Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 193 (2004) 2139–2154
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A stabilized edge-based implicit incompressible flow formulation

O. Soto *, R. Löhner, J. Cebral, F. Camelli

SCS/Laboratory for Computational Fluid Dynamics, George Mason University, MS 4C7, 4400 University Drive, Fairfax, VA 22030-4444, USA

Received 21 March 2003; received in revised form 9 October 2003; accepted 2 January 2004

Abstract

An edge-based implementation of an implicit, monolithic, finite element (FE) scheme for the solution of the incompressible Navier–Stokes (NS) equations is presented. The original element formulation is based on the pressure stability properties of an implicit second-order in time fractional step (FS) method, which is conditionally stable. The final monolithic scheme preserves the second-order accuracy of the FS method, and is unconditionally stable. Furthermore, it can be demonstrated that the final pressure stabilizing term is practically the same fourth-order pressure term added by some authors (but following different arguments) to obtain high order accurate results, and that the final discretized convective terms are formally a second-order discretization of the respective continuous one.

The development of the edge implementation is supported by two criteria: the properties of the element based one, which has already been extensively tested and for which convergence and stability analysis has already been presented, and on the enforcement of global conservation and symmetry at the discrete level. A monotonicity preserving term which decreases the discretization order in sharp gradient regions to avoid localized oscillations (overshoots and undershoots), is formulated and tested. Some numerical examples and experimental comparisons are presented.

Keywords: Incompressible flows; Stabilized methods; Edge-based schemes

1. Introduction

Among the schemes developed over the last decade for the solution of the incompressible NS equations (monolithic schemes [13,16], projection or fractional step (FS) schemes [3,8,18,20–22,25], artificial compressibility (AC) [7,15,17,23,24,28], pre-conditioning of the compressible NS equations [6,34,35], etc.) the FS schemes yield highly accurate, pressure-stable results by integrating in an explicit manner the advective terms of the NS equations. However, the timestep imposed by the smallest elements may be orders of

* Corresponding author.
E-mail address: sorlando@gmu.edu (O. Soto).
URL: http://www.science.gmu.edu/~rlohner/.

0045-7825/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2004.01.018

magnitude smaller than the timestep required to obtain time-accurate results. For many classes of problems, e.g. biological flows (blood and air flow) and environmental flows (contaminant release), this implies tens of thousands of timesteps per simulation, rendering the schemes impractical. Most of the artificial compressibility and pre-conditioned schemes suffer from the same shortcoming.

On the other hand, the monolithic schemes treat, in general, the advective term in an implicit manner, which avoids these disadvantages. Nevertheless, these methods are very expensive from a computational point of view: the velocity and pressure discrete equations are coupled. For this reason, an implicit monolithic but uncoupled scheme was developed, which is unconditionally stable, and which preserves the accuracy and stability of a second-order FS method [11,32].

This method was implemented using an element-based discretization, which involves a loop over the elements, the computations of the element contributions (to the system matrix and to the force vector), and the assembly of these to the global arrays. Such procedure implies the recalculation of the mass, Laplacian, and gradient matrices in each iteration of each timestep, which is highly time consuming. In the implementation that is presented here, the mass, gradients, and Laplacian matrices are computed and stored only once at the beginning of the run (or each time a remeshing is done). All the left-hand-side (LHS) and right-hand-side (RHS) terms involved in the final algebraic system are computed by looping over the points of the mesh, and then over the points connected to a given point. The storage of the final system of equations, and of the different edge-based arrays (mass, Laplacian and gradients) is done by using a standard *compressed sparse row* (CSR) format [29]. In this way, the computation of the different terms can be parallelized in a straightforward manner, since the mesh edge ij is touched only when the loop over the points goes through the point i (in this work the edge ji is different from the edge ij).

The rest of the paper is organized as follows: In Section 2 the standard element-based stabilized formulation is briefly summarized. In Section 3 the edge-based implementation is presented, and some aspects of its development are discussed. Section 4 is dedicated to show some 2D and 3D numerical examples, and some comparisons with the element-based scheme. Finally, in Section 5 some conclusions are drawn.

2. Element-based scheme

The continuous incompressible NS equations to be solved can be written as:

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

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

where Ω is the flow domain, t is the time variable, $(0,t_f)$ the time interval for the simulation, $\boldsymbol{u}$ the velocity field, ∇ the gradient operator, ν the kinematic viscosity, Δ the Laplacian operator, p the pressure and $\boldsymbol{f}$ the external body forces (i.e. the gravity and the Boussinesq forces).

Let $\boldsymbol{\sigma}$ be the viscous stress tensor and $\boldsymbol{n}$ the unit outward normal to the boundary $\partial\Omega$. Denoting by an overbar prescribed values, the boundary conditions for (2) to be considered here are:

$$\begin{aligned} &\boldsymbol{u} = \bar{\boldsymbol{u}} \text{ on } \Gamma_{\mathrm{du}}, \quad p = \bar{p} \quad \text{and} \quad \boldsymbol{n}\cdot\boldsymbol{\sigma} = \bar{\boldsymbol{t}} \text{ on } \Gamma_{\mathrm{nu}},\\ &\boldsymbol{u}\cdot\boldsymbol{n} = \bar{u}_n, \quad \boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{g}_1 = \overline{t_1} \quad \text{and} \quad \boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{g}_2 = \overline{t_2} \text{ on } \Gamma_{\mathrm{mu}} \end{aligned} \tag{3}$$

for $t\in(t_0,t_f)$. The boundary $\partial\Omega$ has been considered split into three sets of disjoint components Γ_{du}, Γ_{nu} and Γ_{mu}, the latter being the part where mixed conditions are prescribed (i.e. the normal velocity and the tangent stresses). Vectors $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ (for the three-dimensional case) span the space tangent to Γ_{mu}. Finally, Γ_{du} and Γ_{nu} are the two disjoint components of $\partial\Omega$ where Dirichlet and Neumann boundary conditions for the velocity are prescribed. Initial conditions have to be appended to problem (2) and (3).

If the above equations are discretized in time using an implicit trapezoidal rule ($\theta = 1.0$ backward-Euler or $\theta = 0.5$ Crank–Nicholson), in space using the *orthogonal subscales stabilization* (OSS) type scheme presented in [12,14,32,33], and a Picard linearization to deal with the non-linear terms, the following expression is obtained: Given $\boldsymbol{u}_h^n$, find $(\boldsymbol{u}_h^{n+1}, p_h^{n+1}, \boldsymbol{\pi}_h^{n+1}, \boldsymbol{\xi}_h^{n+1})$ in $\boldsymbol{V}_h \times Q_h \times \tilde{\boldsymbol{V}}_h \times \tilde{\boldsymbol{V}}_h$ such that

$$\begin{aligned}&\frac{1}{\theta\delta t}(\boldsymbol{u}_h^{n+\theta,i} - \boldsymbol{u}_h^n, \boldsymbol{v}_h) + (\boldsymbol{u}_h^{n+\theta,i-1} \cdot \nabla \boldsymbol{u}_h^{n+\theta,i}, \boldsymbol{v}_h) + (\nu \nabla \boldsymbol{u}_h^{n+\theta,i}, \nabla \boldsymbol{v}_h) - (p_h^{n+1,i-1}, \nabla \cdot \boldsymbol{v}_h)\\ &\quad + (\tau(\boldsymbol{u}_h^{n+\theta,i-1} \cdot \nabla \boldsymbol{u}_h^{n+\theta,i} - \beta^{n+1,i-1}\boldsymbol{\pi}_h^{n+\theta,i-1}), \boldsymbol{u}_h^{n+\theta,i-1} \cdot \nabla \boldsymbol{v}_h)\\ &\quad = (\boldsymbol{f}^{n+\theta}, \boldsymbol{v}_h) + (\boldsymbol{\sigma}^{n+\theta,i-1} \cdot \boldsymbol{u}, \boldsymbol{v}_h)_{\Gamma_{\mathrm{nu}}},\end{aligned} \tag{4}$$

$$\delta t(\nabla p_h^{n+1,i} - \nabla p_h^{n+1,i-1}, \nabla q_h) + (\tau(\nabla p_h^{n+1,i} - \beta^{n+1,i-1}\boldsymbol{\xi}^{n+1,i-1}), \nabla q_h) = -(\nabla \cdot \boldsymbol{u}_h^{n+1,i}, q_h), \tag{5}$$

$$(\boldsymbol{\pi}_h^{n+\theta,i}, \tilde{\boldsymbol{v}}_h) = (\boldsymbol{u}_h^{n+\theta,i} \cdot \nabla \boldsymbol{u}_h^{n+\theta,i}, \tilde{\boldsymbol{v}}_h), \tag{6}$$

$$(\boldsymbol{\xi}_h^{n+1,i}, \tilde{\boldsymbol{v}}_h) = (\nabla p_h^{n+1,i}, \tilde{\boldsymbol{v}}_h) \tag{7}$$

$\forall (\boldsymbol{v}_h, q_h, \tilde{\boldsymbol{v}}_h, \tilde{\boldsymbol{v}}_h) \in \boldsymbol{V}_h \times Q_h \times \tilde{\boldsymbol{V}}_h \times \tilde{\boldsymbol{V}}_h$. The superscripts n and i stand for the timestep and for the iteration counter into each timestep, respectively, the subscript h refers to the discrete problem, δt is the timestep size, and β is the numerical switch given below (see Section 4). The functional spaces are the standard finite element ones, and the functional form $(\cdot,\cdot)$ is defined as:

$$(\boldsymbol{a}, \boldsymbol{b}) = \int_\Omega \boldsymbol{a} \cdot \boldsymbol{b}\, \mathrm{d}\Omega, \qquad (\boldsymbol{a}, \boldsymbol{b})_\Gamma = \int_\Gamma \boldsymbol{a} \cdot \boldsymbol{b}\, \mathrm{d}\Gamma.$$

The intrinsic time τ has been introduced into the discrete forms to indicate that it has to be computed element by element. The stability and convergence analysis dictates that it must be computed as [2,9,16]:

$$\tau = \frac{h^2}{4\nu + 2|u|h}, \tag{8}$$

where h and $|u|$ are the typical element size and velocity, respectively.

The scheme (4)–(7) is second-order accurate in time if a Crank–Nicholson time discretization is used, and also in space if standard linear elements are employed (see [11,32]). The solution in each timestep is performed using a staggered iterative scheme: First the non-symmetric discrete momentum equation (4) is solved using a standard GMRES method with diagonal pre-conditioning. Then, the pressure equation (5) is computed by using a conjugate gradient with incomplete LU pre-conditioning (CG-ILU) solver for isotropic meshes, or a CG-Linelet solver for highly stretched grids (see [33] for details). The standard lumped mass matrix is used for Eqs. (6) and (7).

The convective and pressure (incompressibility) stability is enforced by the fifth term of (4) and the second term of (5) respectively. It may be noticed that the convective stabilizing term is the subtraction between the classical SUPG term [4,25] and its projection onto the finite element space. In this sense, the stabilizing term is the orthogonal projection of the classical SUPG term onto the finite element space, which, with its Galerkin counterpart (second term of (4)), turns out to be a second-order streamline upwind discretization of the original convective operator. This is easily corroborated by writing down the one-dimensional stencil for the nodal point k of such terms (second and fifth terms of the LHS of Eq. (4)) on a mesh of uniform size h using linear elements:

$$\mathscr{L}_k = \tfrac{1}{8}a(u_{k-2} - 8u_{k-1} + 6u_k + u_{k+2}) = a\frac{\mathrm{d}u_k}{\mathrm{d}x} + \frac{h^2}{6}\frac{\mathrm{d}^3 u_k}{\mathrm{d}x^3} + \mathrm{O}(h^3), \tag{9}$$

where a is the advective velocity ($\boldsymbol{u}_h^{n+\theta,i-1}$ for the case of Eq. (4)).

In the same way, the pressure stabilizing term (second term of Eq. (5)) is the orthogonal projection of the pressure gradients onto the finite element space times the intrinsic time τ. If, as was done for the convective terms, the one-dimensional stencil of such a term is written down using a mesh of uniform size h, the following expression is obtained:

$$\mathscr{S}_k = \frac{\tau}{h}\left(\frac{1}{4}p_{k-2} - p_{k-1} + \frac{3}{2}p_k - p_{k+1} + \frac{1}{4}p_{k+2}\right) = \frac{\tau}{h}\frac{3}{12}h^4\frac{\mathrm{d}^4 p_k}{\mathrm{d}x^4} + \mathrm{O}(h^6). \tag{10}$$

Taking into account that τ is proportional to h or h^2 for convective or viscous dominated flows respectively (see (8)), the pressure stabilizing term is at least of fourth-order. Exactly the same pressure stabilizing term has been used by other authors, but following different arguments, to obtain highly accurate stable results (i.e. see [19–21]).

In addition, it must be noted that when convergence is achieved in each timestep (when $\boldsymbol{u}_h^{n+\theta,i-1} \approx \nabla \boldsymbol{u}_h^{n+\theta,i}$ and $p_h^{n+1,i-1} \approx p_h^{n+1,i}$) the first term of (5) has to be zero (or very close to), and the whole system converges to the solution of a monolithic scheme. Hence, the only purpose to keep such a term into the final scheme is for pre-conditioning the whole system (4)–(7), and, in such a way, to enforce the convergence of the block iterative scheme. Extensive numerical experience indicates that if this term is dropped, the system (4)–(7) becomes highly ill-conditioned, and, therefore, convergence is impossible to achieve in most cases. The incorporation of the mentioned term in the final scheme was based on a standard fractional step splitting of the original stabilized monolithic system, as was already presented in detail in Refs. [11,32].

As a final remark, it is important to mention that the stability of the staggered and the monolithic schemes is the same: the staggered scheme is deduced from an unconditionally stable implicit FS formulation (see [11,32] for details). Moreover, our numerical experience indicates that for many problems the uncoupled scheme allows the use of larger timesteps. This is not due to stability issues, but to the fact that the coupled system (velocity–pressure) becomes very stiff for large timesteps (Courant numbers around 10 for some cases). Therefore, the iterative solver used to obtain the solution of the discrete monolithic system of equations (pre-conditioned GMRES) stalls very often. On the other hand, different types of solvers can be used for the different set of equations if they are uncoupled. Such a feature allows to take advantage of the intrinsic character of each system. For example, a pre-conditioned GMRES solver is highly suitable for the unsymmetric hyperbolic momentum equations, and a CG-ILU or CG-Linelet for the symmetric elliptic continuity (pressure) equations.

3. Edge-based implementation

The element-based implementation has been tested in numerous complex applications obtaining accurate results [5,14,31–33]. For most real cases the global matrix of the systems (LHS) and its global force vectors (RHS) has to be re-computed in each iteration of each time step due to the non-linearity of the convective and stabilization terms. If a standard element implementation is used, most of the discrete terms have to be re-computed by looping across the elements, doing the required numerical integrations, doing numerous gather and scatter operations, and doing many redundancies of information in the element data structure [19]. This can be highly demanding in terms of cpu-time: the flops overhead ratio between an element-based implementation and an edge-based one is approximately 2.5 [19]. In addition, a standard shared-memory parallelization of the elemental loop is complicated, since the contributions to the term ij of the different matrices come from more than one element. Then, some type of coloring algorithm must be implemented to enforce that the edge ij is not accessed by elements in different processors at the same time [19].

On the other hand, in an edge-based implementation most of the discrete terms, and all the numerical integrations, are computed at the beginning ot the run, and only a few operations must be implemented to re-compute the different LHS and RHS non-linear terms. Furthermore, the scatter operations are eliminated, and, because of the type of implementation presented in this work, the parallelization is straightforward: two nested loops are performed, the main loop (which is the one to parallelize) is made over the mesh points i, and the inner one is made over the points surrounding the point i (the edges connected to point i). The contributions of the edge ij are computed only when the nodal point i is accessed, and, therefore no coloring algorithm is required. In this work the edge ji (accessed only for the nodal point j) is considered different from the edge ij. However, as will be shown later, the symmetry of the Laplacian-like terms is preserved.

In an edge-based implementation, the idea is to express all the contributions in terms of

$$\int_\Omega N_i N_j \,\mathrm{d}\Omega, \quad \int_\Omega \partial_l N_i \partial_k N_j \,\mathrm{d}\Omega, \quad \int_\Omega N_i \partial_l N_j \,\mathrm{d}\Omega \quad \text{and} \quad \int_\Omega \partial_l N_i N_j \,\mathrm{d}\Omega \tag{11}$$

for $i, j = 1, \ldots, n_{\text{pts}}$ and $l, k = 1, \ldots, n_{\text{dof}}$. Hereafter n_{pts} is the number of nodal points of the computational mesh, n_{dof} the number of degrees of freedom per nodal point (three momentum components for 3D cases), l and k refer to the Cartesian components l and k of a vector or tensor field, i and j to the nodal points i and j, and the integrals (11) will be referred as the Mass ($\boldsymbol{M}$), the Laplacian ($\boldsymbol{L}$), the Gradient ($\boldsymbol{G}$), and the transposed Gradient ($\boldsymbol{G}^{\mathrm{T}}$) matrix, respectively. For fixed domains, all the integrals in (11) can be computed at the beginning of the run, and stored in a standard CSR format. In the above terms, N_i is the standard shape or test function associated to the nodal point i, and $\partial_l N_i = \partial N_i / \partial x_l$. It is important to remark that for interior points (points that are not boundary points of the mesh), $\int_\Omega N_i \partial_l N_j \,\mathrm{d}\Omega = -\int_\Omega \partial_l N_i N_j$. Some authors use such a property to save computer memory [19], However, to simplify the implementation and parallelization of the code, and taking into account that computer memory at present is very cheap, in this work both matrices are stored. Another side effect of doing this, is the reduction of the cache misses, which implies an important reduction of the cpu time.

In the following subsections we show the different hypotheses and assumptions that have to be done to compute the different terms of (4)–(7), using only the integrals presented in (11).

3.1. Viscous term

The classical viscous Galerkin term (third term of (4)), is linear for constant viscosity problems. Hence, its computation using the standard Laplacian matrix (second integral of (11)) is straightforward, and does not involve any approximation. For variable viscosity problems, i.e. turbulent problems, temperature or pressure dependent viscosity, etc., some approximations have to be done. In this work the following two properties, which automatically hold for element-based implementations, are enforced at discrete level: conservation and symmetry.

The conservation property has to be verified at discrete level not only for the viscous term, but also for all the stationary terms of (4)–(7). This can be expressed by writing the system of Eq. (4) as:

$$\frac{1}{\theta \delta t} \boldsymbol{M}(\boldsymbol{U}^{n+\theta} - \boldsymbol{U}^{n}) + \boldsymbol{C}\boldsymbol{U}^{n+\theta} = \boldsymbol{F}, \tag{12}$$

where $\boldsymbol{U}$ is the array of nodal velocities for the whole mesh, $\boldsymbol{C}$ contains all the stationary contributions to the global system matrix (LHS), and $\boldsymbol{F}$ is the force vector (gravity, Boussinesq, etc.). Then, the conservation property can be expressed at discrete level as:

$$\sum_{j=1}^{n_{\text{pts}}} \left(\sum_{k=1}^{n_{\text{dof}}} C_{li,kj} \right) = 0 \tag{13}$$

for $i = 1, \ldots, n_{\text{pts}}$ and $l = 1, \ldots, n_{\text{dof}}$. $C_{li,kj}$ is the row entry li and the column kj of the matrix $\boldsymbol{C}$. Basically (13) implies that the sum of any row li of C has to be zero. This property enforces the right balance of momentum (i.e. no momentum is created by the numerical scheme). A consequence of this is that a constant solution remains constant across time if no source term is present.

The other important property that must be fulfilled by the viscous term, and that also automatically holds for element-based implementations, is the symmetry (viscous term $C^v_{li,kj} = C^v_{kj,li}\ \forall li, kj$). From a physics point of view, it means that the diffusive transport from the point i to the point j is equal to the diffusive transport from the point j to the point i. This is a very important property for Stokes problems (when the convective term can be neglected), because it allows to use cheap symmetric solvers (i.e. conjugate gradient) to deal with the final system of equations. The symmetry is enforced at the discrete level, by computing the viscous contributions as:

$$C^v_{li,lj} \approx \frac{1}{2}(v_i + v_j) \sum_{l=1}^{n_{\text{dof}}} \int_\Omega \partial_l N_i \partial_l N_j \, \mathrm{d}\Omega, \tag{14}$$

where v_i refers to the viscosity evaluated at nodal point i. Note that if the viscosity is not constant, the conservation property (13) could be violated by (14). This shortcoming is avoided by calculating only the non-diagonal terms $C^v_{li,lj}$, and by computing the diagonal one as the subtraction of the same-row non-diagonal terms:

$$C^v_{li,li} = -\sum_{kj \neq li} C^v_{li,kj}, \tag{15}$$

li and $kj = 1, \ldots, n_{\text{dof}} * n_{\text{pts}}$. Then, conservation (13) is enforced at discrete level. Eq. (15) can be re-interpreted as choosing a viscosity for the coefficient $C^v_{li,li}$ that fulfills the conservation property. This is going to be the standard procedure in this work to enforce Eq. (13) for the rest of the stationary terms (Galerkin and stabilization terms).

3.2. Pressure Galerkin term

The pressure term in (4) is computed with the nodal pressures at the last iteration using the Gradient matrix $\boldsymbol{G}$, and is placed at the RHS as:

$$F^p_{li} = -G_{l,ij} p_j = \sum_{j \neq i} \int_\Omega N_i \partial_l N_j \, \mathrm{d}\Omega \, (p_i - p_j). \tag{16}$$

Note that (16) exactly fulfills the conservation property (13). The "diagonal" term is automatically computed as the subtraction of the same-row non-diagonal ones.

3.3. Convective Galerkin term

The (li, lj) contribution of the convective Galerkin term (second term of (4)), is computed in an element-based implementation as:

$$C^c_{li,lj} = \sum_{k=1}^{n_{\text{dof}}} \int_\Omega N_i a_k \partial_k N_j \, \mathrm{d}\Omega, \tag{17}$$

where $\boldsymbol{a}_k$ is the k component of the advective velocity ($\boldsymbol{a} = \boldsymbol{u}_h^{n+\theta,i-1}$ in (4)). At this point some approximation must be done to express (17) using the pre-computed Gradient matrix $\boldsymbol{G}$. This may be done by calculating $C^c_{li,lj}$ as:

$$C^c_{li,lj} \approx \sum_{k=1}^{n_{\mathrm{dof}}} a_{k,ij} \int_\Omega N_i \partial_k N_j \, \mathrm{d}\Omega, \tag{18}$$

where $a_{k,ij}$ is the k component of the advective velocity associated with the edge ij. The first idea is to follow the same procedure used for the viscous term, i.e. taking $\boldsymbol{a}_{ij}$ as the average velocity of nodal points i and j, and enforcing the conservation property by computing the diagonal as the subtraction of the same-row non-diagonal terms. However, this procedure would destroy the second-order Galerkin, or central difference, approximation of the convective term (assuming that linear elements are used). Such a second-order approximation is reflected at discrete level by the fact that $C^c_{li,li} = 0$ for the interior nodal points. Something that naturally arises in a standard finite element approximation using linear elements (or in finite differences using a central scheme).

The only way to fulfill the second-order discretization condition is to take $\boldsymbol{a}_{ij}$ as a function of only the nodal point $i(\boldsymbol{a}_{ij} = \boldsymbol{a}_i)$. In this case it is easy to verify for the interior points that

$$C^c_{li,li} = -\sum_{kj \neq li} a_{k,i} \int_\Omega N_i \partial_k N_j = 0, \tag{19}$$

implying that the approximation is of second-order, and the conservation property (13) holds. Now the question is how to compute $\boldsymbol{a}_i$. The first choice is the velocity at nodal point i, $\boldsymbol{u}_i$. However, in this work the following average has been implemented for the interior points:

$$a_{k,i} = \frac{m_{ij} u_{k,j}}{\sum_{j \neq i} m_{ij} u_{k,j}} \quad \forall j \neq i, \tag{20}$$

where $a_{k,i}$ is the k component of $\boldsymbol{a}_i$, m_{ij} is the term ij of the consistent mass matrix (see (11)), and $u_{k,j}$ is the k component of the velocity at nodal point j. Note that the computation only has to be done over the points j connected to i ($m_{ij} = 0$ if i is not connected to j).

For the boundary points, the same Eq. (20) is used, but including the nodal point i into the weighted average:

$$a_{k,i} = \frac{m_{ij} u_{k,j}}{\sum_{j=1}^{n_{\mathrm{pts}}} m_{ij} u_{k,j}} \quad \forall j. \tag{21}$$

Numerical experience indicates that Eqs. (20) and (21) for the advective velocity associated with the nodal point i produce a better convergence rate and more accurate results than taking only $\boldsymbol{a}_i = \boldsymbol{u}_i$. Moreover, it can be checked that given a discrete velocity field $\boldsymbol{u}$, the convective RHS obtained using a standard element-based implementation

$$f^{\mathrm{element}}_{li} = \sum_{j=1}^{n_{\mathrm{pts}}} \left(\sum_{k=1}^{n_{\mathrm{dof}}} \int_\Omega N_i u_k \partial_k N_j \, \mathrm{d}\Omega \, u_{lj} \right), \tag{22}$$

is much better approximated by using Eqs. (20) and (21) for the advective velocity than $\boldsymbol{a}_i = \boldsymbol{u}_i$. In addition, it is important to remark that (20) and (21) naturally arise from a central finite difference (or central finite volume) discretization of the convective NS term.

3.4. Convective stabilization terms

The high order SUPG term of the momentum equation (last LHS term of (4)) can be expressed in two parts: The low order one given by

$$C_{li,lj}^{\text{low}} = \int_{\Omega} \tau \sum_{k=1}^{n_{\text{dof}}} (a_k \partial_k N_i) \sum_{m=1}^{n_{\text{dof}}} (a_m \partial_m N_j)\, \mathrm{d}\Omega \tag{23}$$

and the projection, or high order term, which is placed at the RHS as:

$$F_{li}^{\text{high}} = \sum_{j=1}^{n_{\text{pts}}} \left(\int_{\Omega} \tau \sum_{k=1}^{n_{\text{dof}}} (a_k \partial_k N_i) N_j \, \mathrm{d}\Omega \, \pi_{lj} \right). \tag{24}$$

In order to express (23) and (24) in terms of the integrals (11), two approximations have to be realized: The advective velocity $\boldsymbol{a}$ and the intrinsic time τ must be taken out of the integrals.

For the advective velocity the natural choice is given by (20) and (21). This is not only consistent with its Galerkin convective counterpart, but also approximates in a closer way the element-based terms. In the same way, the intrinsic time τ is evaluated at nodal point i as:

$$\tau_i = \frac{h_i^2}{4\nu_i + 2|\boldsymbol{a}_i| h_i}, \tag{25}$$

where h_i is the minimum length of all the edges surrounding the nodal point i. The dependence of τ only on the nodal point i is required for consistency reasons: if τ depends also on node j, exact nodal values of the velocity would not satisfy the discrete equations [10]. The reason is that the sum over j would not produce the convective term if the partial derivatives of N_j are multiplied by a factor that depends on node j. This can be clarified by writing the edge-based terms times the velocity vector for the equation li:

$$C_{li,lj}^{\text{low}}\, u_{lj} \approx \sum_{k,m=1}^{n_{\text{dof}}} \tau_i a_{k,i} a_{m,i} \int_{\Omega} \partial_k N_i \partial_m N_j \, \mathrm{d}\Omega \, u_{lj}. \tag{26}$$

Note that if $\boldsymbol{a}$ or τ depend on j, the convective term $a_m \partial_m N_j u_{lj}$ would not be recovered: Each contribution would be multiplied by a different factor. The low order SUPG terms are computed as:

$$C_{li,lj}^{\text{low}} \approx \sum_{k,m=1}^{n_{\text{dof}}} \tau_i a_{k,i} a_{m,i} \int_{\Omega} \partial_k N_i \partial_m N_j \, \mathrm{d}\Omega \tag{27}$$

for the non-diagonal terms $i \neq j$, and, following the same procedure as for the viscous Galerkin term, the diagonal is calculated by enforcing the conservation at discrete level:

$$C_{li,li}^{\text{low}} = -\sum_{kj \neq li} C_{li,kj}^{\text{low}}. \tag{28}$$

The high order terms (24) are also calculated by enforcing the conservation property at the discrete level using τ_i and $\boldsymbol{a}_i$, and the transposed Gradient matrix $\boldsymbol{G}^{\mathrm{T}}$:

$$F_{li}^{\text{high}} = -\tau_i \sum_{j \neq i} \left(\sum_{k=1}^{n_{\text{dof}}} \int_{\Omega} \partial_k N_i N_j \, \mathrm{d}\Omega \, (\pi_{lj} - \pi_{li}) \right). \tag{29}$$

Finally, the projection of the convective term $\boldsymbol{\pi}$ is computed using an approximation to (6), and, again, enforcing the conservation property:

$$m_{ii}^{l} \pi_{li} = \sum_{j \neq i} \left(\sum_{k=1}^{n_{\text{dof}}} a_{k,i} \int_{\Omega} N_i \partial_k N_j \, \mathrm{d}\Omega \, (u_{lj} - u_{li}) \right), \tag{30}$$

where m_{ii}^{l} is the component ii of the standard lumped mass matrix.

3.5. Incompressibility equation

The discrete terms of (5) are approximated following similar procedures than those presented above. The LHS Laplacian terms (part of the first and of the third term of Eq. (5)) are approximated as follows:

$$L_{i,j} = \sum_{l=1}^{n_{\text{dof}}} \int_{\Omega} (\delta t + \tau) \partial_l N_i \partial_l N_j \, \mathrm{d}\Omega \approx (\delta t + \tau_{ij}) \sum_{l=1}^{n_{\text{dof}}} \int_{\Omega} \partial_l N_i \partial_l N_j \, \mathrm{d}\Omega \tag{31}$$

$\forall j \neq i$, and the diagonal term is computed, again, by enforcing the conservation property (13). This is:

$$L_{i,i} = -\sum_{j \neq i} L_{i,j}. \tag{32}$$

The intrinsic time τ_{ij} is taken as:

$$\tau_{ij} = \frac{h_i + h_j}{4(v_i + v_j) + 2(|\boldsymbol{a}_i| + |\boldsymbol{a}_j|)}, \tag{33}$$

where h_i is the minimum length of the edges surrounding the nodal point i. Note that even though the term $\boldsymbol{L}$ is conservative (the conservation is enforced at the discrete level), it is not consistent in the sense expressed above for the low order stabilizing convective term. This is, τ depends on nodal point j, and, therefore, exact nodal values of the pressure would not satisfy the discrete equation. The consistency could be held by making τ only depending on the nodal point i. However, this procedure would destroy the symmetry of the system (5). In addition, it has been demonstrated that the pressure stabilizing terms are formally of fourth-order (see Section 2), which means that their effect in the final accuracy is very small. The choice was made: To preserve symmetry and conservation at the price of losing some consistency.

The approximation of the high order stabilizing term (the other part of the second term of (5)), is carried out by imposing the global conservation at the discrete level as:

$$F_i^p = \sum_{j \neq i} \left(\tau_{ij} \sum_{l=1}^{n_{\text{dof}}} \int_{\Omega} \partial_l N_i N_j \, \mathrm{d}\Omega \, (\xi_{lj} - \xi_{li}) \right) \tag{34}$$

and the projection of the pressure gradients onto the finite element space computed using (7) as:

$$m_{ii}^l \xi_{li} = \sum_{j \neq i} \int_{\Omega} N_i \partial_l N_j \, \mathrm{d}\Omega \, (p_j - p_i), \tag{35}$$

where the global conservation is imposed, and a lumped approximation to $\boldsymbol{M}$ is again used.

The computation of the last LHS term $-\delta t(\nabla p_h^{n+1,i-1}, \nabla q_h)$ is straightforward and no approximations have to be done. This is expressed as $-\delta t \sum_{l=1}^{n_{\text{dof}}} \int_{\Omega} \partial_l N_i \partial_l N_j \, \mathrm{d}\Omega \, (p_j - p_i)$ (the sub/superscripts has been omitted for simplicity).

Finally, the divergence term (last term of (5)) is computed as:

$$F_i^{\text{div}} = -\sum_{j \neq i} \left(\sum_{l=1}^{n_{\text{dof}}} \int_{\Omega} N_i \partial_l N_j \, \mathrm{d}\Omega \, (u_{lj} - u_{li}) \right). \tag{36}$$

Again, the conservation has been enforced at the discrete level.

Remark 1. Note that for all the stationary terms, only the off-diagonal contributions have to be computed. The main loop to compute the terms goes over the nodal points, and then over the points surrounding the nodal point (taking advantage of the CSR storage). The LHS diagonal entrances are computed later in a

second loop by using the conservation property (13). In a standard edge-based implementation [19], the conservation property is reflected by the fact that whatever is added to a point of the edge, is subtracted from the other.

Remark 2. The temporal terms in the momentum equation are assembled after $\boldsymbol{C}$ is constructed (see (12)). The consistent Mass matrix or its Lumped approximation can be used. In general, the first one is utilized by the authors to obtain time accurate results, and the second one for steady-state solutions.

4. Monotonicity preserving term

The high order formulation presented in (4)–(7) could not avoid localized oscillations (overshoots and undershoots) which deteriorate the convergence of the formulation. Even though most numerical problems that have been solved to date have not presented such anomalies, a type of non-linear numerical switch had to be designed for some of them to avoid local spurious oscillations.

The idea is to identify the flow regions with strong pressure gradients by the difference between the pressure term and its projection. Then, at each iteration, a variable β is computed at the edge ij as:

$$\beta = 1 - \frac{|p_i - p_j| + 0.5 l_{k,ij}(\xi_{k,i} + \xi_{k,j})}{|p_i - p_j| + |0.5 l_{k,ij}(\xi_{k,i} + \xi_{k,j})|}, \tag{37}$$

where p_i is the pressure at nodal point i, $l_{k,ij} = x_{k,i} - x_{k,j}$ is the dimension of the edge ij in the k direction, and $\xi_{k,i}$ is the k component of the projected pressure gradient at nodal point i. It is straightforward to see that $0 \leqslant \beta \leqslant 1$, taking values close to 1 where the pressure field is smooth, and close to 0 in flow regions with sharp pressure gradients. A numerical switch to decrease the discretization order in such sharp gradient regions, is obtained by multiplying the high order SUPG and pressure stabilizing terms by β (see (4) and (5)). It is important to remark, that a very similar pressure sensor is utilized by some authors, to avoid expensive limiting procedures when Riemann solvers are used to deal with convection dominated flows (see [19] for details).

5. Numerical examples

5.1. Backward-facing step

The first example considered is the laminar backward-facing step at different *Re* numbers. It was performed to verify the effectiveness of the edge-based formulation (4)–(7) to deal with NS steady-state problems at low and high *Re*. The problem geometry and mesh is shown in Fig. 1. The aspect ratio of the backward-facing step H to the overall sectional width is 1:2 and the total length in the horizontal direction is 40 H. A fully developed parabolic velocity profile is prescribed at the inflow boundary. At the outflow, the pressure and the viscous stresses were set to zero. The mesh is composed of 16,472 linear triangles and 8542 nodal points.

According to Armaly et al. [1], the *Re* number will be based on the average value of the inlet velocity profile and the cross-sectional width of the whole domain. For $Re < 500$, only one re-circulation zone exists behind the step. For higher values of *Re*, another re-circulation zone appears at the top wall of the channel. Experimental results indicate that a third re-circulation zone appears at the bottom of the wall for values of $Re > 1000$.

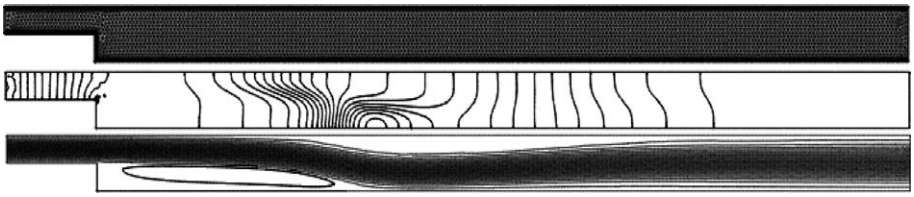

Fig. 1. From top to bottom: Mesh of 16,472 linear elements and 8542 nodal points. Streamline pattern. Pressure contours; peaks (-1.012×10^{-1}, 3.067×10^{-4}).

The numerical results using the edge-based implementation agree very well for $Re < 600$ with those that can be found in the above mentioned reference. For brevity, they have not been included here, and only those for $Re = 1000$ are included. In Fig. 1 the streamline pattern is presented. The length of the vortex behind the step is approximately 13.0. The experimental result is 14.3. Taking into account that some discrepancies should be expected due to the three-dimensionality of the experimental flow at this Re number, the results are good. In Fig. 1 the pressure field is presented. Note that there are no pressure reflections at the outflow: The pressure gradient is parallel to the horizontal direction. Finally, it is important to mention that the same results were obtained with the element-based code, but using almost three times more cpu-time than with the edge-based code.

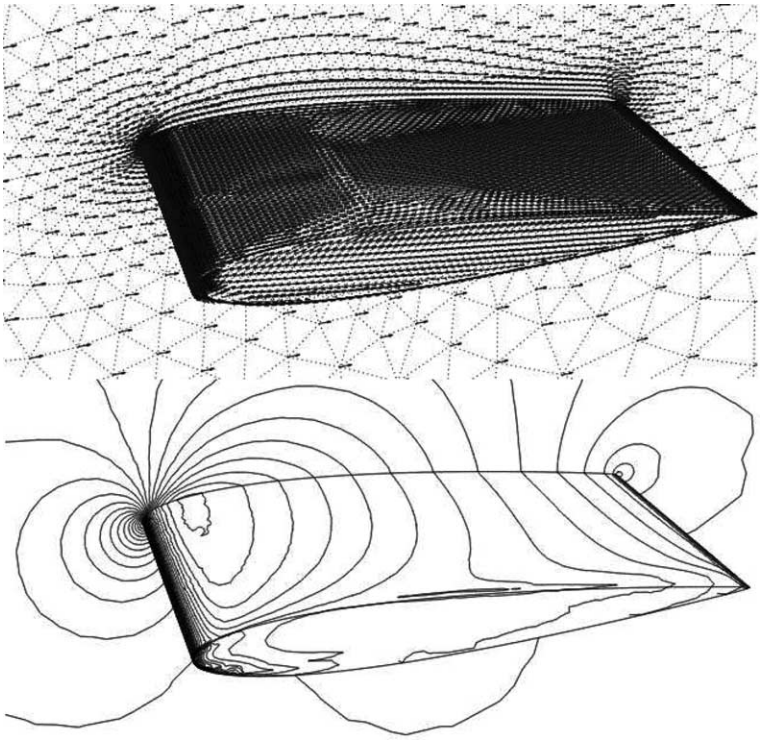

Fig. 2. From top to bottom: Mesh of 369,713 linear elements 68,491 nodal points and velocity field. Pressure contours; peaks (-4.205×10^{-1}, 3.804×10^{-1}).

5.2. NACA0012 wing

This example was performed to see if the edge-based formulation could deal with incompressible Euler flows in 3D geometries. The volumetric mesh consisted of 369,713 elements and 68,491 nodes. The angle of attack of the flow was fixed at a value of five degrees, the velocity field was prescribed at the inflow, and the pressures were set to zero at the outflow boundaries.

In Fig. 2 some results using the edge-based formulation are presented. It can be observed that they are free of spurious oscillations. The cpu-time to decrease the $L2$ residual norm five-orders of magnitude on an Pentium Intel386 processor at 1.0 GHz was about 6000 s. The cpu-time ratio between the element-based code and the edge was around 3. The main difference was found in the computation of the global LHS and RHS coefficients. The element-based scheme spent almost seven times more cpu-time in such operations than the edge-based one.

5.3. Air flow through the respiratory ways

This transient real 3D case consisted of computing the air flow through a portion of the respiratory ways (mesh of 992,354 linear tetrahedra and 184,395 points). The timestep size used for the implicit scheme was of 0.0625 s, which corresponds to 64 timesteps per respiratory cycle. This problem was also computed with an explicit second-order accurate fractional step (FS) edge-based code. The timestep size that had to be used for such a solver was at least three-orders of magnitude smaller than the timestep used for the implicit method. Therefore, the implicit scheme was about 10 times faster than the explicit one, and the numerical results were practically the same. In addition, the important flow features were captured in an accurate manner, as can be observed in Fig. 3.

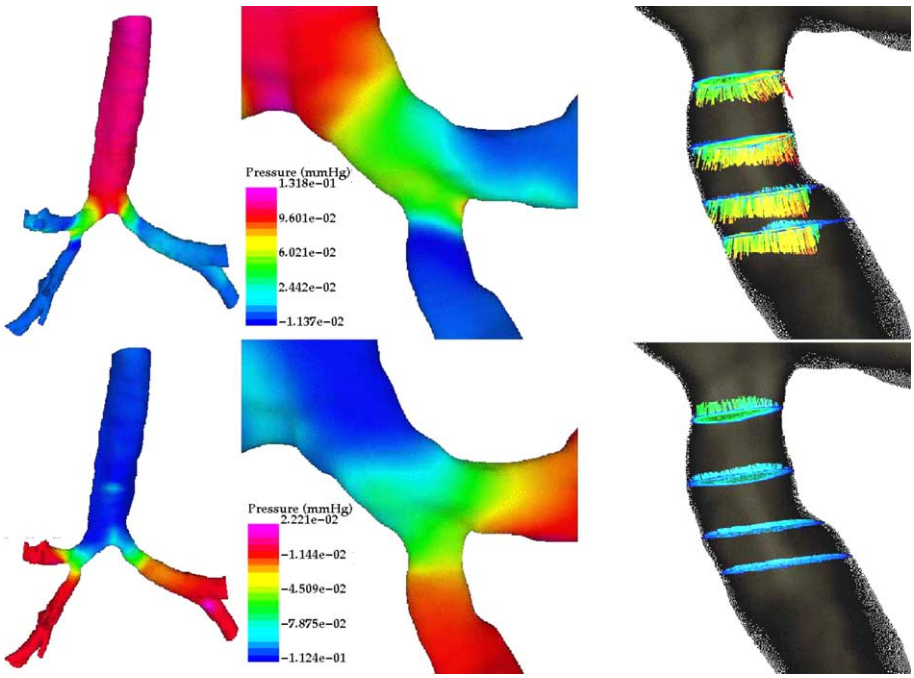

Fig. 3. Global pressure drops, pressure and velocity details in the stenosis region. Note the reversal in the pressure drop and velocities from inspiration to expiration. From top to bottom: $t = 1.0$ s and $t = 3.0$ s.

6. Ship topside flow study

The purpose of this example is to compare the numerical results of the edge-based formulation with experimental data on a real 3D problem. The geometry of the ship (LPD17) is illustrated in Fig. 4 (top left). The dimensions are: 200 m in length, 30 m in width, and 50 m in height above the waterline. The computational mesh consisted of 5,640,000 tetrahedra and 990,000 points.

The inflow condition was set to uniform flow with a velocity of 30 knots (15.43 m/s) assuming maximum ship speed. The inflow was in the x direction. The timestep was taken to 0.01 s, the density to 1.225 kg/m^3, and the viscosity to 1.789×10^{-5} kg/m/s. The Reynolds number based on the height of the ship is 5×10^7. The flow is in the turbulent regime for this Reynolds number. Hence a LES simulation was performed with the Smagorinsky turbulence model described in [30]. The logarithmic wall law was used for the boundary condition, and the pressure was set to the hydrostatic pressure at the outflow.

In the first part of the run, a pseudo steady flow was established. After this state was reached, the flow was integrated for 90 s of real time. Buoyancy effects were considered in the simulation using the Boussinesq's approximation.

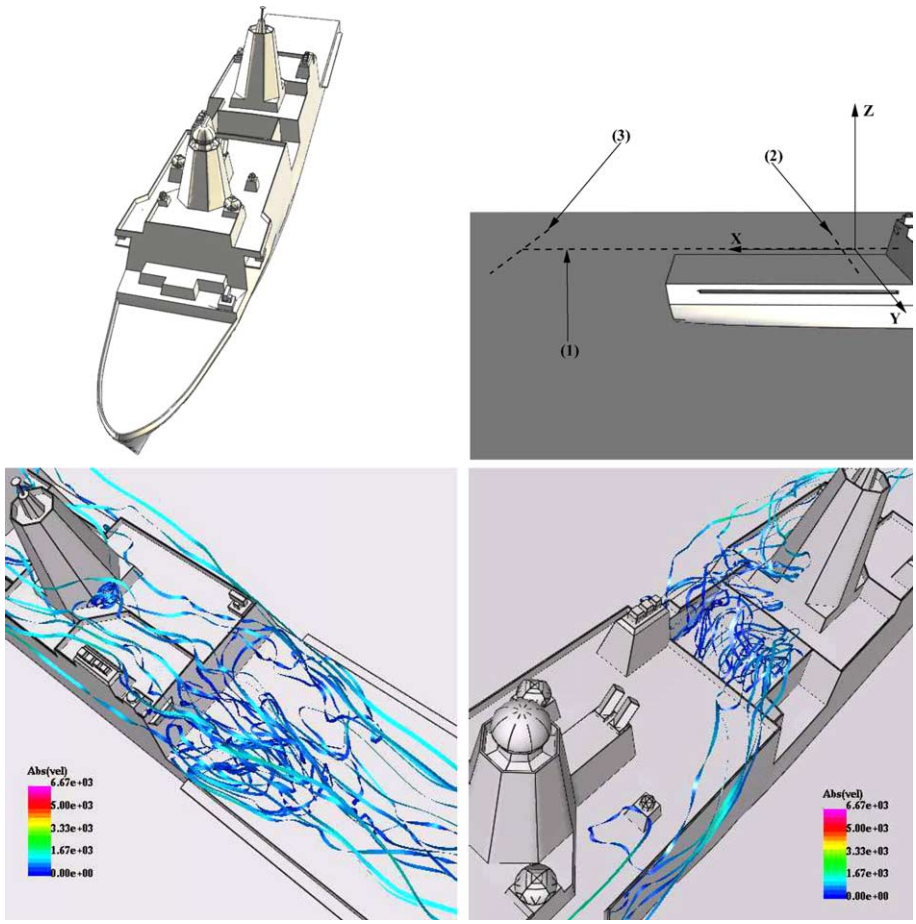

Fig. 4. From top to bottom and left to right: LPD17 geometry. Position of station data time histories: side view of the flight deck; experimental measurements were obtained at lines (1), (2), and (3). Ribbons colored with the velocity absolute value.

Some instantaneous streamlines of the flow (ribbons) are shown in Fig. 4. This technique helps to capture the areas with re-circulation. Fig. 4 (bottom left) shows a large re-circulation above the landing deck. Fig. 4 (bottom right) shows a particularly interesting feature: The ribbons indicate a pronounced crossflow from port to starboard near the middle of the ship.

Experimental velocity data from wind tunnel LDV measurements was available at the positions shown in Fig. 4 (top right). The experimental velocities were averaged and compared with averaged numerical data over a period of 90 s. Figs. 5–7 show the comparisons. The plotted velocities are in the x direction (u), in the y

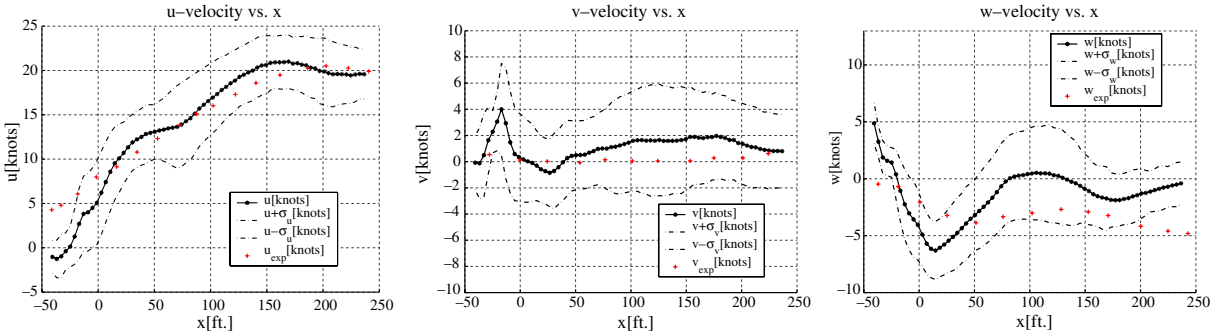

Fig. 5. From left to right: LPD17 u average velocity (1), v average velocity (2), w average velocity (3).

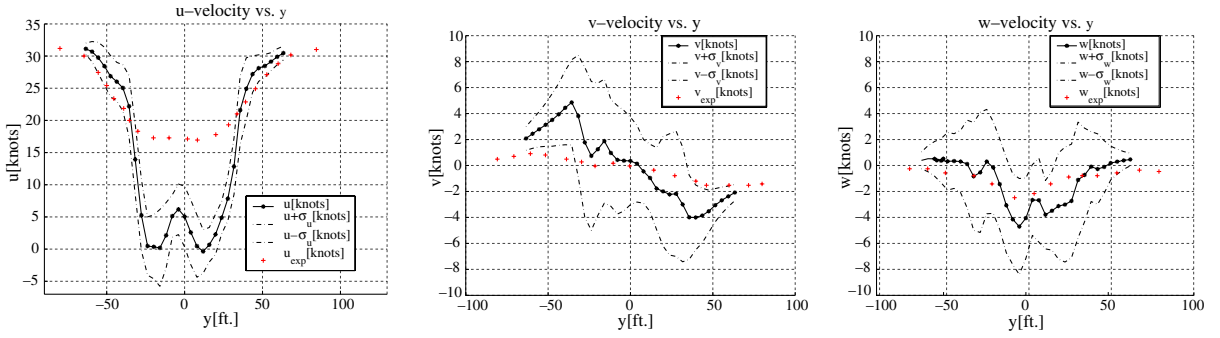

Fig. 6. From left to right: LPD17 u average velocity (1), v average velocity (2), w average velocity (3).

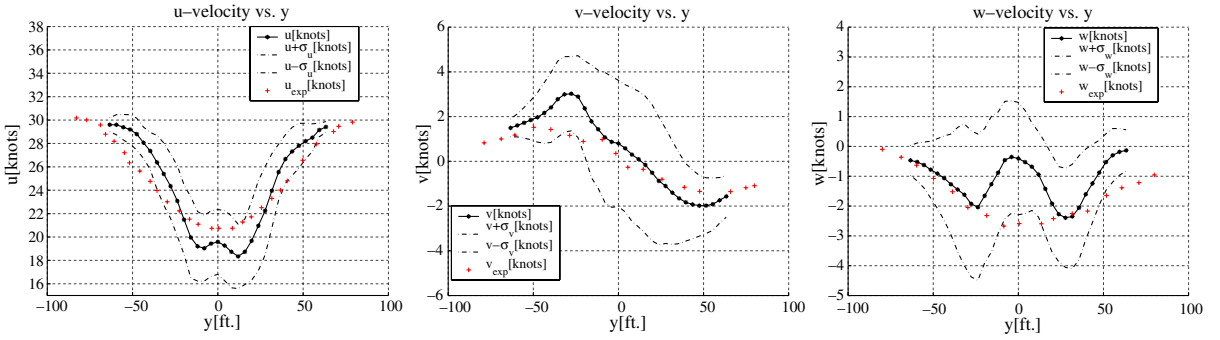

Fig. 7. From left to right: LPD17 u average velocity (1), v average velocity (2), w average velocity (3).

direction (v), and in the z direction (w). The crosses are the experimental data, the stars are the average of the numerical results, and the dash-dotted lines are the standard deviation of the numerical results. The numerical results agree fairly well with the experimental data for most of the plots. The numerical velocity in the x direction in the line station (2) (see Fig. 6) shows a significant deviation from the experimental data at the center positions. One possible explanation is the lack of resolution of the experimental instruments. This was also seen in earlier computations [26,27]. Experimental errors were not provided for the LDV data.

7. Conclusions

An implicit second-order accurate monolithic edge-based scheme was presented to solve incompressible flow problems. It was developed from an element-based formulation, which has already been extensively tested. The main difference between the element-based and the edge-based formulation is the speed to construct the terms of the final LHS and RHS system. The edge-based solver does less operations than the element one, which implies an important reduction in the total CPU time.

Other ingredients of the formulation are: The incompressible and convective terms are stabilized using an OSS scheme. For the incompressibility equation, it was demonstrated that such methods reduce to the same fourth-order pressure term added by some authors to obtain high order accurate results. For the convective term, it was shown that the OSS stabilization is nothing but a second-order approximation of the respective convective operator. The numerical experience indicates that the formulation is very efficient for all Reynolds numbers, and that its time accuracy is excellent. Such scheme seems to be very efficient for transient cases, when the critical timestep of the problem is orders of magnitude smaller than the timestep required to obtain time-accurate results (physical timestep).

References

[1] B. Armaly, F. Durst, J. Pereira, B. Schonung, Experimental and theoritical investigation of backward-facing step flow, J. Fluid Mech. 127 (1983) 473–476.
[2] M. Behr, T. Tezduyar, Finite element solution strategies for large-scale flow simulations, Comput. Methods Appl. Mech. Engrg. 112 (1994) 3–24.
[3] J. Bell, P. Colella, H. Glaz, A second order projection method for the Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[4] A. Brooks, T. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equation, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.
[5] J. Cebral, P. Yim, R. Löhner, O. Soto, H. Marcos, P. Choyke, New methods for computational fluid dynamics of carotid artery from magnetic resonance angiography, in: Proceedings of SPIE Medical Imaging, vol. 4321, paper No. 22, San Diego, California, February, 2001.
[6] Y.H. Choi, C.L. Merkle, The application of preconditioning in viscous flows, J. Comput. Phys. 105 (1993) 207–233.
[7] A. Chorin, A numerical method for solving incompressible viscous problems, J. Comput. Phys. 2 (1967) 12–26.
[8] A. Chorin, On the convergence of discrete approximation to the Navier–Stokes equations, Math. Comput. 23 (1969).
[9] R. Codina, A stabilized finite element method for generalized stationary incompressible flows, Comput. Methods Appl. Mech. Engrg. 190 (2001) 2681–2706.
[10] R. Codina, A nodal-based implementation of a stabilized finite element method for incompressible flow problems, Int. J. Numer. Methods Fluids 33 (2000) 737–766.
[11] R. Codina, Pressure stability in fractional step finite element methods for incompressible flows, J. Comput. Phys. 170 (2001) 112–140.
[12] R. Codina, Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods, Comput. Methods Appl. Mech. Engrg. 190 (2000) 1579–1599.
[13] R. Codina, O. Soto, Finite element solution of the Stokes problem with dominating Coriolis force, Comput. Meth. Appl. Mech. Engrg. 142 (1997) 215–234.
[14] R. Codina, O. Soto, A numerical model to track two-fluid interfaces based on a stabilized finite element method and the level set technique, Int. J. Numer. Methods Fluids 4 (2002) 293–301.

[15] G. Cowles, L. Martinelly, Fully non-linear hydrodynamic calculations for ship design on parallel computing platforms, in: Proceedings of 21st Symposium on Naval Hydrodynamics, Trondheim, Norway, 1996.
[16] L. Franca, S. Frey, Stabilized finite element methods: II. The incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 99 (1992) 209–233.
[17] T. Hino, A unstructured grid method for incompressible viscous flows with a free surface, AIAA-97-0862, 1997.
[18] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[19] R. Löhner, Applied CFD Techniques, John Wiley & Sons, 2001.
[20] R. Löhner, C. Yang, E. Oñate, S. Idelsohn, An unstructured grid-based, parallel free surface solver, AIAA-97-1830, 1997.
[21] R. Löhner, C. Yang, E. Oñate, S. Idelsohn, An unstructured grid-based, parallel free surface solver, Appl. Numer. Math. 31 (1999) 271–293.
[22] D. Martin, R. Löhner, An implicit Linelet-based solver for incompressible flows, AIAA-92-0668, 1992.
[23] L. Martinelli, J.R. Farmer, Sailing through the nineties: computational fluid dynamics for ship performance analysis and design, in: D.A. Caughey, M.M. Hafez (Eds.), Frontiers of Computational Fluid Dynamics, J. Wiley, 1994 (Chapter 27).
[24] J. Peraire, K. Morgan, J. Peiro, The simulation of 3D incompressible flows using unstructured grids, in: D.A. Caughey, M.M. Hafez (Eds.), Frontiers of Computational Fluid Dynamics, J. Wiley, 1994 (Chapter 16).
[25] R. Ramamurti, R. Löhner, A parallel implicit incompressible flow solver using unstructured meshes, Comput. Fluids 5 (1996) 119–132.
[26] R. Ramamurti, W.C. Sandberg, Lpd-17 topside aerodynamic study: FEFLO, NRL Memorandum Report NRL/MR/6410-00-8498, Center for Reactive Flow and Dynamical Systems, Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory, October 2000.
[27] R. Ramamurti, W.C. Sandberg, Unstructured grids for unsteady ship airwakes: a successful validation, Naval Eng. J. 114 (2002), Fall Technical Paper.
[28] A. Rizzi, L. Eriksson, Computation of inviscid incompressible flow with rotation, J. Fluid Mech. 153 (1985) 275–312.
[29] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishers, Boston, 1996.
[30] J. Smagorinsky, General circulation experiments with the primitive equations. I. The basic experiment, Monthly Weather Rev. 91 (1963) 99–164.
[31] O. Soto, R. Löhner, CFD shape optimization using an incomplete-gradient adjoint formulation, Int. J. Numer. Methods Engrg. 51 (2001) 735–753.
[32] O. Soto and R. Löhner, An implicit monolithic time accurate finite element scheme for incompressible flow problems, AIAA-2001-2616, 2001.
[33] O. Soto, R. Löhner, F. Camelli, A Linelet preconditioner for incompressible flow solvers, Int. J. Numer. Methods Heat Fluid Flow 13 (2003) 133–147.
[34] T. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations, J. Comput. Phys. 72 (1987) 277–298.
[35] J.M. Weiss, W.A. Smith, Preconditioning applied to variable and constant density time-accurate flows on unstructured grids, AIAA J. 33 (1995).