that is damped out after the first few time steps. This in particular implies that the steady state, whenever it exists, is reached faster by treating implicitly the PGP, as it will be shown in one of the numerical examples. This phenomenon was identified and analyzed in Ref. [27].

4. NUMERICAL EXAMPLES

In this section we present a study of the computational performance of all the different schemes discussed before. Our reference to decide whether the SPGP method is computationally feasible or not is the ASGS method, which is also formulated in terms of primitive variables and allows the use of equal order interpolations.

An overview of the numerical examples presented below and what they intend to demonstrate is the following:

1. A general performance test. This example serves to illustrate several facts: (1) the most efficient non-linear solver among those presented in Section 3.3 is the nested non-linearity-PGP update; (2) the use of the lumped mass matrix is preferable to the consistent one in the projection of the pressure gradient; (3) for stationary problems, the SPGP is only marginally more expensive than the ASGS method and more accurate.
2. Flow in a cavity. We obtain the steady state solution of this classical example by stepping in time to show that: (1) sometimes it is faster to treat implicitly the PGP, since the steady state is reached in fewer time steps; (2) pressure peaks using the SPGP are better captured than using the ASGS method.
3. Kovasznay flow. This is a general convergence test that shows that optimal orders of convergence are achieved for the four element types analyzed there, namely, P_1, P_2, Q_1 and Q_2 elements.
4. Newtonian flow through sinusoidally constricted tubes. This example shows the behavior of the SPGP in a more realistic problem, for which experimental and numerical reference results exist. It demonstrates that the SPGP is more accurate in such situations than the ASGS formulation.

4.1. Performance test

Let us consider a three-dimensional steady state test with analytical solution to check the behavior of the SPGP method. We take Ω as the unit cube and the force term so that the exact solution is $p = 0$ and $\boldsymbol{u}(x, y, z) = (h(z)f(x)g'(y), -h(z)f'(x)g(y), 0)$, with $f(x) = x^2(1-x)^2$, $g(y) = y^2(1-y)^2$ and $h(z) = z(1-z)$. This velocity field vanishes on $\partial\Omega$. The viscosity has been set to $\nu = 0.1$.

We have used a uniform finite element mesh of 21^3 nodal points, connected in different manners so as to obtain meshes of P_1, P_2, Q_1 and Q_2 elements. The resulting value of the element Reynolds number is not very high and for this particular example and therefore no

stabilization is needed for the convective term. The convergence tolerance has been taken as 0.01 per cent in the Euclidian norm of the arrays of nodal velocities.

The first point that is investigated is which is the best iterative scheme among those discussed in Section 3.3. For that we have considered the mesh of P_1 elements and approximated the mass matrix by $\mathbf{M}_\ell$. The number of iterations required to converge and the CPU needed are displayed in Table I. Since the relative importance of the CPU time needed to construct the matrices and to solve the algebraic system depends on the algorithms used in both cases, we have split these two contributions to the total CPU. In this example, the algorithm employed to construct the matrices is that proposed in [20], whereas the linear system has been solved with a standard direct solver.

From Table I it is seen that when the non-linearity (NL in Table I) and the PGP update (simply identified by PGP in Table I) are coupled, the latter clearly drives the iterative process, making the scheme inefficient due to the need for solving different systems at each iteration. The same is true for the nested PGP-NL, although the situation is now aggravated by the inner iterations. The best method is the nested NL-PGP. The cost of the inner iterations is extremely low, since the matrix of the system needs not to be recomputed and it needs to be factored only once. The CPU time increase with respect to the ASGS method is only 7 per cent for the linear solver and 3 per cent for the construction of the matrices.

The second point tested is whether it is preferable to use $\mathbf{M}$ or $\mathbf{M}_\ell$. As before, we have used the mesh of P_1 elements and now the nested NL-PGP scheme. Table II shows that the convergence of the inner iterations is better using $\mathbf{M}_\ell$ and the CPU time is smaller. Moreover, the accuracy is not deteriorated and, in this case, it is even slightly higher using the lumped mass matrix. The error has been measured in the discrete ℓ^2 norm, which is defined as

Table I. Test to determine the best iterative scheme.

Method	Outer iterations	Inner iterations	Matrices CPU	Solver CPU	Total CPU
ASGS	3	—	100	100	100
coupled SPGP	6	—	181	182	182
Nested SPGP, PGP-NL	6	(3, 3, 2, 2, 2, 2)	389	404	402
Nested SPGP, NL-PGP	3	(6, 3, 2)	103	107	106

Lumped mass matrix $\mathbf{M}_\ell$, P_1 element.

Table II. Test to compare $\mathbf{M}$ and $\mathbf{M}_\ell$.

Method	Outer iterations	Inner iterations	Matrices CPU	Solver CPU	Total CPU	ℓ^2 error
ASGS	3	—	100	100	100	6.32×10^{-2}
SPGP, $\mathbf{M}$	3	(6, 6, 4)	111	107	108	4.32×10^{-2}
SPGP, $\mathbf{M}_\ell$	3	(6, 3, 2)	103	107	106	4.01×10^{-2}

For the SPGP, the nested NL-PGP scheme is used, P_1 element.

$$E = \left[\sum_{n=1}^{n_{\text{pts}}} \sum_{i=1}^{2} (\mathsf{U}_i^a - u_i(\boldsymbol{x}^a))^2 \right]^{1/2} \left[\sum_{a=1}^{n_{\text{pts}}} \sum_{i=1}^{2} (u_i(\boldsymbol{x}^a))^2 \right]^{-1/2}$$

where n_{pts} is the total number of nodal points, U_i^a is the i-th component of the nodal velocity at node a and $\boldsymbol{x}^a$ are the co-ordinates of this node.

For all the interpolations we have tested the behavior of $\mathbf{M}$ and $\mathbf{M}_\ell$ is similar, except for the P_2 element. The reason may be due to the fact that for this particular case the diagonal mass matrix obtained from nodal quadrature has zero diagonal entries (those corresponding to the vertex nodes). To obtain a non-singular diagonal matrix, we split the quadratic tetrahedra into linear tetrahedra and build up the lumped mass matrix from this splitting. The behavior of the inner iterations depending on $\mathbf{M}$ and $\mathbf{M}_\ell$ is highly problem dependent.

A comparison of the CPU time and accuracy between the ASGS and the SPGP, using in this case the nested NL-PGP scheme and $\mathbf{M}_\ell$, is shown in Table III. It is seen there that the SPGP is slightly more expensive in all the cases and also slightly more accurate for first-order elements. The behavior of the P_2 element may be due to the construction of $\mathbf{M}_\ell$, and for the Q_2 it is seen that accuracy is much higher using the SPGP formulation.

4.2. Flow in a cavity

In this example we solve the classical cavity flow problem in the unit square. The velocity is prescribed to (1, 0) at the lid, including the corners (leaky case) and to (0, 0) on the rest of the boundary. The domain is discretized using a mesh of 2888 P_1 elements and 1521 nodal points, uniformly refined near the boundaries.

Let us take $\nu = 0.001$ first. The resulting Reynolds number is $Re = 1000$, which is small enough to ensure that a steady state solution exists and can be well captured without stabilizing convection. In this example the steady state is reached through a transient evolution from rest, both for the ASGS and the SPGP methods. The parameter θ is set to 1 in (10), although, as mentioned before, it is often interesting to replace the convective term in this equation by $(\boldsymbol{u}_h^n \cdot \nabla \boldsymbol{u}_h^{n+1}, \boldsymbol{v}_h)$. Thus, we take it as $(\boldsymbol{u}_h^{n+\gamma} \cdot \nabla \boldsymbol{u}_h^{n+1}, \boldsymbol{v}_h)$ and consider the cases $\gamma = 0$ and $\gamma = 1$.

Table III. Test to compare the element behavior.

Element and method	Outer iterations	Inner iterations	Matrices CPU	Solver CPU	Total CPU	ℓ^2 error
ASGS, P_1	3	—	100	100	100	6.32×10^{-2}
SPGP, P_1	3	(6, 3, 2)	103	107	106	4.01×10^{-2}
ASGS, Q_1	3	—	183	246	233	9.38×10^{-3}
SPGP, Q_1	3	(6, 4, 2)	190	263	250	7.02×10^{-3}
ASGS, P_2	6	—	159	237	227	3.53×10^{-1}
SPGP, P_2	6	(7, 5, 5, 4, 4, 2)	167	258	247	4.60×10^{-1}
ASGS, Q_2	5	—	412	829	798	5.01×10^{-2}
SPGP, Q_2	5	(6, 4, 3, 2, 2)	433	904	872	5.56×10^{-3}

Lumped mass matrix $\mathbf{M}_\ell$.

Table IV. Number of time steps to reach the steady state for the cavity flow problem (case $Re = 1000$).

Method	Number of time steps	Matrices CPU	Solver CPU	Total CPU
ASGS, $\gamma = 0$	14	59.4	6.7	66.5
ASGS, $\gamma = 1$	6	89.4	10.1	100.0
SPGP, $\gamma = 0$, $\beta = 0$	26	129.8	13.0	144.2
SPGP, $\gamma = 0$, $\beta = 1$	16	81.0	9.7	92.2
SPGP, $\gamma = 1$, $\beta = 0$	22	753.6	74.4	829.4
SPGP, $\gamma = 1$, $\beta = 1$	7	128.5	16.0	145.9

The number of time steps needed to reach the steady state with $\delta t = 100$, as well as the required CPU time, are reported in Table IV. The steady state tolerance has been set to 10^{-4}, normalized with $\delta_t \mathbf{U}^1$, and the convergence tolerance to 10^{-3}. It can be observed that the case $\gamma = 0$ requires more time steps due to the smaller numerical damping of the scheme, even though it is less time consuming that the non-linear case $\gamma = 1$. For the SPGP method, it turns out that the case $\beta = 1$ needs less CPU time than $\beta = 0$, especially when $\gamma = 1$. This is due to the faster convergence to the steady state when the PGP is treated implicitly, case which does not suffer from the spurious pressure wave of the explicit case mentioned earlier.

In this relatively small problem, the increase of CPU time of the SPGP with respect to the ASGS is bigger than in the first example. Considering the cases with $\beta = 1$, it is 38.6 per cent higher for $\gamma = 0$ and 45.9 per cent for $\gamma = 1$. This is due to the bigger relative cost of the forward and backward substitutions (needed at each PGP update) with respect to the matrix factorization in the direct solver that we have used (see remark 2 to Box 3).

This example also serves to point out the difference in the pressure peaks obtained using the ASGS and the SPGP methods. Table V shows the difference between the maximum and the minimum pressure values obtained for $\nu = 0.001$ and $\nu = 0.1$. Results are more 'diffusive' using the ASGS method, as it can be also seen from the curvature of the iso-pressure lines near the boundary in Figures 1 and 2, which in the ASGS case tend to be slightly orthogonal to it. Nevertheless, this does not affect the velocity solution, which is very similar in both cases.

4.3. Kovasznay flow

In order to check numerically the optimal orders of accuracy given in (14), we considered a problem introduced by Kovasznay (see [21]), modeling laminar flow behind a two-dimensional

Table V. Maximum pressure differences $p_{\max} - p_{\min}$ for the cavity flow problem.

Method	$Re = 10$	$Re = 1000$
ASGS	27.131	0.632
SPGP	44.093	0.866

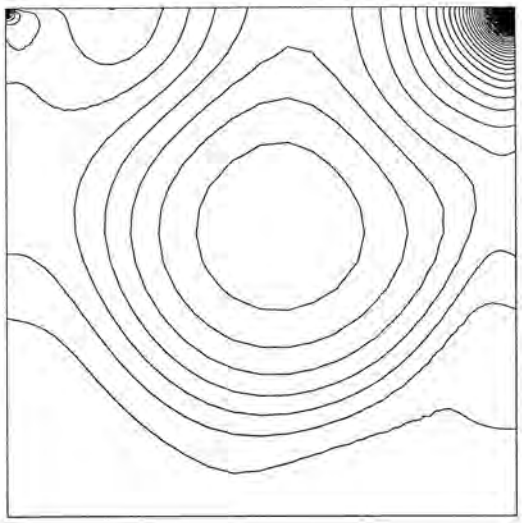

Figure 1. Pressure contours for the flow in a cavity (case $Re = 1000$). SPGP method.

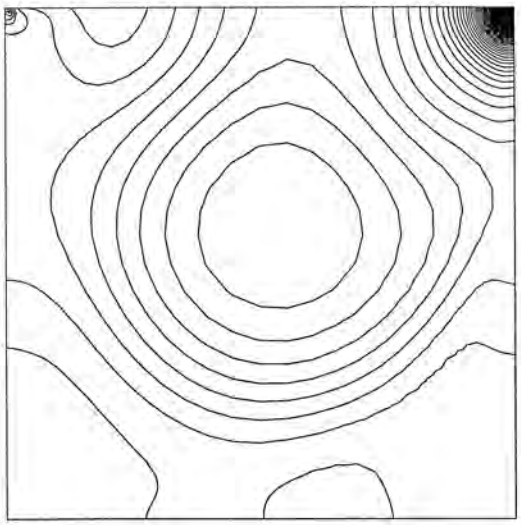

Figure 2. Pressure contours for the flow in a cavity (case $Re = 1000$). ASGS method.

grid, in which an analytical solution of the steady incompressible Navier–Stokes equations with no forcing term is available. The velocity solution $\boldsymbol{u} = (u, v)$ is given by

$$u(x, y) = 1 - e^{\lambda x} \cos(2\pi y)$$

$$v(x, y) = \frac{\lambda}{2\pi} e^{\lambda x} \sin(2\pi y)$$

for $(x, y) \in \mathbb{R}^2$, whereas the pressure is $p(x, y) = p_0 - e^{\lambda x}/2$, where p_0 is an arbitrary constant and the parameter λ is given in terms of the Reynolds number Re by $\lambda = Re/2 - (Re^2/4 + 4\pi^2)^{1/2} < 0$. This flow problem was solved numerically in [22,23] for a value of $Re = 40$. We solved it in the domain $\Omega = [-1/2, 1] \times [-1/2, 1/2]$ for that value of the Reynolds number (that is, for $\nu = 0.025$), with the elements P_1, Q_1, P_2 and Q_2, and on four different uniform meshes, made up with 19×13, 31×21, 43×29 and 61×41 nodes respectively. In all cases, the solution was obtained by the nested non-linearity-PGP update scheme with a Newton–Raphson approximation of the convective term, starting from the fluid at rest, but for the prescribed boundary conditions (which were given by the value of the analytical solution at the boundary). The tolerance for convergence in the iteration for non-linearity was 10^{-4}, and the same value was taken for the tolerance of the inner iteration. It took five iterations of Newton–Raphson's method in all cases to find the solution. The only exception was the P_2 element case with the finer 61×41 mesh: Newton–Raphson's method diverged in that case; we used Picard's iteration instead, which required nine iterations to find the solution for the same values of the tolerances. The number of inner iterations decreased with the outer iteration scheme in all cases, from about 100 in the first iteration, in the worst cases, to one in the fifth iteration in all cases.

We then computed the exact errors $\|\boldsymbol{u} - \boldsymbol{u}_h\|$, (velocity error in L^2), $\|\nabla \boldsymbol{u} - \nabla \boldsymbol{u}_h\|$, (velocity error in H^1) $\|p - p_h\|$ (velocity error in L^2) and $\|\nabla p - \nabla p_h\|$ (pressure error in H^1) for each mesh and element. The results obtained can be seen in Figure 3 as a function of the mesh size, where we have included the errors obtained for the ASGS method with a Q_1 element interpolation for comparison. As can be observed, optimal orders of accuracy were found in all cases. Those of the velocity solution are specially sharp, whereas for the pressure there seems to be a gain of one order of accuracy both in the pressure and its gradient. In particular, the convergence of the pressure gradient was not ensured by the theory for linear and bilinear elements.

Moreover, it is clearly seen that the ASGS method, although optimal in all cases, produces less accurate results than our method, specially for the pressure. The Q_2 element provides the most accurate results.

4.4. Newtonian flow through sinusoidally constricted tubes

This numerical example concerns a rather well-studied flow with recirculating regions but without singularities. Precise measurements and simulations [24,25] exist so as to compare the SPGP and ASGS methods with regard to accuracy, with emphasis on that of global quantities.

The problem consists of a periodic flow through a tube with radius dependent on x_1 as

$$r(x_1) = R_0\left(1 + \alpha \cos\frac{2\pi x_1}{\lambda}\right)$$

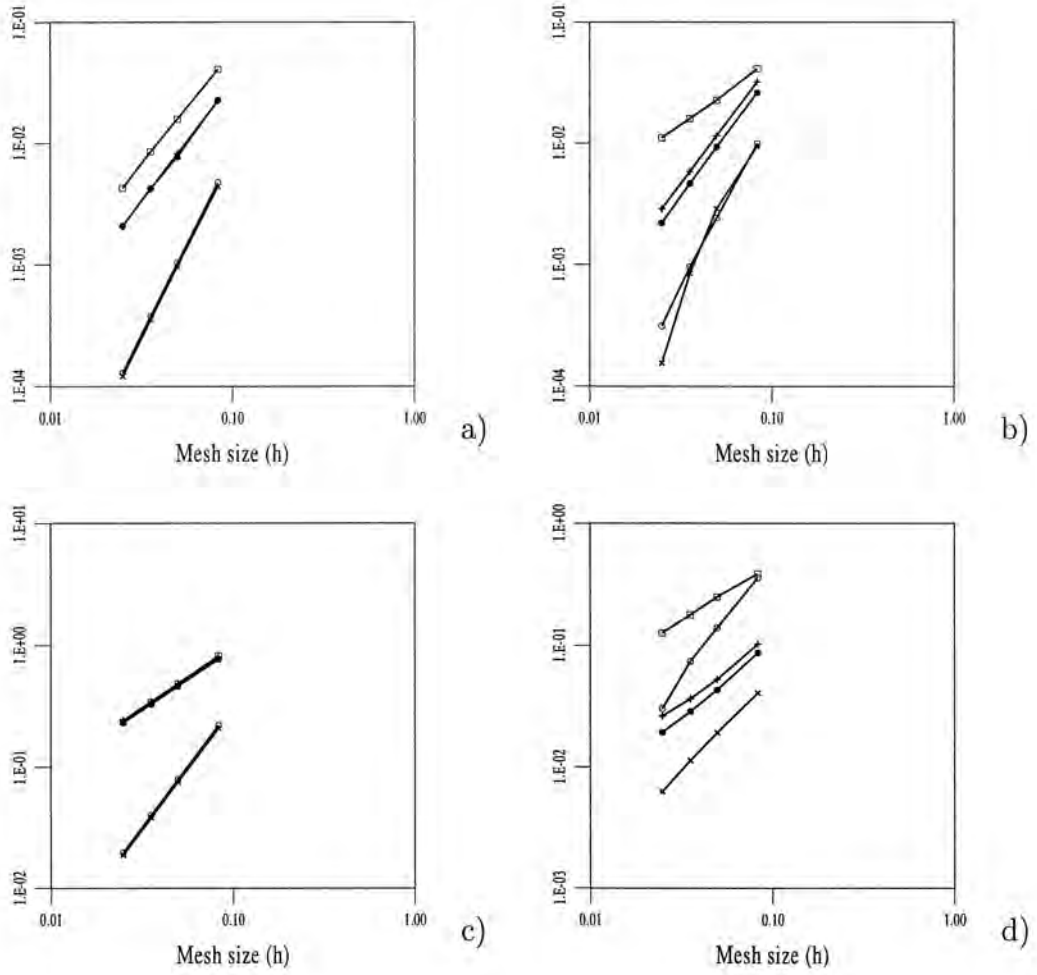

Figure 3. Kovasznay flow: (a) velocity error in L^2, (b) pressure error in L^2; (c) velocity error in H^1; (d) pressure error in H^1. $+$, P_1 element; ●, Q_1 element; ○, P_2 element; $\times$, Q_2 element; □, ASGS method, Q_1 element.

and thus the geometry involves two non-dimensional quantities: The constriction ratio α, and the non-dimensional period $\Lambda = 2\pi R_0/\lambda$. In [24] detailed results are provided for the case $\alpha = 0.3$, $\Lambda = 1$, that we will consider in the following. The quantity of interest is the friction factor f, usually considered within the product

$$f\,Re = \frac{2\pi R_0^4 \overline{\nabla p}}{\mu Q}$$

where $\overline{\nabla p}$ is the mean pressure gradient and Q the flow rate. In the steady case and for given α and Λ, $f\,Re$ is a function of the Reynods number alone, defined as

$$Re = \frac{2Q}{\pi \nu R_0}$$

Experiments indicate that the flow is steady and axisymmetric up to $Re = 200$ or greater, depending on the geometry. Our computations are thus axisymmetric, stepping in time until a steady state is reached.

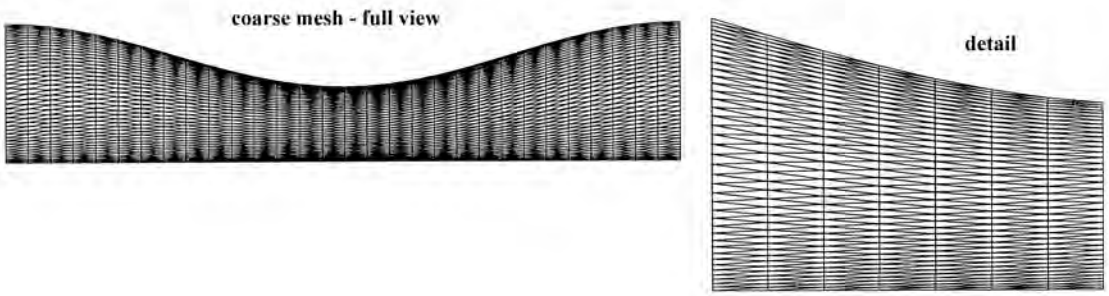

Figure 4. Flow through sinusoidal tubes: 30×40 mesh.

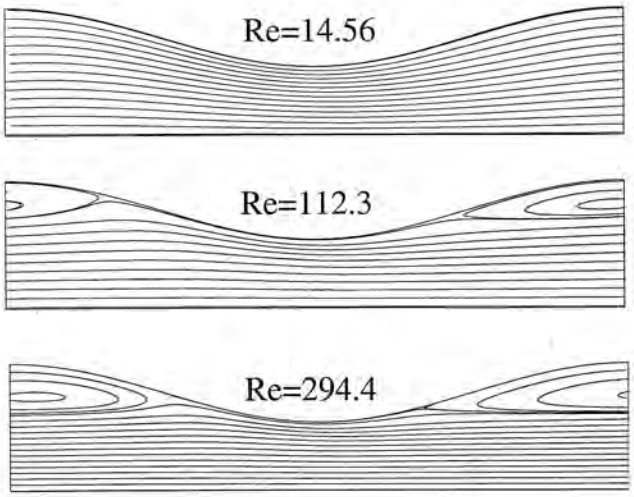

Figure 5. Flow through sinusoidal tubes: flow pattern for different Re.

We consider two meshes: A coarse one consisting of 30×40 nodes that is shown in Figure 4, and a fine one consisting of 120×80 nodes used as reference. We investigate the range $0 < Re < 300$ using the SPGP method and the ASGS method. For the former we consider two treatments of the momentum equation, namely Galerkin and SUPG, denoted respectively by SPGP + GAL and SPGP + SUPG. The flow structure, shown by means of the streamline pattern, is shown in Figure 5 as computed on the 120×80 mesh using SPGP + GAL. Detachment regions that grow with Re are evident.

In Figures 6 and 7 we compare the periodic part of the pressure fields obtained with the different methods with that were obtained using the 120×80 mesh. In the case $Re = 0$ the SPGP method is more accurate (this is most noticeable from the extreme values of the pressure, which suggests an overdiffusive behavior of ASGS). For $Re \sim 295$ the differences between the pressure fields obtained with SPGP + GAL, SPGP + SUPG and ASGS are less significant. However, ASGS remains the one for which $p_{max} - p_{min}$ is smallest.

Quantitative comparison between the computed friction factors is carried out in Figures 8 and 9 (which is a detail of the former for lower values of Re). Excellent agreement exists between numerical results from the literature [24,25] and the results obtained on the 120×80 mesh. The plots prompt the definition of two regimes in what concerns numerical behavior of the methods. In the low-Re regime ($Re < 25$) momentum upwinding is negligible, SPGP + GAL coincides with SPGP + SUPG, both being significantly more accurate than the ASGS method. In the high-Re regime ($Re > 100$) momentum upwinding is the dominant source of error and the predictions of SPGP + SUPG and ASGS are very close. In the whole range, of $f\,Re$ predicted by SPGP + GAL agrees within 0.2 per cent with the reference computation.

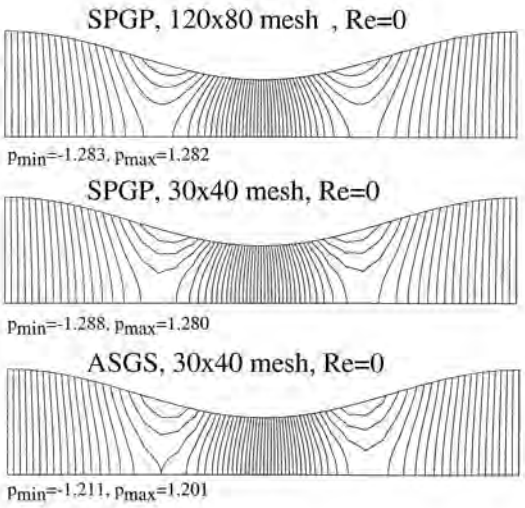

Figure 6. Flow through sinusoidal tubes: periodic part of the pressure field as computed by the SPGP and ASGS methods for $Re = 0$. In this case SPGP + GAL and SPGP + SUPG are equivalent.

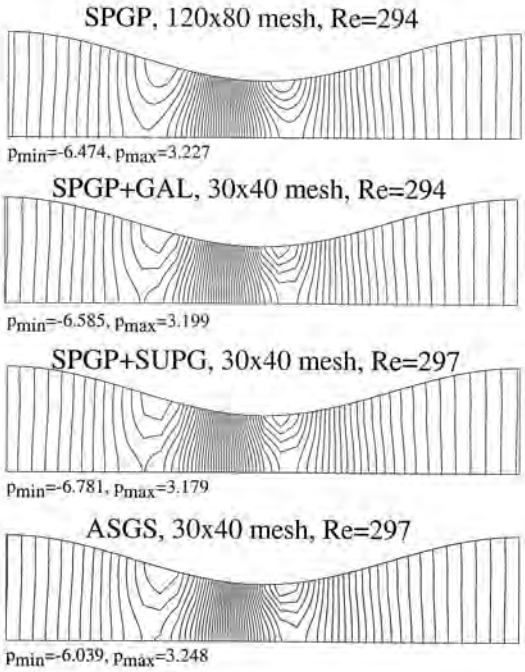

Figure 7. Flow through sinusoidal tubes: periodic part of the pressure field as computed by the SPGP and ASGS methods for $Re \sim 295$.

5. CONCLUSIONS

In this paper we have fully described the SPGP method and its implementation aspects, which have been tested through numerical experiments. The main conclusions that may be drawn form these are the following:

- If a non-linear iterative solver needs to be used solving also for the PGP, the best option turns out to be a pair of nested loops, the outermost of which accounts for the linearization of the convective term and the innermost for the updating of the PGP. The inner iterations are very inexpensive relative to the global cost of a non-linear transient analysis, since the matrix of the algebraic system remains unchanged.
- In general, it is preferable to use the lumped mass matrix rather than the consistent one. From numerical examples we have found that the accuracy of the scheme is similar in both cases, whereas the computational cost is lower using the lumped mass matrix. This is due not only to the fact that the projection step is trivial in this case, but also to the smaller number of iterations required for convergence in the iterative schemes analyzed to segregate the calculation of the PGP from the velocity and the pressure.

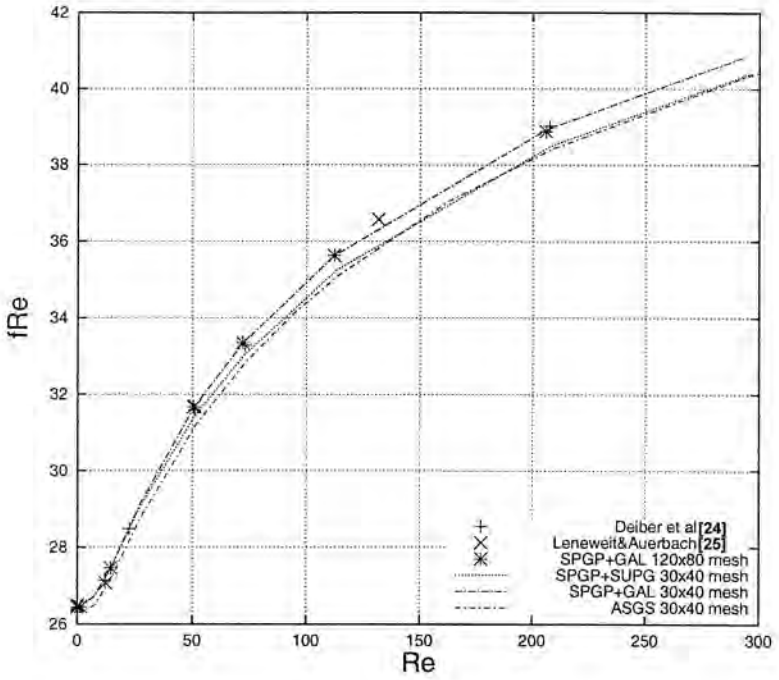

Figure 8. Flow through sinusoidal tubes: product $f\,Re$ versus Re for the different methods as compared to other authors and to the reference computation.

- For transient problems, the explicit treatment of the PGP yields what to our knowledge is the *computationally simplest* finite element formulation which allows equal velocity–pressure interpolations. A pressure Laplacian-like term needs to be introduced in the left-hand side of the equations and the divergence of the projected pressure gradient at the previous time step needs to be added as a force term of the continuity equation. This ensures consistency and does not deteriorate stability. However, when a steady state is sought, more time steps than for example with the ASGS need to be performed. This reluctance to reach the steady state is similar to that found in some fractional step methods which solve, at the steady state, an equation similar to (20).
- The accuracy of the scheme is certainly one of its most important features. Even in stationary calculations, when the cost of the PGP update is not negligible, it might be worth using it. In particular, the pressure near the boundary is more accurate than with the ASGS method or classical projection schemes.

As a general conclusion, we think that the SPGP method is an accurate finite element formulation which originates from an innovative stabilization concept. This may stimulate the development of new stabilization techniques in other areas of computational mechanics.

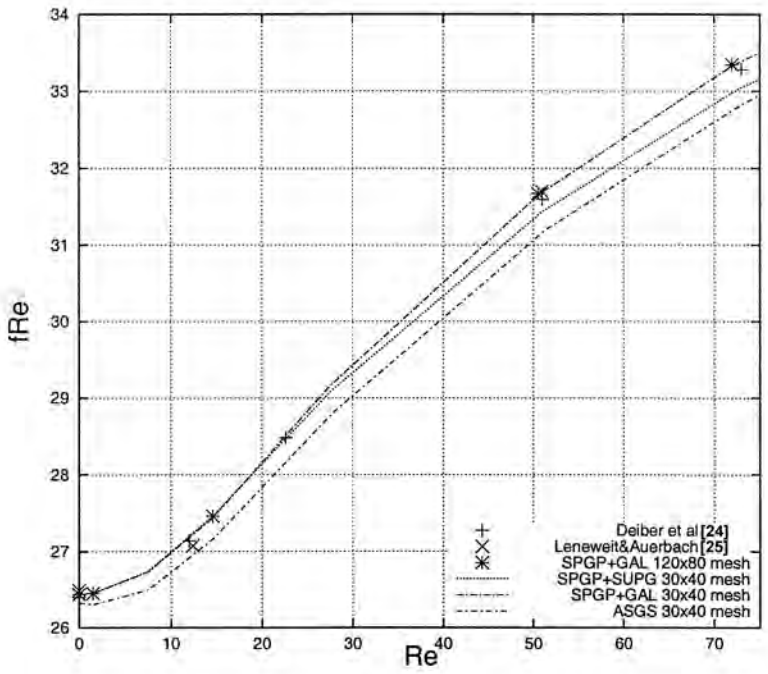

Figure 9. Detail of the low-*Re* region of Figure 8.

On the other hand, its weak point is its robustness in certain situations, which deserves further research.

REFERENCES

1. Brezzi F, Douglas J. Stabilized mixed methods for the Stokes problem. *Numerische Mathematik* 1988; **53**: 225–235.
2. Douglas J, Wang J. An absolutely stabilized finite element method for the Stokes problem. *Mathematics of Computation* 1989; **52**: 495–508.
3. Hughes TJR, Franca LP, Balestra M. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška–Brezzi condition: a stable Petrov–Galerkin formulation for the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**: 85–99.
4. Franca LP, Hughes TJR. Convergence analyses of Galerkin least-squares methods for advective-diffusive forms of the Stokes and incompressible Navier–Stokes equations. *Comupter Methods in Applied Mechanics and Engineering* 1993; **105**: 285–298.
5. Franca L, Stenberg R. Error analysis of some Galerkin least-squares methods for the elasticity equations. *SIAM Journal on Numerical Analysis* 1991; **28**: 1680–1697.
6. Bochev P, Cai Z, Manteuffel TA, McCormick SF. Analysis of velocity-flux first-order system least-squares principles for the Navier–Stokes equations: Part I. *SIAM Journal on Numerical Analysis* 1998; **35**: 990–1009.
7. Fortin M, Boivin S. Iterative stabilization of the bilinear velocity–constant pressure element. *International Journal for Numerical Methods in Fluids* 1990; **10**: 125–140.
8. Silvester PJ, Kechkar N. Stabilised bilinear-constant velocity–pressure finite elements for the conjugate gradient solution of the Stokes problem. *Computer Methods in Applied Mechanics and Engineering* 1990; **79**: 71–86.

9. Codina R, Blasco J. A finite element formulation for the Stokes problem allowing equal velocity–pressure interpolation. *Computer Methods in Applied Mechanics and Engineering* 1997; **143**: 373–391.
10. Codina R, Blasco J. Stabilized finite element method for the transient Navier–Stokes equations based on a pressure gradient projection. *Computer Methods in Applied Mechanics and Engineering* 2000; **182**: 287–310.
11. Codina R, Blasco J. Analysis of a pressure-stabilized finite element approximation of the stationary Navier–Stokes equations. *Numerische Mathematik* 2000; **87**: 59–81.
12. Habashi W, Peeters M, Robichaud M, Nguyen VN. A fully-coupled finite element algorithm, using direct and iterative solvers, for the incompressible Navier–Stokes equations. In *Incompressible Computational Fluid Dynamics*, Gunzburger M, Nicolaides R (eds). Cambridge University Press: Cambridge, 1993; 151–182.
13. Hughes TJR, Feijóo GR, Mazzei L, Quincy JB. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering* 1998; **166**: 3–24.
14. Heywood JG, Rannacher R. Finite element approximation of the nonstationary Navier–Stokes problem. IV: Error analysis for second-order time discretization. *SIAM Journal on Numerical Analysis* 1990; **27**: 353–384.
15. Franca LP, Frey SL. Stabilized finite element methods: II. The incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**: 209–233.
16. Codina R. Stabilization of incompressibility and convection through orthogonal subscales in finite element methods. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**: 1579–1599.
17. Codina R, Vázquez M, Zienkiewicz OC. A general algorithm for compressible and incompressible flow—Part III. The semi-implicit form. *International Journal for Numerical Methods in Fluids* 1998; **27**: 13–32.
18. Gresho PM. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part I: Theory. *International Journal for Numerical Methods in Fluids* 1990; **11**: 587–620.
19. Gresho PM, Chan ST, Lee RL, Upson CD. A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 2: Applications. *International Journal for Numerical Methods in Fluids* 1984; **4**: 619–640.
20. Codina R. A nodal-based implementation of a stabilized finite element method for incompressible flow problems. *International Journal for Numerical Methods in Fluids* 2000; **33**: 737–766.
21. Kovasznay IG. Laminar flow behind a two-dimensional grid. *Proceedings of the Cambridge Philosophical Society* 1948; **44**: 58–63.
22. Sherwin SJ, Karniadakis GE. A triangular spectral element method; application to the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1995; **123**: 189–229.
23. Karniadakis GE, Israeli M, Orzag SE. High order splitting methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1991; **59**: 414–443.
24. Deiber JA, Peirotti M, Bortolozzi R, Durelli R. Flow of Newtonian fluids through sinusoidally constricted tubes: numerical and experimental results. *Chemical Engineering Communications* 1992; **117**: 241–262.
25. Leneweit G, Auerbach D. Detachment phenomena in low Reynolds number flows trough sinusoidally constricted tubes. *Journal of Fluid Mechanics* 1999; **387**: 129–150.
26. Codina R. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**: 2681–2706.
27. Buscaglia GC, Basombrio FG, Codina R. Fourier analysis of an equal-order incompressible flow solver stabilized by pressure-gradient projection. *International Journal for Numerical Methods in Fluids* 2000; **34**: 65–92.