INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2001; **36**: 659–685 (DOI: 10.1002/fld.151)

Edge-based finite element method for shallow water equations

F. L. B. Ribeiro[a,*,1], A. C. Galeão[b,2] and L. Landau[a,3]

[a] *Programa de Engenharia Civil, COPPE/Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil*
[b] *Laboratório Nacional de Computação Científica, Av. Getúlio Vargas 333, Petrópolis, Brazil*

SUMMARY

This paper describes an edge-based implementation of the generalized residual minimum (GMRES) solver for the fully coupled solution of non-linear systems arising from finite element discretization of shallow water equations (SWEs). The gain in terms of memory, floating point operations and indirect addressing is quantified for semi-discrete and space–time analyses. Stabilized formulations, including Petrov–Galerkin models and discontinuity-capturing operators, are also discussed for both types of discretization. Results illustrating the quality of the stabilized solutions and the advantages of using the edge-based approach are presented at the end of the paper. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: Petrov–Galerkin; shallow water equations; stabilization methods

1. INTRODUCTION

Shallow water equations (SWEs) play an important role in hydraulic engineering. Many problems, such as the prediction of dam breaks, currents and water elevation in coastal estuaries, bays, oceans and rivers can be modeled by the SWEs. The solution of these equations also provides the necessary data for the determination of pollutant transport in water quality models.

For many years finite difference methods have been extensively used to solve problems governed by the SWEs. Only in the late 1970s did the finite element method appear as a new alternative in the context of fluid dynamics. With the increase in computational power in the past 25 years, much attention has been given to the finite element method. Although computationally more complex, the finite element method was recognized as having some

* Correspondence to: Programa de Engenharia Civil, COPPE/Universidade Federal do Rio de Janeiro, Caixa Postal 68506, Rio de Janeiro, RJ, 21945-970, Brazil.
[1] E-mail: fernando@coc.ufrj.br
[2] E-mail: acng@lncc.br
[3] E-mail: landau@lamce.ufrj.br

Received July 1999
Revised February 2000

evident advantages over finite difference schemes. The inherent capacity of treating the boundaries in a more natural way (capacity of dealing with unstructured meshes) and its solid mathematical basis have encouraged the use of the finite element method.

As is well known, the vertically averaged version (2DH model) of the SWEs may constitute an incompletely parabolic or a hyperbolic system. This system, if written in terms of conservation variables, leads to non-symmetric forms. Symmetric forms can be obtained by imposing a change of variables. One example of symmetric form is the one obtained with the use of the so-called entropy variables [1–3]. In this case, it can be shown that for incompletely parabolic systems, the Galerkin method satisfies the entropy condition [4,5]. Nevertheless, this is not the case with hyperbolic systems, where stability bounds can only be achieved by adding quadratic weighted residuals to the variational formulation. These positive terms are provided by Petrov–Galerkin models and discontinuity-capturing operators. In other words, stabilized methods should be used if shocks/discontinuities and/or sharp layers need to be correctly represented.

The finite element discretization of the SWEs leads to a non-linear coupled non-symmetric system of algebraic equations. Transient solutions may be obtained by using predictor multi-corrector time stepping algorithms, which means that a sequence of linear systems must be solved. Rather than direct solvers, iterative methods are frequently used to solve these linear systems. In this paper we discuss the implementation of the generalized residual minimum (GMRES) iterative solver using an edge-based data structure. In the context of finite elements, edge-based representation has already been used in order to efficiently compute the nodal balance of fluxes [6,7]. However, other benefits can be attained from edge-based data structures. In comparison with the conventional element-by-element techniques, edge-by-edge formulation results in a remarkable improvement, in terms of memory and CPU requirements, when used to perform matrix–vector product operations [8]. We address this issue by applying an edge-based data structure to obtain the fully coupled solution of the SWEs.

This paper is organized as follows: in Section 1 we recall the governing equations, written in terms of velocity/celerity variables [9–11]. In Sections 2 and 3, the space–time and semi-discrete stabilized formulations are discussed. In Section 4 the predictor multi-corrector algorithms for both types of discretization are described, and finally, in Section 5 we present the edge-based solution strategy, quantifying the gain in terms of memory requirements, CPU time and indirect addressing for the linear triangle and its correspondent space–time wedge-shaped element. Three numerical examples are shown at the end of the paper. The first example is the classical one-dimensional dam-break problem, where the stabilized solutions are compared with the exact solution obtained with a Riemann solver. The other two examples were selected in order to demonstrate the superior performance of the edge-based approach when compared with the usual element-by-element techniques. The second example is a two-dimensional extension of the first and the third one is the simulation of a tidal flow.

2. GOVERNING EQUATIONS

In the free surface flow of relatively thin layers (here represented by H, the total water depth of water, measured along the vertical direction) the horizontal velocities are of primary importance and the problem can be reasonable approximated in two dimensions.

Let $(x_1, x_2) \in \Omega \subset \Re^2$ define a set of points on the horizontal plane $x_3 = 0$, representing the undisturbed water surface. For a given time t, let η represent the water surface elevation (perturbed motion) and h be the depth of water, both measured from the undisturbed water surface. The domain of interest is the shallow layer of width $H = \eta + h$ confined between the free water surface, described by the function $x_3 = \eta(x_1, x_2, t)$, and the bottom bed surface, given by the function $x_3 = -h(x, y)$. The perturbed shallow water body motion is governed by the three-dimensional Navier–Stokes equations for incompressible flow, and the basic assumption to derive the SWEs relies on the use of a barotropic model to physically approximate the shallow water motion. This is achieved assuming that the pressure field is near to hydrostatic equilibrium (undisturbed configuration). Integrating the three-dimensional Navier–Stokes equations along the vertical direction, under the simplifying assumption of hydrostatic pressure distribution (negligible vertical acceleration and viscous forces), combined with the free surface kinematic condition and the non-impenetrability bottom bed condition, we arrive at the 2DH model for the SWEs

$$\mathbf{U}_{,t} + \mathscr{F}_{i,i} = \mathbf{F} \tag{1}$$

where the inferior comma represents partial differentiation, repeated indices indicate summation, $\mathbf{U}^T = [Hu_1 \quad Hu_2 \quad H]$ denotes the vector of conservation variables, $\mathscr{F}_i$ $(i = 1, 2)$ are the hydraulic fluxes

$$\mathscr{F}_1^T = \left[Hu_1^2 + \frac{1}{2} gH^2 \quad Hu_1u_2 \quad Hu_1 \right] \tag{2}$$

$$\mathscr{F}_2^T = \left[Hu_1u_2 \quad Hu_2^2 + \frac{1}{2} gH^2 \quad Hu_2 \right] \tag{3}$$

and $\mathbf{F}$ is the source term

$$\mathbf{F} = \begin{bmatrix} gH\dfrac{\partial h}{\partial x_1} + fHu_2 - \gamma u_1 + \dfrac{\tau_1^s}{\rho} + \dfrac{\partial}{\partial x_i}\left(vH\dfrac{\partial u_1}{\partial x_i}\right) \\ gH\dfrac{\partial h}{\partial x_2} - fHu_1 - \gamma u_2 + \dfrac{\tau_2^s}{\rho} + \dfrac{\partial}{\partial x_i}\left(vH\dfrac{\partial u_2}{\partial x_i}\right) \\ 0 \end{bmatrix} \tag{4}$$

In the above expressions, (u_1, u_2) are the vertically averaged horizontal velocity components, g is the gravity acceleration, f is the Coriolis parameter, ρ is the constant density (well mixed layers), τ_i^s are the wind stress components at the free surface and v is the eddy viscosity coefficient, which takes into account the horizontal diffusion effects for an 'appropriate' turbulence model. Bottom friction forces are modeled by the Chezy formula $\gamma = g(u_1^2 + u_2^2)^{1/2}/C^2$, where C is the Chezy coefficient.

System (1) is written in the so-called divergence or conservative form and represents an incomplete parabolic system of equations. On the other hand, the reduced equation

$$\mathbf{U}_{,t} + \mathscr{F}_{i,i} = \mathbf{0} \tag{5}$$

represents a hyperbolic system.

Alternatively to the divergence form, the SWEs can be written in advective form if we apply the chain rule. From the hydraulic fluxes definition we have that

$$\mathscr{F}_{i,i}(\mathbf{U}) = \frac{\partial \mathscr{F}_i}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x_i} = \mathbf{A}_i \mathbf{U}_{,i} \tag{6}$$

where $\mathbf{A}_i$ are the Jacobian flux matrices

$$\mathbf{A}_1 = \begin{bmatrix} 2u_1 & 0 & gH - u_1^2 \\ u_2 & u_1 & -u_1 u_2 \\ 1 & 0 & 0 \end{bmatrix}, \qquad \mathbf{A}_2 = \begin{bmatrix} u_2 & u_1 & -u_1 u_2 \\ 0 & 2u_2 & gH - u_2^2 \\ 0 & 1 & 0 \end{bmatrix} \tag{7}$$

Using these definitions, Equation (1) can be rewritten as

$$\mathbf{U}_{,t} + \mathbf{A}_i \mathbf{U}_{,i} = \mathbf{F} \tag{8}$$

or in equivalent way

$$\mathbf{U}_{,t} + \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{F} \tag{9}$$

with

$$\mathbf{A}^T = [\mathbf{A}_1 \quad \mathbf{A}_2], \qquad \nabla \mathbf{U} = \begin{bmatrix} \mathbf{U}_{,1} \\ \mathbf{U}_{,2} \end{bmatrix}, \qquad \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{A}^T \nabla \mathbf{U} \tag{10}$$

The dot product notation $\mathbf{A} \cdot \nabla \mathbf{U}$ is used to represent a matrix product in order to keep an analogy with the scalar transport equation, as this term plays the role of a generalized advective term. Since the matrices $\mathbf{A}_i$ are non-symmetric, Equation (9) represents a non-symmetric incomplete parabolic system, written in terms of the variables (Hu_1, Hu_2, H).

Any change of variables $\mathbf{U} \rightarrow \mathbf{V}$ can be achieved using the following relations:

$$\mathbf{U}_{,t} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial t} = \mathbf{A}_0 \mathbf{V}_{,t} \tag{11}$$

$$\mathscr{F}_{i,i} = \frac{\partial \mathscr{F}_i}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial x_i} = \mathbf{A}_i \mathbf{A}_0 \mathbf{V}_{,i} \tag{12}$$

Substituting these results into Equation (1) we arrive at the transformed system

$$\mathbf{A}_0\mathbf{V}_{,t} + \tilde{\mathbf{A}}_i\mathbf{V}_{,i} = \mathbf{F} \tag{13}$$

where $\tilde{\mathbf{A}}_i = \mathbf{A}_i\mathbf{A}_0$.

Symmetric forms are desirable first because they possess certain stability properties if entropy variables are used, and second because stabilized methods such as the streamline upwind Petrov–Galerkin (SUPG) method and its space–time version, the space–time Petrov–Galerkin (STPG) method, can be straightforwardly applied. As is well known, symmetric forms of hyperbolic systems can be constructed once the correspondent entropy function for the original non-symmetric hyperbolic system has been derived. If in the incomplete parabolic system (13) **V** is chosen to be the vector of entropy variables, a symmetric system is obtained. Moreover, it can be shown that the Galerkin finite element formulation satisfies the entropy condition. However, this is not the case of hyperbolic systems, where stability bounds can only be obtained with the addition of stabilizing terms, such as the SUPG and shock/discontinuity-capturing operators. This can be translated by the fact that, to well represent solutions exhibiting shocks/discontinuities and or sharp layers, these stabilized methods should be used.

Another type of symmetric form can be achieved using velocity/celerity variables. To this end let us define the change of variables $\mathbf{U} \to \mathbf{V} = (u_1, u_2, \theta)$, where $\theta = 2c$ and $c = \sqrt{gH}$ is the gravitational wave propagation velocity (celerity). For this change of variables we can write

$$\mathbf{A}_0 = \mathbf{U}_{,\mathbf{V}} = \frac{c}{g}\begin{bmatrix} c & 0 & u_1 \\ 0 & c & u_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$\tilde{\mathbf{A}}_1 = \mathbf{A}_1\mathbf{A}_0 = \frac{c}{g}\begin{bmatrix} 2cu_1 & 0 & c^2 + u_1^2 \\ cu_2 & cu_1 & u_1u_2 \\ c & 0 & u_1 \end{bmatrix} \tag{15}$$

$$\tilde{\mathbf{A}}_2 = \mathbf{A}_2\mathbf{A}_0 = \frac{c}{g}\begin{bmatrix} cu_2 & cu_1 & u_1u_2 \\ 0 & 2cu_2 & c^2 + u_2^2 \\ 0 & c & u_2 \end{bmatrix} \tag{16}$$

Substituting the above expressions into Equation (13) and pre-multiplying by $\mathbf{A}_0^{-1}$ we obtain the SWEs written in terms of velocity/celerity variables

$$\mathbf{V}_{,t} + \bar{\mathbf{A}}_i\mathbf{V}_{,i} = \bar{\mathbf{F}} \tag{17}$$

where

$$\bar{\mathbf{A}}_1 = \begin{bmatrix} u_1 & 0 & c \\ 0 & u_1 & 0 \\ c & 0 & u_1 \end{bmatrix}, \qquad \bar{\mathbf{A}}_2 = \begin{bmatrix} u_2 & 0 & 0 \\ 0 & u_2 & c \\ 0 & c & u_2 \end{bmatrix} \tag{18}$$

$$\bar{\mathbf{F}} = \begin{bmatrix} g\dfrac{\partial h}{\partial x_1} + fu_2 - \dfrac{\gamma u_1}{H} + \dfrac{\tau_1^s}{\rho H} + \dfrac{1}{H}\left(\dfrac{\partial}{\partial x_i}\left(\nu H \dfrac{\partial u_1}{\partial x_i}\right)\right) \\ g\dfrac{\partial h}{\partial x_2} - fu_1 - \dfrac{\gamma u_2}{H} + \dfrac{\tau_2^s}{\rho H} + \dfrac{1}{H}\left(\dfrac{\partial}{\partial x_i}\left(\nu H \dfrac{\partial u_2}{\partial x_i}\right)\right) \\ 0 \end{bmatrix} \tag{19}$$

For convenience, at this point we will change the notation by making $\mathbf{U} = \mathbf{V}$, $\mathbf{A} = \bar{\mathbf{A}}$, and rewrite Equation (17) as

$$\mathbf{U}_{,t} + \mathbf{A}\cdot\nabla\mathbf{U} = \bar{\mathbf{F}} \tag{20}$$

Assuming that the quantities $(1/H)\nabla H\cdot(\nu\nabla u_i)$ can be neglected, the viscous terms included in $\bar{\mathbf{F}}$ can be written in the form of the diffusion operator $\nabla\cdot(\boldsymbol{K}\nabla\mathbf{U})$, where $\boldsymbol{K}$ is the diffusivity matrix of the system

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{bmatrix}, \qquad \boldsymbol{K}_{ij} = \delta_{ij}\nu \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{21}$$

Here δ_{ij} denotes the Kronecker delta, i.e. $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ otherwise.

Finally, the SWEs may be written in terms of velocity/celerity variables as

$$\mathbf{U}_{,t} + \mathbf{A}\cdot\nabla\mathbf{U} - \nabla\cdot(\boldsymbol{K}\nabla\mathbf{U}) = \mathbf{F} \tag{22}$$

where the source terms are now given by

$$\mathbf{F} = \begin{bmatrix} g\dfrac{\partial h}{\partial x_1} + fU_2 - \dfrac{\gamma}{H}U_1 + \dfrac{1}{\rho H}\tau_1^s \\ g\dfrac{\partial h}{\partial x_2} - fU_1 - \dfrac{\gamma}{H}U_2 + \dfrac{1}{\rho H}\tau_2^s \\ 0 \end{bmatrix} \tag{23}$$

3. SPACE–TIME FORMULATION

At a time t of the period of observation $[0, T]\in\Re^+$, the spatial domain is mathematically represented by an open set $\Omega_t\in\Re^2$ with boundary Γ_t. For $n = 0, 1, 2, \ldots, N$ we will consider a partition of $[0, T]$ given by $t_0 = 0 < t_1 \cdots < t_n < t_{n+1} \cdots < t_N = T$. We denote by $I_n = (t_n, t_{n+1})$ the nth time interval and let $\Omega_n = \Omega_{t_n}$ and $\Gamma_n = \Gamma_{t_n}$. We will say that for each n, the space–time domain of interest is the 'slab' S_n, enclosed by Ω_n, Ω_{n+1} and $\bar{\Gamma}_n$, the lateral surface bounded by the curves Γ_n and Γ_{n+1} (see Figure 1). Then, for $n = 0, 1, 2, \ldots$, we will define a finite element partition of S_n such that

$$S_n = \bigcup_{e=1}^{(\mathrm{nel})_n} S_n^e, \qquad S_n^i \cap S_n^j = \varnothing, \quad \text{for } i \neq j \tag{24}$$

Here $(\mathrm{nel})_n$ is the number of elements in the space–time slab S_n.

Under the above definitions we will assume that the finite element subspace of weighting functions $\hat{\mathscr{U}}_n^h$ is the set of continuous piecewise polynomials in S_n, which may be discontinuous across the slab interfaces, i.e.

$$\hat{\mathscr{U}}_n^h \equiv \{\hat{\mathbf{U}}^h;\ \hat{\mathbf{U}}^h \in (C^0(S_n))^3;\ \hat{\mathbf{U}}^h|_{S_n^e} \in (P^k(S_n^e))^3;\ \hat{\mathbf{U}}^h|_{\bar{\Gamma}_n} = \mathbf{0}\} \tag{25}$$

In the above expression, P^k is the set of polynomials of degree less than or equal to k. For a prescribed boundary condition $\mathbf{g}$ on $\bar{\Gamma}_n$ a general trial function is then an element of $\mathscr{U}_n^h$:

$$\mathscr{U}_n^h \equiv \{\mathbf{U}^h;\ \mathbf{U}^h \in (C^0(S_n))^3;\ \mathbf{U}^h|_{S_n^e} \in (P^k(S_n^e))^3;\ \mathbf{U}^h|_{\bar{\Gamma}_n} = \mathbf{g}\} \tag{26}$$

Considering that the approximations are discontinuous at the slab interfaces, let us define the jump of $\mathbf{U}^h$ at time t_n

$$[\mathbf{U}^h(t_n^+)] = \mathbf{U}^h(t_n^+) - \mathbf{U}^h(t_n^-) \tag{27}$$

where

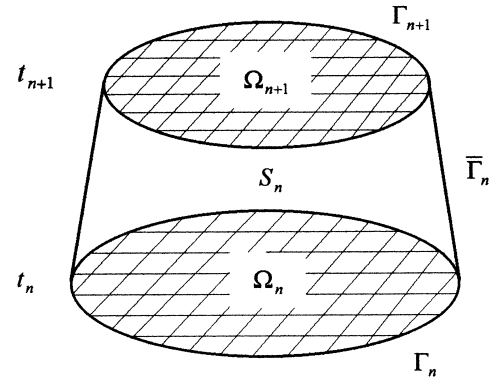


Figure 1. Space–time slab.

$$\mathbf{U}^h(t_n^{\pm}) = \lim_{\varepsilon \to 0} \mathbf{U}^h(t_n \pm \varepsilon) \tag{28}$$

The space–time finite element formulation consists of: given $\mathbf{U}(t_0)$, for each S_n, $n = 0, 1, 2, \ldots$, find $\mathbf{U}^h \in \mathscr{U}_n^h$ such that for all $\hat{\mathbf{U}}^h \in \hat{\mathscr{U}}_n^h$ the following variational equation is satisfied:

$$\int_{S_n} \mathbf{R} \cdot \hat{\mathbf{U}}^h \, \mathrm{d}\Omega \, \mathrm{d}t + \int_{\Omega} [\mathbf{U}^h(t_n^+)] \cdot \hat{\mathbf{U}}^h(t_n^+) \, \mathrm{d}\Omega + \sum_{e=1}^{(\mathrm{nel})_n} \int_{S_n^e} \mathbf{R} \cdot \boldsymbol{\tau}(\hat{\mathbf{U}}_{,t}^h + \mathbf{A} \cdot \nabla \hat{\mathbf{U}}^h) \, \mathrm{d}\Omega \, \mathrm{d}t$$
$$+ \sum_{e=1}^{(\mathrm{nel})_n} \int_{S_n^e} \bar{\tau} \nabla \mathbf{U}^h \cdot \nabla \hat{\mathbf{U}}^h \, \mathrm{d}\Omega \, \mathrm{d}t = 0 \tag{29}$$

The residual $\mathbf{R}$ associated with the approximate solution $\mathbf{U}^h$ is given by

$$\mathbf{R}(\mathbf{U}^h) = \mathbf{U}_{,t}^h + \mathbf{A}(\mathbf{U}^h) \cdot \nabla \mathbf{U}^h - \nabla \cdot (\boldsymbol{K} \nabla \mathbf{U}^h) - \mathbf{F}(\mathbf{U}^h) \tag{30}$$

The first two integrals in Equation (29) correspond to the time-discontinuous Garlerkin method. The third integral is the contribution from the STPG term, while the last integral corresponds to the discontinuity-capturing term.

The 3×3 symmetric positive matrix $\boldsymbol{\tau}$ of intrinsic time scales and the scalar $\bar{\tau}$ measure the necessary amount of diffusion to be added to the system. Using the definition found in Reference [12] we have

$$\boldsymbol{\tau} = \left[\left(\frac{\partial \xi_0}{\partial x_0} \right)^2 \mathbf{I}_3 + \frac{\partial \xi_i}{\partial x_j} \frac{\partial \xi_i}{\partial x_k} \mathbf{A}_j \mathbf{A}_k + \left(\frac{\partial \xi_i}{\partial x_k} \frac{\partial \xi_j}{\partial x_l} \frac{\partial \xi_i}{\partial x_m} \frac{\partial \xi_j}{\partial x_n} \right) \boldsymbol{K}_{kl} \boldsymbol{K}_{mn} \right]^{-1/2} \tag{31}$$

where $x_0 = t$, ξ_i $(i = 0, 1, 2)$ are the local co-ordinates of the parent element S_n^e and $\mathbf{I}_3$ denotes the 3×3 identity matrix.

For the consistent approximate upwind (CAU) method proposed in References [13,14], the scalar $\bar{\tau}$ has the form

$$\bar{\tau} = \frac{(\mathbf{R}^T \boldsymbol{\tau}_c \mathbf{R})}{|\nabla \mathbf{U}^h|^2} \tag{32}$$

which ensures that a quadratic weighted residual term is always added to the variational equation. In fact, for $\hat{\mathbf{U}}^h \equiv \mathbf{U}^h$, the CAU discontinuity-capturing term becomes

$$\|\mathbf{R}\|_{\tau_c}^2 = \sum_{e=1}^{(\mathrm{nel})_n} \int_{S_n^e} \mathbf{R}^T \boldsymbol{\tau}_c \mathbf{R} \, \mathrm{d}\Omega \, \mathrm{d}t \tag{33}$$

The definition for the matrix $\boldsymbol{\tau}_c$ can be found in Reference [13], leading to

$$\bar{\tau} = \begin{cases} \max\left(0, \dfrac{|\mathbf{R}|}{|\nabla_{\xi}\mathbf{U}^h|} - \dfrac{(\mathbf{U},_t^h + \mathbf{A}\cdot\nabla\mathbf{U}^h)^T\boldsymbol{\tau}\mathbf{R}}{|\nabla\mathbf{U}^h|^2}\right), & \text{if } \nabla\mathbf{U}^h \neq \mathbf{0} \\ 0, & \text{if } \nabla\mathbf{U}^h = \mathbf{0} \end{cases} \tag{34}$$

The subscript ξ denotes differentiation with respect to the local co-ordinates ξ_i $(i = 1, 2)$.

Remark
Note that for hyperbolic problems $\mathbf{R} = \mathbf{U},_t^h + \mathbf{A}\cdot\nabla\mathbf{U}^h$, which means that for $\hat{\mathbf{U}}^h \equiv \mathbf{U}^h$ the STPG term also leads to a positive weighted square residual term

$$\|\mathbf{R}\|_{\tau}^2 = \sum_{e=1}^{(\text{nel})_n} \int_{S_n^e} \mathbf{R}^T\boldsymbol{\tau}\mathbf{R}\, \mathrm{d}\Omega\, \mathrm{d}t \tag{35}$$

Let us assume that the spatial domain does not change from time t_n to time t_{n+1}, i.e. $\Omega_n \equiv \Omega_{n+1}$ and $\Gamma_n \equiv \Gamma_{n+1}$. Defining a spatial discretization for Ω_n such that

$$\Omega_n = \bigcup_{e=1}^{(\text{nel})_n} \Omega_n^e, \qquad \Omega_n^i \cap \Omega_n^j = \varnothing \quad \text{for } i \neq j \tag{36}$$

each slab S_n with boundary $\bar{\Gamma}_n$ may be defined by

$$S_n = \Omega_n \times I_n, \qquad \bar{\Gamma}_n = \Gamma \times I_n \tag{37}$$

Then each Ω_n^e is associated with the space–time finite element $S_n^e = \Omega_n^e \times I_n$. Under these circumstances, the linear space–time finite element approximations over each slab S_n will be given by

$$\mathbf{U}^h = \sum_{j=1}^{(\text{nnode})_n} \varphi_j(x_1, x_2)[N_1(t)\mathbf{U}_{n+}^j + N_2(t)\mathbf{U}_{n+1-}^j] \tag{38}$$

$$\hat{\mathbf{U}}^h = \sum_{i=1}^{(\text{nnode})_n} \varphi_i(x_1, x_2)[N_1(t)\hat{\mathbf{U}}_{n+}^i + N_2(t)\hat{\mathbf{U}}_{n+1-}^i] \tag{39}$$

where $(\text{nnode})_n$ is the number of nodal points of Ω_n, φ_i is the linear spatial interpolation function associated with each node i in Ω_n and N_1, N_2 are the linear time interpolation functions associated with $t = t_n^+$ and $t = t_{n+1}^-$ respectively. The unknown nodal values of $\mathbf{U}^h$ and $\hat{\mathbf{U}}^h$ are denoted by $\mathbf{U}_{n+}^j$, $\mathbf{U}_{n+1-}^j$ and $\hat{\mathbf{U}}_{n+}^j$, $\hat{\mathbf{U}}_{n+1-}^j$.

Introducing the above approximations in the variational equation (29), the following non-linear discrete system of equations is obtained:

$$\mathbf{KU} = \mathbf{F} \tag{40}$$

where

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} \mathbf{U}_{n+} \\ \mathbf{U}_{n+1-} \end{bmatrix} \tag{41}$$

$$\mathbf{K}_{11} = \mathbf{K}_{11}^{\text{ad}} + \mathbf{M}_{11}^{\text{G}} + \mathbf{M} \tag{42}$$

$$\mathbf{K}_{12} = \mathbf{K}_{12}^{\text{ad}} + \mathbf{M}_{12}^{\text{G}} \tag{43}$$

$$\mathbf{K}_{21} = \mathbf{K}_{21}^{\text{ad}} + \mathbf{M}_{21}^{\text{G}} \tag{44}$$

$$\mathbf{K}_{22} = \mathbf{K}_{22}^{\text{ad}} + \mathbf{M}_{22}^{\text{G}} \tag{45}$$

The matrices $\mathbf{K}_{ij}^{\text{ad}}$ are the result of the weighting of the advective and diffusive terms and the vector $\mathbf{F}_i$ is the weighted source term

$$\mathbf{K}_{ij}^{\text{ad}} = \mathbf{K}_{ij}^{\text{G}} + \mathbf{K}_{ij}^{\text{PG}} + \mathbf{K}_{ij}^{\text{DC}} \tag{46}$$

$$\mathbf{F}_1 = \mathbf{F}_1^{\text{G}} + \mathbf{F}_1^{\text{PG}} + \mathbf{M}\mathbf{U}_{n-}^{h} \tag{47}$$

$$\mathbf{F}_2 = \mathbf{F}_2^{\text{G}} + \mathbf{F}_2^{\text{PG}} \tag{48}$$

The superscripts G, PG and DC refer to the contributions of the Galerkin, Petrov–Galerkin and discontinuity-capturing operators, and the indices i, j refer to the time interpolation functions N_i, N_j $(i, j = 1, 2)$. The matrices $\mathbf{M}_{ij}^{\text{G}}$ come from the Galerkin term $\int_{S_n} \mathbf{U}_{,t}^{h} \cdot \hat{\mathbf{U}}^h \, \mathrm{d}\Omega \, \mathrm{d}t$ and $\mathbf{M}$ comes from the jump term.

4. SEMI-DISCRETE FORMULATION

In the semi-discrete formulation, the spatial domain is a fixed two-dimensional domain Ω with boundary Γ, which is partitioned into nel elements Ω^e

$$\Omega = \bigcup_{e=1}^{\text{nel}} \Omega^e, \qquad \Omega^i \cap \Omega^j = \varnothing \quad \text{for } i \neq j \tag{49}$$

The finite element subspaces considered at time $t = t_n$ are given by

$$\hat{\mathcal{U}}_n^h \equiv \{\hat{\mathbf{U}}^h;\ \hat{\mathbf{U}}^h \in (C^0(\Omega))^3;\ \hat{\mathbf{U}}^h|_{\Omega^e} \in (P^k(\Omega^e))^3;\ \hat{\mathbf{U}}^h|_{\Gamma} = \mathbf{0}\} \tag{50}$$

$$\mathcal{U}_n^h \equiv \{\mathbf{U}^h;\ \mathbf{U}^h \in (C^0(\Omega))^3;\ \mathbf{U}^h|_{\Omega^e} \in (P^k(\Omega^e))^3;\ \mathbf{U}^h|_{\Gamma} = \mathbf{g}\} \tag{51}$$

The problem now consists of: for a given $\mathbf{U}(t_0)$ and for each time t_n, $n = 1, 2, \ldots$, find $\mathbf{U}^h \in \mathcal{U}_n^h$ such that for all $\hat{\mathbf{U}}^h \in \hat{\mathcal{U}}_n^h$ the following variational form is satisfied:

$$\int_{\Omega} \mathbf{R}\cdot\hat{\mathbf{U}}^h\,d\Omega + \sum_{e=1}^{\text{nel}} \int_{\Omega^e} \mathbf{R}\cdot(\tau\mathbf{A}\cdot\nabla\hat{\mathbf{U}}^h)\,d\Omega + \sum_{e=1}^{\text{nel}} \int_{\Omega^e} \bar{\tau}\nabla\mathbf{U}^h\cdot\nabla\hat{\mathbf{U}}^h\,d\Omega = 0 \tag{52}$$

where

$$\tau = \left[\frac{\partial\xi_i}{\partial x_j}\frac{\partial\xi_i}{\partial x_k}\mathbf{A}_j\mathbf{A}_k + \left(\frac{\partial\xi_i}{\partial x_k}\frac{\partial\xi_j}{\partial x_l}\frac{\partial\xi_i}{\partial x_m}\frac{\partial\xi_j}{\partial x_n}\right)\boldsymbol{K}_{kl}\boldsymbol{K}_{mn}\right]^{-1/2} \tag{53}$$

and $\bar{\tau}$ is the same as in Equation (34).

Remark
The SUPG results in a positive weighted square residual term only in time-independent hyperbolic problems.

In the semi-discrete version, the finite element approximations depend only on the spatial variables. Thus, for time $t = t_n$, the approximation functions are given by

$$\mathbf{U}^h = \sum_{j=1}^{\text{nnode}} \varphi_j(x, y)\mathbf{U}_n^j \tag{54}$$

$$\hat{\mathbf{U}}^h = \sum_{i=1}^{\text{nnode}} \varphi_i(x, y)\hat{\mathbf{U}}_n^i \tag{55}$$

where $\mathbf{U}_n^j$ are the unknowns values of $\mathbf{U}^h$ for node j, $\hat{\mathbf{U}}_n^i$ are the nodal values of $\hat{\mathbf{U}}^h$ and φ_j is the spatial interpolation function for node j. The subscript n refers to time t_n and nnode is the total number of nodes.

Substituting approximations (54) and (55) into the variational formulation (52), the following ordinary differential system of equations is obtained:

$$\mathbf{M}\dot{\mathbf{U}}_n + \mathbf{K}\mathbf{U}_n = \mathbf{F} \tag{56}$$

where $\mathbf{U}_n$ is the vector of nodal values of $\mathbf{U}^h$, $\dot{\mathbf{U}}_n$ represents the time derivatives $(\partial\mathbf{U}^h/\partial t)_{t=t_n}$, and $\mathbf{M}$, $\mathbf{K}$ and $\mathbf{F}$ are equal to

$$\mathbf{M} = \mathbf{M}^{\mathrm{G}} + \mathbf{M}^{\mathrm{PG}} \tag{57}$$

$$\mathbf{K} = \mathbf{K}^{\mathrm{G}} + \mathbf{K}^{\mathrm{PG}} + \mathbf{K}^{\mathrm{DC}} \tag{58}$$

$$\mathbf{F} = \mathbf{F}^{\mathrm{G}} + \mathbf{F}^{\mathrm{PG}} \tag{59}$$

The time derivatives can be approximated by a finite difference scheme, such as the trapezoidal rule

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t\dot{\mathbf{U}}_{n+\alpha} \tag{60}$$

$$\dot{\mathbf{U}}_{n+\alpha} = (1-\alpha)\dot{\mathbf{U}}_n + \alpha\dot{\mathbf{U}}_{n+1} \tag{61}$$

The parameter α controls the precision and stability of the method. The values $\alpha = 0.5$ and $\alpha = 1.0$ correspond to the well-known Crank–Nicolson and Euler-backward approximations.

5. SOLUTION OF THE NON-LINEAR SYSTEM OF EQUATIONS

Both systems of Equations (40) and (56) can be solved using predictor multi-corrector algorithms, derived from the Newton–Raphson family methods. For system (40), this algorithm has the form

$$\mathbf{U}^0 = \begin{bmatrix} \mathbf{U}^0_{n+} \\ \mathbf{U}^0_{n+1-} \end{bmatrix} \left.\vphantom{\begin{bmatrix} a \\ b \end{bmatrix}}\right\} \text{(predictor phase)} \tag{62}$$

For $i = 0, 1, 2, \ldots$

$$\left.\begin{aligned} &\mathbf{R}^i = \mathbf{F} - \mathbf{K}\mathbf{U}^i \\ &\mathbf{K}\Delta\mathbf{U}^i = \mathbf{R}^i \\ &\mathbf{U}^{i+1} = \mathbf{U}^i + \Delta\mathbf{U}^i \end{aligned}\right\} \text{(multi-corrector phase)} \tag{63}$$

In above, the starting vectors $\mathbf{U}^0_{n+}$, $\mathbf{U}^0_{n+1-}$ can be both taken as $\mathbf{U}_{n-}$ (see Figure 2). Note that the space–time discrete system has ($2 \times$ neq) equations, where neq is the number of equations arising from the equivalent semi-discrete discretization. Therefore, it can be conveniently split into two coupled systems with neq equations each

$$\mathbf{K}_{11}\mathbf{U}_1 + \mathbf{K}_{12}\mathbf{U}_2 = \mathbf{F}_1 \tag{64}$$

$$\mathbf{K}_{21}\mathbf{U}_1 + \mathbf{K}_{22}\mathbf{U}_2 = \mathbf{F}_2 \tag{65}$$

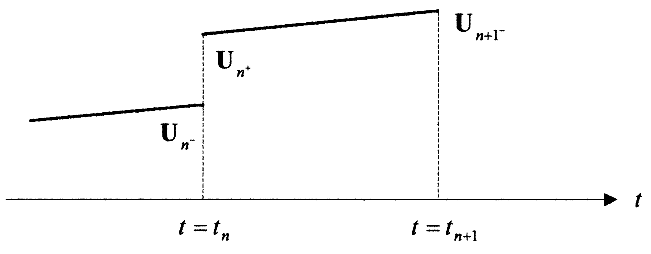


Figure 2. Starting solution for the time interval $I_n = (t_n, t_{n+1})$: $\mathbf{U}^0_{n+} \equiv \mathbf{U}^0_{n+1-} \equiv \mathbf{U}_{n-}$.

For simplicity, we have used the notation $\mathbf{U}_1 = \mathbf{U}_{n+}$ and $\mathbf{U}_2 = \mathbf{U}_{n+1-}$. The corresponding predictor multi-corrector algorithm is given by

For $i = 0, 1, 2, \ldots$

$$\left.\begin{array}{l} \mathbf{R}_1^i = \mathbf{F}_1 - \mathbf{K}_{11}\mathbf{U}_1^i - \mathbf{K}_{12}\mathbf{U}_2^i \\ \mathbf{K}_{11}\Delta\mathbf{U}_1^i = \mathbf{R}_1^i \\ \mathbf{U}_1^{i+1} = \mathbf{U}_1^i + \Delta\mathbf{U}_1^i \\ \mathbf{R}_2^{\mathrm{i}} = \mathbf{F}_2 - \mathbf{K}_{21}\mathbf{U}_1^{\mathrm{i}+1} - \mathrm{K}_{22}\mathbf{U}_2^{\mathrm{i}} \\ \mathbf{K}_{22}\Delta\mathbf{U}_2^{\mathrm{i}} = \mathbf{R}_2^{\mathrm{i}} \\ \mathbf{U}_2^{\mathrm{i}+1} = \mathbf{U}_2^{\mathrm{i}} + \Delta\mathbf{U}_2^{\mathrm{i}} \end{array}\right\} \text{(multi-corrector phase)} \tag{66}$$

For the semi-discrete system we can write

$$\left.\begin{array}{l} \mathbf{U}_{n+1}^0 = \mathbf{U}_n + \Delta t(1-\alpha)\dot{\mathbf{U}}_n \\ \dot{\mathbf{U}}_{n+1}^0 = \mathbf{0} \end{array}\right\} \text{(predictor phase)} \tag{67}$$

For $i = 0, 1, 2, \ldots$

$$\left.\begin{array}{l} \mathbf{R}^i = \mathbf{F}_{n+1} - \mathbf{M}\dot{\mathbf{U}}_{n+1}^i - \mathbf{K}\mathbf{U}_{n+1}^i \\ \mathbf{M}^*\Delta\dot{\mathbf{U}}_{n+1}^i = \mathbf{R}^i \\ \dot{\mathbf{U}}_{n+1}^{i+1} = \dot{\mathbf{U}}_{n+1}^i + \Delta\dot{\mathbf{U}}_{n+1}^i \\ \mathbf{U}_{n+1}^{i+1} = \mathbf{U}_{n+1}^0 + \alpha\Delta t\dot{\mathbf{U}}_{n+1}^{i+1} \end{array}\right\} \text{(multi-corrector phase)} \tag{68}$$

where

$$\mathbf{M}^* = \mathbf{M} + \alpha\Delta t\mathbf{K} \tag{69}$$

As can be seen, each non-linear iteration involves three steps

$$\text{Evaluate the non-linear residual } \mathbf{R}^i(\mathbf{U}^i) \tag{70}$$

$$\text{Solve the system } \mathbf{K}(\mathbf{U}^i)\Delta\mathbf{U}^i = \mathbf{R}^i \tag{71}$$

$$\text{Update solution } \mathbf{U}^{i+1} = \mathbf{U}^i + \Delta\mathbf{U}^i \tag{72}$$

Given a tolerance ε, the non-linear process stops when

$$\|\mathbf{R}^i\| < \varepsilon\|\mathbf{R}^0\| \tag{73}$$

Iterative solvers are especially suitable for this type of algorithm, as the accuracy required to solve Equation (71) depends on the non-linear residual $\mathbf{R}^i$. The number of linear iterations can be drastically reduced using a variable tolerance γ to control the linear residual

$$(\mathbf{K}\Delta\mathbf{U}^i - \mathbf{R}^i) < \gamma\mathbf{R}^i \tag{74}$$

This kind of procedure gives rise to *inexact Newton methods* [15,16]. The results presented in this paper were obtained using the formula [17]

$$\gamma = \min\left(\gamma_0, \left(\frac{\|\mathbf{R}^i\|}{\|\mathbf{R}^0\|}\right)^{1/2}\right) \tag{75}$$

where γ_0 is the minimum initial value for γ.

Another advantage of iterative solvers is that edge-based data structures can be exploited to efficiently perform the matrix–vector products appearing in Equation (71). The required edge-based matrices can be assembled in the element-based loop used in Equation (70), where the element matrices and the non-linear residual are evaluated in the usual manner. In next section, the benefits of using the edge-based approach will be examined in details.

6. ITERATIVE SOLUTION USING EDGE-BASED DATA STRUCTURE

As it was shown in the previous section, both semi-discrete and space–time formulations lead to a non-linear system of equations which are solved by a sequence of linear systems, where the coefficient matrix is non-symmetric with a symmetric skyline profile. Obviously, direct methods based on Gaussian elimination can be used to solve such systems. However, as the number of equations increases, these methods become memory and CPU expensive when compared with iterative solvers. Among the latter, one of the most popular and efficient is the GMRES solver, designed for non-symmetric systems. This method, developed by Saad and Schultz [18], minimizes the residual norm and is based on the generation of Krylov spaces. The performance of any iterative method can be significantly improved by using some type of preconditioning. Preconditioning simply means transforming the original system of equations into another one having the same solution, but which is easier to be solved. A full and detailed discussion on iterative methods and preconditioners can be found in Reference [19]. Another way to improve performance is to optimize the code, reducing memory requirements, the number of floating point operations (flop) and the number of indirect addressing (i/a) operations.

The core of the GMRES algorithm lies on three operations

$$\mathbf{x} = \mathbf{x} + \alpha\mathbf{p} \qquad \text{(Vector update)} \tag{76}$$

$$\alpha = \mathbf{r}^t\mathbf{r} \qquad \text{(Dot product)} \tag{77}$$

$$\mathbf{p} = \mathbf{K}\mathbf{u} \qquad \text{(Matrix–vector product)} \tag{78}$$

Operations of type (76) and (77) can be executed in a very efficient way in parallel/vector machines, as well as in scalar machines. Clearly, an operation of type (78) is the one with the highest computational demand. Mainly in non-linear, time-dependent problems, as is the case of the SWEs, this operation is repeated many times, and special attention must be given to it. This kind of operation is usually performed at an element-based approach, but a great improvement can be obtained if an edge-based data structure is used.

6.1. Edge-based data structure

We know beforehand that the matrix **K** is sparse as it is a result of a finite element discretization. To take advantage of this characteristic, the global matrix **K** does not need to be assembled. The usage of element matrices guarantees that only non-zero coefficients will be stored. Unlike the global storage, the element level storage does not depend on the bandwidth of the system. Using the element level storage, the matrix–vector product can be performed at an element-by-element basis

$$\mathbf{p} = \mathop{\mathsf{A}}_{e=1}^{\mathrm{nel}} \mathbf{K}^{\mathrm{e}}\mathbf{u}^{\mathrm{e}} \tag{79}$$

In the above expression, A is the assembly operator, $\mathbf{K}^{\mathrm{e}}$ are the element matrices, $\mathbf{u}^{\mathrm{e}}$ are components of **u** referred to as local degrees of freedom (df) and nel is the total number of elements.

Let us consider a mesh composed by nel elements of a given type, with nnode nodal points. Denoting

neq as the total number of equations
nne as the number of nodes per element
ndf as the number of degrees of freedom per node
nd as the number of degrees of freedom per element

the number of stored coefficients per element is equal to nd^2, for non-symmetric matrices. In each pass of the element loop, the number of floating point operations is $2 \times \mathrm{nd}^2$ and the number of indirect addressing is $3 \times \mathrm{nd}$. The performance of the element-by-element matrix–vector product algorithm can be improved using the technique proposed by Gijzen in Reference [20]. This technique consists in splitting the product in

$$\mathbf{p} = (\mathrm{diag}\,\mathbf{K})\mathbf{u} + \mathop{\mathsf{A}}_{e=1}^{\mathrm{nel}} (\mathbf{K}^{\mathrm{e}} - \mathrm{diag}\,\mathbf{K}^{\mathrm{e}})\mathbf{u}^{\mathrm{e}} \tag{80}$$

where diag **K** is the block diagonal of **K**, with each block having dimensions (ndf, ndf) as shown in Figure 3. Extracting the block diagonal from the element matrices, the number of stored coefficients for each element is reduced to $\mathrm{nen} \times (\mathrm{nen} - 1) \times \mathrm{ndf}^2$. The number of flops in each pass of the element loop is reduced to $2 \times \mathrm{nen} \times (\mathrm{nen} - 1) \times \mathrm{ndf}^2$ and the number of i/a operations is not changed.

With the above element-by-element techniques, the sparsity of the system is fully exploited by the element level storage, and the assemblage of a global nodal block diagonal reduces the amount of necessary memory and the number of flops as well. However, coefficients relating

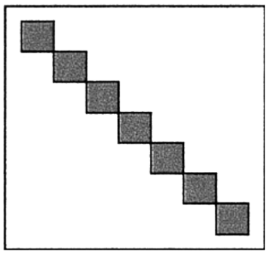


Figure 3. Block diagonal.

two different nodes still remain spread over the contributing elements, as shown in Figure 4. In this figure, two elements (A and B) contribute to the global coefficient $s(i, j)$. Both contributions $s^A(i, j)$ and $s^B(i, j)$ are stored in the corresponding element matrices. If the edge connecting node i to node j is used to store the coefficient $s(i, j)$, the spreading of element contributions is avoided. This can be done performing a loop on the elements and assembling the coefficients by edges. Figure 5 shows these edges for the linear triangle and its correspondent space–time element, the wedge-shaped element.

Using edge-based matrices, the matrix–vector product given in Equation (80) is replaced by

$$\mathbf{p} = (\text{diag}\,\mathbf{K})\mathbf{u} + \mathop{\mathbf{A}}_{a=1}^{\text{nedge}} (\mathbf{K}^a - \text{diag}\,\mathbf{K}^a)\mathbf{u}^a \tag{81}$$

where $(\mathbf{K}^a - \text{diag}\,\mathbf{K}^a)$ are the edge matrices containing only off-diagonal coefficients and nedge is the total number of edges. The edge-based algorithm for matrix–vector product is shown in the bullet list below. The number of stored coefficients per edge is equal to $2 \times \text{ndf}^2$. The number of flops is $4 \times \text{ndf}^2$ and the number of i/a operations is $6 \times \text{ndf}$ for each pass of the edge-based loop.

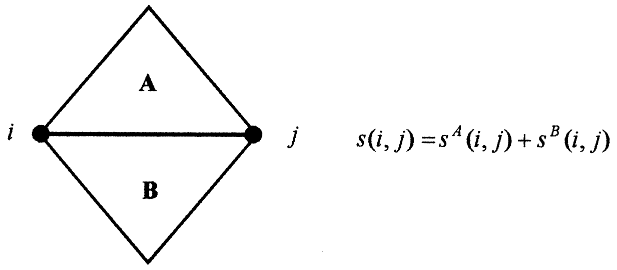


Figure 4. Element contributions to an off-diagonal coefficient.

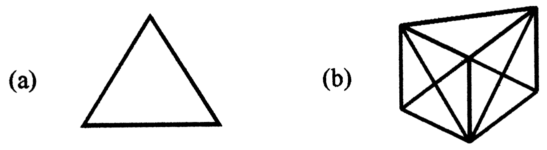


Figure 5. Element edges: (a) triangle (three edges); (b) wedge-shaped element (15 edges).

General algorithm for edge-based matrix–vector product

- Perform global product: $\mathbf{p} = (\text{diag}\,\mathbf{K})\mathbf{u}$.
- Loop on edges:
  - Recover the global numbering of the local degrees of freedom.
  - Gathering operation: $\mathbf{u}^a \leftarrow \mathbf{u}$.
  - Perform product: $\mathbf{p}^a = (\mathbf{K}^a - \text{diag}\,\mathbf{K}^a)\mathbf{u}^a$.
  - Scattering and accumulation operation: $\mathbf{p} \leftarrow \mathbf{p}^a$.
- End of loop.

Tables I and II compare the edge-based (EDGE) and the element-based approaches for the linear triangle and the wedge-shaped element. The element-by-element techniques are denoted by EBE (full element matrices) and EBE-BD (block diagonal assembled). As can be noted in these tables, the reduction in terms of memory and flops is almost the same for both elements. However, in terms of i/a operations there is a significant increase for the wedge-shaped element and no reduction for the linear triangle. Also, it is seen that the number of flops per element is twice the number of i/a operations. The number of i/a operations can be reduced by performing the main loop over groups of edges rather than over individual edges [8]. Some of these groups are illustrated in Figure 6. The corresponding numbers of i/a operations is indicated in Table III.

Table I. Linear triangle: number of stored coefficients (nc), flops and i/a operations per element (ndf = 3; nedge/nel = 1.5).

	nc	flop	i/a
EBE	81	162	27
EBE-BD	54	108	27
EDGE	27	54	27

Table II. Wedge-shaped element: number of stored coefficients (nc), flops and i/a operations per element (ndf = 3; nedge/nel = 6.5).

	nc	flop	i/a
EBE	324	648	54
EBE-BD	270	540	54
EDGE	117	234	117

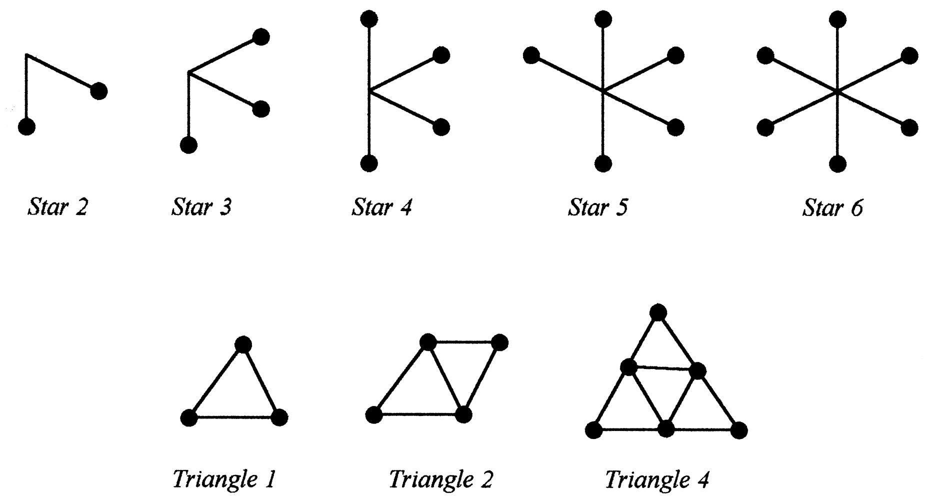


Figure 6. Groups of edges.

Table III. Number of i/a operations per edge (ndf = 3).

	i/a
Edge	18
Star 2	13.5
Star 3	12
Star 4	11.25
Star 5	10.8
Star 6	10.5
Triangle 1	9
Triangle 2	7.2
Triangle 4	6

For the wedge-shaped element, the edges can be separated into three sets, as shown in Figure 7. When solving the space–time system (66), only sets 1 and 2 are required (if the non-linear residuals $\mathbf{R}_1^i$ and $\mathbf{R}_2^i$ are evaluated at an element-by-element basis). Moreover, if the spatial discretization is the same for both time levels, we have that set $1 \equiv$ set 2.

7. NUMERICAL EXAMPLES

In this section three numerical examples are presented. The first test case is the one-dimensional dam break problem, for which the stabilized solutions in semi-discrete and

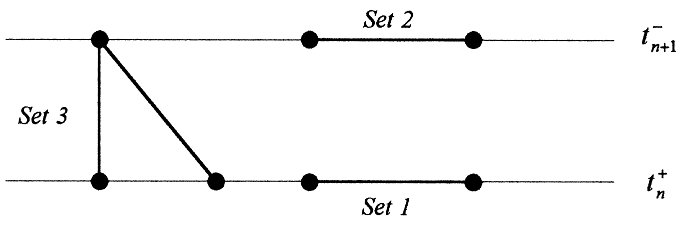


Figure 7. Edges belonging to triangulation in t_{n+} (set 1), edges belonging to triangulation in t_{n+1-} (set 2) and edges connecting both triangulations (set 3).

space–time versions are compared with the exact solution. The other two examples test the performance of the GMRES solver on a scalar machine (single 400 MHz Pentium II CPU) using element- and edge-based approaches. The second example is a two-dimensional extension of the first, and the third one is the simulation of a tidal flow. In all test cases a simple diagonal preconditioning has been used and the dimension of the Krylov subspace was set to 10. For the non-linear iterations the tolerance was set to $\varepsilon = 10^{-4}$, and a variable tolerance, with $\gamma_0 = 10^{-1}$ was used for the GMRES.

7.1. One-dimensional dam break

This problem consists of a wall separating two undisturbed water levels that is suddenly removed (Figure 8(a)). Friction effects are neglected and the spatial discretization is given by a 4×100 triangular elements mesh, as illustrated in Figure 8(b). Figures 9 and 10 show the results for $t = 7.50$, with $\Delta t = 0.25$. In these figures, we show the approximate solutions for the water elevation and the velocity obtained with the pure Galerkin, the STPG and the CAU methods. The results plotted in these figures show a remarkable accuracy improvement when the CAU method is employed. In Figures 11 and 12 we also present the results corresponding

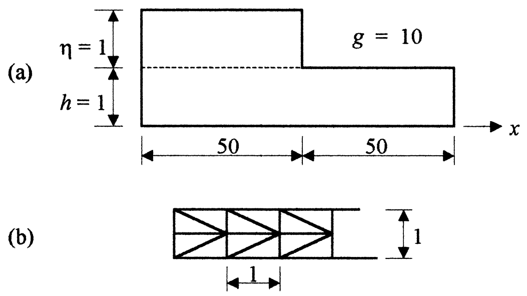


Figure 8. One-dimensional dam break problem: (a) geometry; (b) mesh.

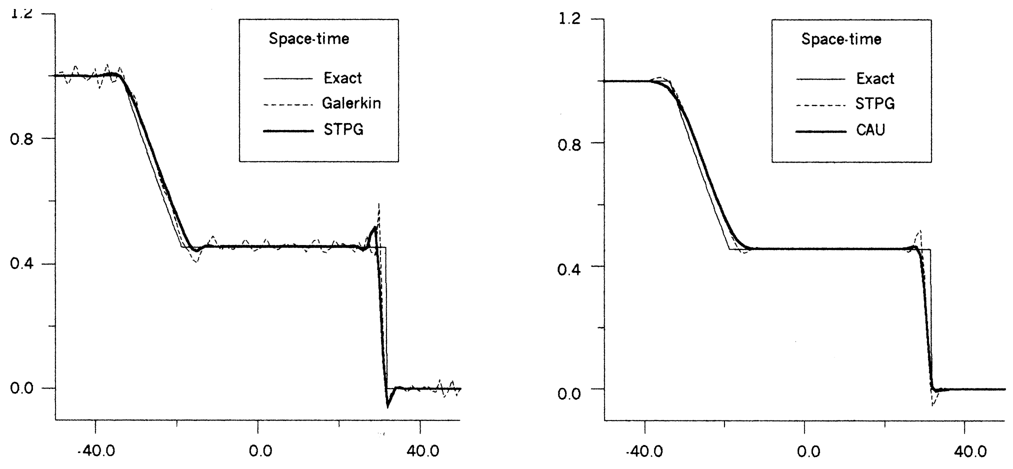


Figure 9. Water elevation using space–time elements.

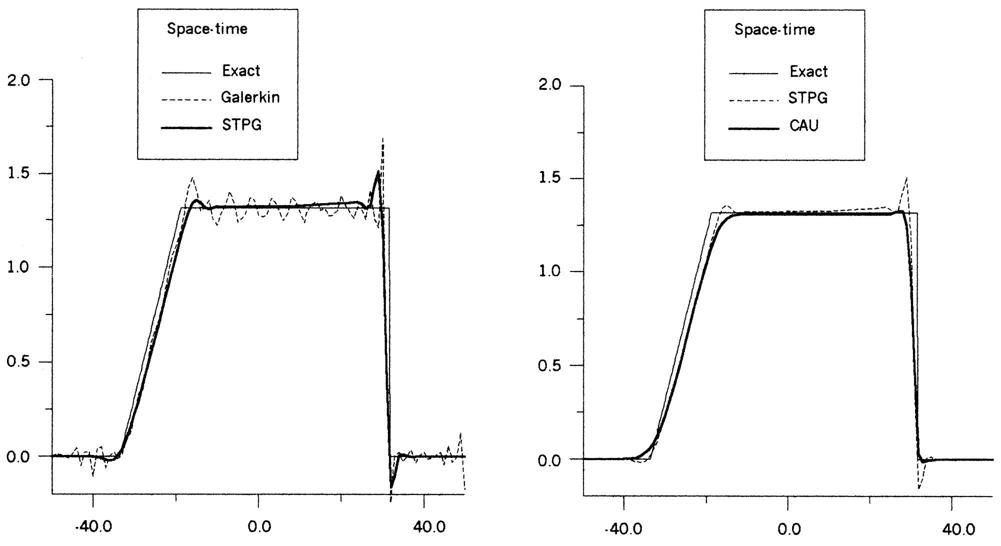


Figure 10. Velocities using space–time elements.

to the semi-discrete version of these methods. As expected, the approximations provided by the CAU method present the best results. Figure 13 shows the results in terms of the Froude number. As can be noted in this figure, the flow occurs under sub-critical conditions.

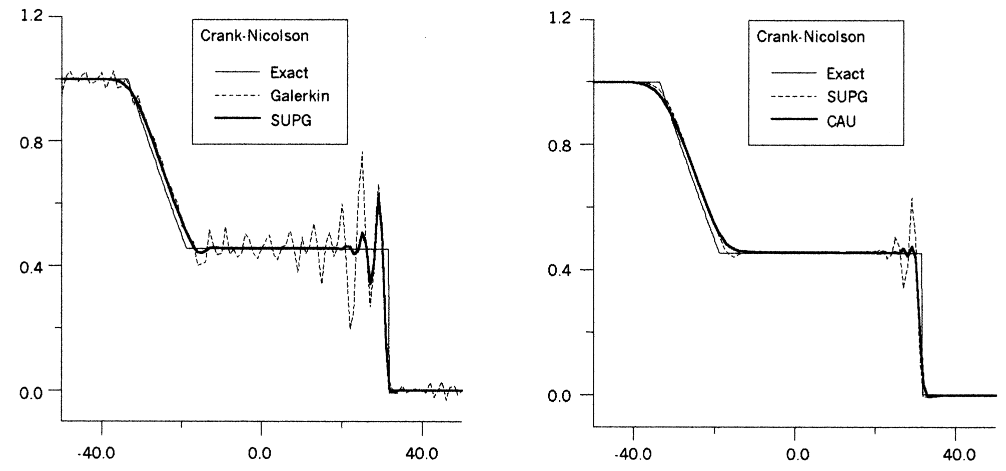


Figure 11. Water elevation using semi-discrete formulation (Crank–Nicolson).

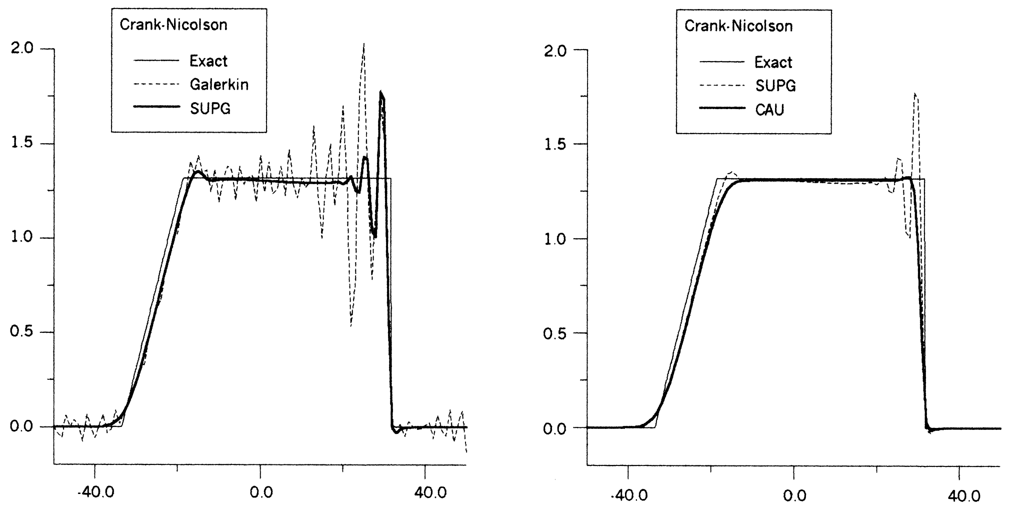
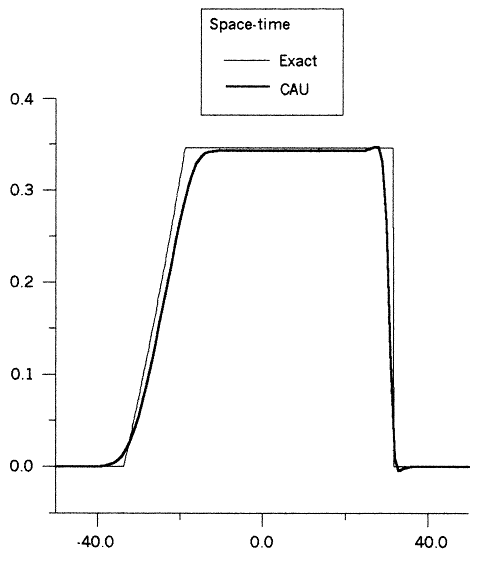


Figure 12. Velocities using Crank–Nicolson.

7.2. Two-dimensional dam break

This example is a two-dimensional extension of the first, as shown in Figure 14(a). The spatial discretization is given by a $2 \times 100 \times 100$ triangular elements mesh (Figure 14(b)), resulting in 131001 edges and 59606 equations for the space–time solution and 30200 edges and 29803



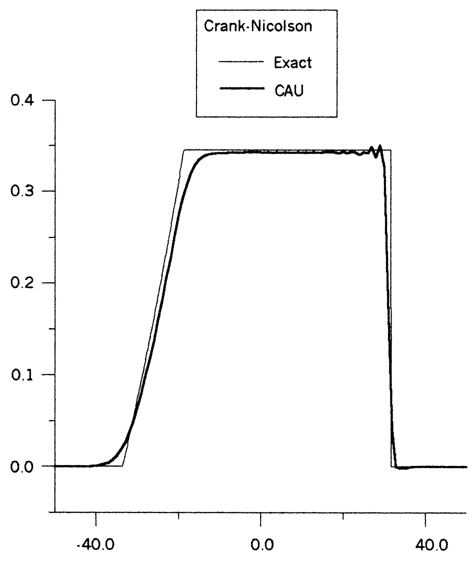


Figure 13. Froude number.

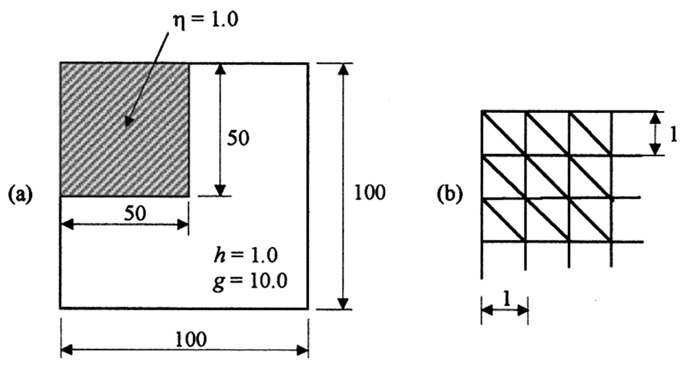


Figure 14. Two-dimensional dam break problem: (a) geometry; (b) discretization.

equations for the correspondent semi-discrete version. The semi-discrete and space–time solutions for $t = 7.5$ can be seen in Figure 15. The time step is the same as in the one-dimensional case ($\Delta t = 0.25$). The advantages of using an edge data structure can be seen in Tables IV and V. These tables present the CPU time spent by the GMRES solver to reach the solution at $t = 30.0$, or 120 time steps. Element-by-element techniques are denoted by EBE (full element matrices) and EBE-BD (block diagonal assembled). In the semi-discrete formulation, the edge-based approach is denoted by EDGE. In the edge-based space–time

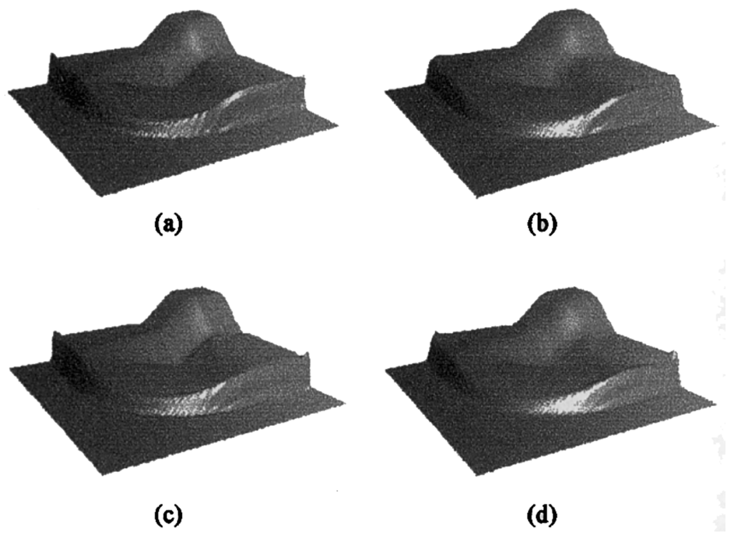


Figure 15. Water elevation for $t = 7.5$: (a) SUPG; (b) semi-discrete CAU; (c) STPG; (d) space–time CAU.

Table IV. Semi-discrete (Crank–Nicolson): SUPG.

	CPU time (s)	Non-linear iterations	GMRES iterations
EBE	301.35	605	2680
EBE-BD	231.79	605	2679
EDGE	157.97	605	2679

formulation, results were obtained in both ways: solving the entire system (EDGE1) as shown by Equation (63) or splitting the system in two (EDGE2) as in Equation (66). While in the former the edges are extracted from the wedge-shaped element, in the latter only the edges belonging to the spatial discretization (triangles) are used. These tables also show the total number of non-linear iterations and the total number of GMRES iterations.

7.3. Tidal flow

The last example presents the semi-discrete analysis of a tidal flow in Guanabara bay, situated in Rio de Janeiro. The mesh with 8609 nodal points and 15901 triangular elements (24512 edges), totaling in 23225 equations, is shown in Figure 16. This mesh was generated in such a way that the Courant–Friedrich–Lewy (CFL) number should be approximately constant everywhere. To this end, only the wave propagation velocities were considered. Figure 17 presents a perspective view of the bathymetry. The tidal wave, prescribed at the open boundary, is represented by a sinusoidal wave with amplitude equal to 0.5 m and period of 43200 s. On the closed boundary, both tangential and normal velocities were set to zero.

Table V. Space–time: STPG.

	CPU time (s)	Non-linear iterations	GMRES iterations
EBE	1303.33	600	3773
EBE-BD	1134.32	600	3773
EDGE1	758.43	600	3773
EDGE2	358.28	910	5216

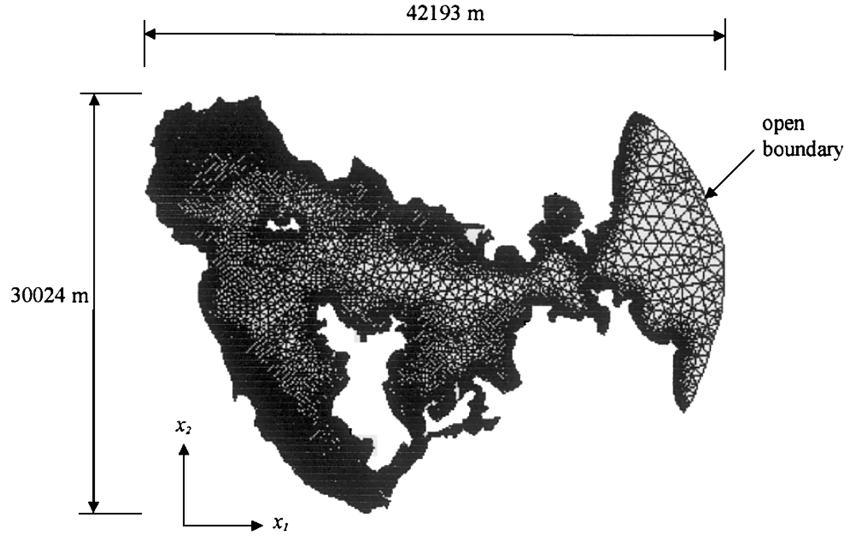


Figure 16. Finite element mesh: 15901 triangular elements and 24512 edges.

Coriolis and wind effects were neglected, the viscosity is $\nu = 2.3 \times 10^{-4}$ m^2 s^{-1} and a constant Chezy coefficient $C = 50.0$ m$^{1/2}$ s^{-1} was assumed. The time step adopted was $\Delta t = 10$ s. The circulation between the two main islands can be seen in Figure 18. Table VI shows the results after 360 time steps, comparing the CPU time using EBE, EBE-BD and EDGE. In this example, as the solution is smooth, presenting no sharp layers, the CAU operator was turned off.

8. CONCLUSIONS

In this paper we have discussed the stabilized space–time and semi-discrete finite element formulations for the SWEs, using Petrov–Galerkin models and discontinuity-capturing operators. Although a symmetric form under velocity/celerity variables has been used, the presented formulations can be straightforwardly applied to the symmetric form derived from entropy

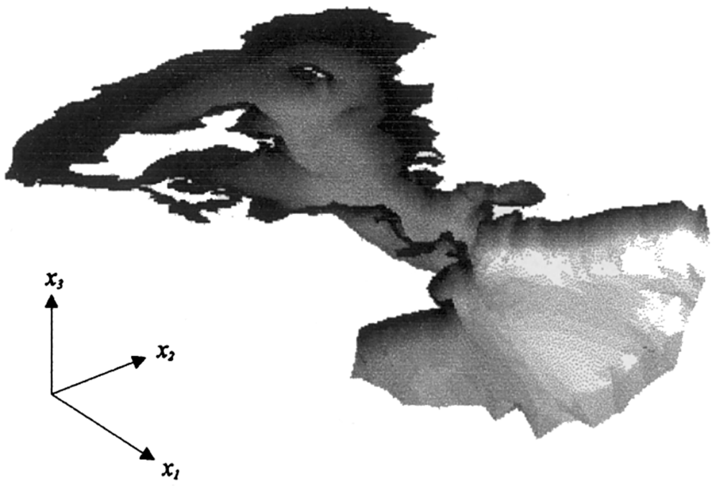


Figure 17. Perspective view of the bathymetry, ranging from 1 m at the closed boundary (in black) to 46 m at the open boundary (in gray).

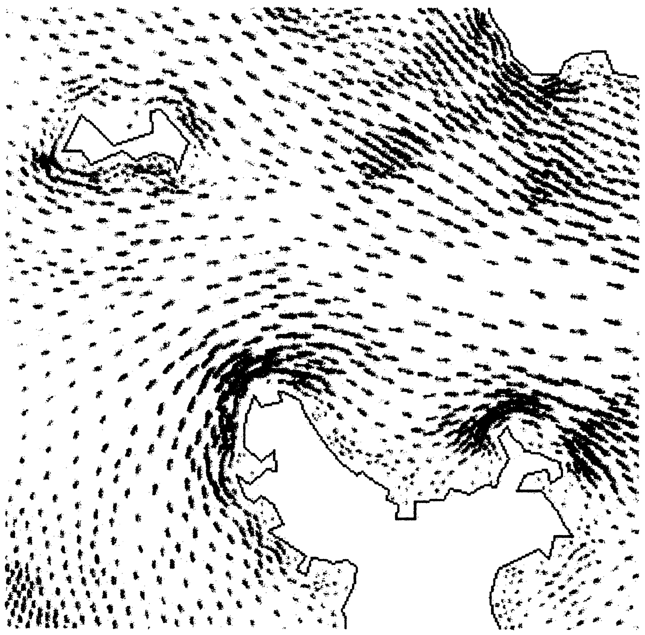


Figure 18. Circulation between the two main islands for time $t = 10800$ s.

Table VI. Semi-discrete (Euler-backward): SUPG.

	CPU time (s)	Non-linear iterations	GMRES iterations
EBE	870.74	1968	9715
EBE-BD	695.78	1968	9715
EDGE	479.09	1968	9715

variables. Fully coupled solutions were obtained using the GMRES solver and an edge-based data structure. Significant improvements, in terms of memory requirements and CPU time were achieved for both versions, space–time and semi-discrete. It was shown that the space–time approximation can be obtained in a very efficient way by splitting the system and using only the edges of the spatial discretization. For the two-dimensional dam break problem, memory and CPU requirements using this procedure were comparable with those of the semi-discrete solution using elements. The results obtained here suggest that the edge-based approach should become a standard for the implementation of iterative solvers.

REFERENCES

1. Bova SW, Carey GF. An entropy variable formulation and applications for the two-dimensional shallow water equations. *International Journal for Numerical Methods in Fluids* 1996; **23**: 29–46.
2. Hauke G, Hughes TJR. A comparative study of different sets of variables for solving compressible and incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 1998; **153**: 1–44.
3. Hauke G. A symmetric formulation for computing transient shallow water flows. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**: 111–122.
4. Harten A. On the symmetric form of systems of conservation laws with entropy. *Journal of Computational Physics* 1983; **41**: 151–164.
5. Hughes TJR, Franca LP, Mallet M. A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier–Stokes equations and the second law of thermodynamics. *Computer Methods in Applied Mechanics and Engineering* 1986; **54**: 223–234.
6. Luo H, Baum J, Löhner R. Edge-based element scheme for the Euler equations. *AIAA Journal* 1994; **32**(6): 1183–1190.
7. Lyra PRM, Manzari MT, Morgan K, Hassam O, Peraire J. Side-based unstructured grid algorithms for compressible viscous flow computations. *International Journal for Engineering Analysis and Design* 1995; **2**: 88–102.
8. Löhner R. Edges, stars, superedges and chains. *Computer Methods in Applied Mechanics and Engineering* 1994; **111**: 255–263.
9. Ribeiro FLB, Galeão AC, Landau L. A space–time finite element formulation for shallow water equations. In *Development and Application of Computer Techniques to Environmental Studies VI*. Computational Mechanics Publications, 1996; 403–414.
10. Ribeiro FLB, Castro RGS, Galeão AC, Loula AFD, Landau L. A space–time finite element formulation for shallow water equations with shock-capturing operator. IV World Congress, Argentina, 1998.
11. Saleri F. Some stabilization techniques in computational fluid dynamics. In *Proceedings of the 9th International Conference on Finite Elements in Fluids*, Venezia, 1995.
12. Shakib F. Finite element analysis of the compressible Euler and Navier–Stokes equations. PhD thesis, Stanford University, 1988.
13. Almeida RC, Galeão AC. An adaptive Petrov–Galerkin formulation for the compressible Euler and Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1996; **129**: 157–176.
14. Galeão AC, Do Carmo EG. A consistent approximate upwind Petrov–Galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering* 1988; **32**: 199–259.

15. Dembo RS, Eisenstat SC, Steinhaug T. Inexact Newton methods. *SIAM Journal of Numerical Analysis* 1982; **19**: 400–408.
16. Eisenstat SC, Walker HF. Globally convergent inexact Newton methods. *SIAM Journal of Optimization* 1994; **4**: 393–422.
17. Papadrakakis M. *Solving Large-scale Problems in Mechanics: The Development and Application of Computational Solution Procedures*. Wiley: Chichester, 1993.
18. Saad Y, Schultz MH. GMRES: generalized minimium residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**: 856–869.
19. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing: Boston, 1996.
20. van Gijzen MB. Large scale finite element computations with GMRES-like methods on a Cray-Y-MP. *Applied Numerical Mathematics* 1995; **19**: 51–62.