Godunov-type solution of the shallow water equations on adaptive unstructured triangular grids

Z. D. SKOULA†*, A. G. L. BORTHWICK‡ and C. I. MOUTZOURIS

†Department of Water Resources, Hydraulic and Maritime Engineering, School of Civil Engineering, National Technical University of Athens, 5 Heroon Polytechniou, Zografou, 15780 Athens, Greece
‡Department of Engineering Science, University of Oxford, Parks Road, Oxford OX1 3PJ, UK

(Received 14 October 2004; revised in final form 14 August 2006)

A Godunov-type upwind finite volume solver of the non-linear shallow water equations is described. The shallow water equations are expressed in a hyperbolic conservation law formulation for application to cases where the bed topography is spatially variable. Inviscid fluxes at cell interfaces are computed using Roe’s approximate Riemann solver. Second-order accurate spatial calculations of the fluxes are achieved by enhancing the polynomial approximation of the gradients of conserved variables within each cell. Numerical oscillations are curbed by means of a non-linear slope limiter. Time integration is second-order accurate and implicit. The numerical model is based on dynamically adaptive unstructured triangular grids. Test cases include an oblique hydraulic jump, jet-forced flow in a flat-bottomed circular reservoir, wind-induced circulation in a circular basin of non-uniform bed topography and the collapse of a circular dam. The model is found to give accurate results in comparison with published analytical and alternative numerical solutions. Dynamic grid adaptation and the use of a second-order implicit time integration scheme are found to enhance the computational efficiency of the model.

Keywords: Shallow water equations; Godunov; Adaptive finite volume; Variable bed topography; Unstructured grids; Implicit time integration

1. Introduction

The non-linear shallow water equations are commonly used to represent almost-horizontal free surface flows, which occur in wide rivers, estuaries, shallow lakes and coastal waters. Whereas many numerical models aim at simulating smoothly varying flows, an area of considerable recent research interest has been the modelling of strongly varying or even discontinuous flows where the gradients of the dependent variables may be undefined and conservative integral methods are required. To this end, a popular approach in addressing these problems has been the solution of the non-linear shallow water equations in hyperbolic conservation form using Godunov-type finite volume schemes (see e.g. Alcrudo and García-Navarro (1993), Zhao et al. (1994), Fraccarollo and Toro (1995), Anastasiou and Chan (1997), Mingham and Causon (1998), Vázquez-Cendón (1999), Zoppou and Roberts (2000), Toro (2001), Zhou et al. (2002)). Godunov-type models deal with wave discontinuities in hyperbolic conservation systems by assuming a piece-wise constant distribution of the conserved variables within computational cells. However, the closely associated Riemann problem is computationally demanding to solve exactly, and so many approximate Riemann solvers have been proposed. Toro (1997) and LeVeque (2002) give comprehensive reviews of Riemann solvers and total variation diminishing schemes that achieve second-order accuracy and limit artificial oscillations. It should be noted that the choice of an approximate Riemann solver can affect the applicability of the scheme. For example, Toro (2001) shows that Roe’s approximate Riemann solver is inappropriate for dry bed dam breaks, and recommends using the HLLE solver in such cases.

Having established a robust way of calculating the spatial evolution of inviscid numerical fluxes, the
parameters that may further enhance numerical accuracy have to be considered. These include the implicit or explicit character of the time integration approach utilised (implicit time integration schemes are more stable than explicit ones at the expense of greater program complexity) and the overall size of the mesh resolution (which has to be proportional to the size of the finest flow phenomenon encountered in the computational field). The optimal balance between numerical accuracy and computational efficiency seems to lie with dynamically adaptive grids whereby local mesh density is altered based on individual cell or patch-based properties (see e.g. George (1991) and Thompson et al. (1999) for detailed reviews). These properties-criteria fall into two main categories: numerical error estimation (see e.g. Strouboulis and Haque (1992)); and rules based on the local values of certain physical flow variables (recommended by George (1991)). In this context, Skoula (2002) investigated refinement approaches that include subdivisions of elements based on certain limits of components such as vorticity, velocity gradients, the Courant number or the Reynolds number and highlighted the appealing approach of refinement criteria estimation based on a fast computational method and the consequent refinement according to established case-specific rules. The refinement may take place through certain approaches, which are classified as h-enrichment, p-polynomial and r-relocation ones. The h-enrichment version of the method entails locating mesh cells based on a posteriori or a priori criteria related to the computed flow regime while in p-polynomial schemes selective or global increase of the spatial order of the numerical fluxes is utilised on a globally-coarse spatial discretisation. Moreover, in r-relocation schemes a fixed number of cells is dynamically relocated according to certain optimisation criteria. It is noted that p- schemes are likely to become inaccurate when local flow discontinuities are encountered while r- schemes are not widely applicable since there is no guarantee of an improvement in accuracy. In practice, h-enrichment (or a combination of h- with p- or r-) is the method widely selected for reducing spatial truncation errors while enhancing the performance of the numerical solver. References to adaptive applications in the context of Godunov-type solvers of the shallow water equations include papers by Sleigh et al. (1998), Hubbard and Dodd (2002) and Rogers et al. (2003).

This paper describes a second-order implicit Godunov-type solver of the non-linear shallow water equations that is capable of accounting for non-uniform bed topographies. The model is based on dynamically adaptive triangular grids that are enriched locally according to physical flow properties’ criteria. Results are presented for standard benchmark tests involving rapidly changing flow fronts, recirculating shear flows, wind-induced topographic gyres and the collapse of a circular dam.

2. The non-linear shallow water equations

The integral matrix form of the non-linear shallow water equations can be written, after a conservation law formulation that includes a source term, as

\[
\frac{\partial}{\partial t} \int_{\Omega} \mathbf{U} \, d\Omega + \int_{S} \left( \mathbf{F}^I - v_{f} \mathbf{F}^V \right) dS = \int_{\Omega} \mathbf{H} \, d\Omega
\]

(1)

where \( \mathbf{U} \) is the vector of conserved variables, \( \mathbf{F} \) is the vector of flux functions through \( S \) (that consists of inviscid \( \mathbf{F}^I \) and viscous \( \mathbf{F}^V \) flux component vectors), \( \mathbf{H} \) is the vector of source terms, \( v_{f} \) is the kinematic eddy viscosity coefficient, \( \Omega \) is the domain of interest and \( S \) is the boundary surrounding \( \Omega \). The vectors are composed as follows:

\[
\mathbf{U} = \begin{bmatrix} \xi \\ uh \\ vh \end{bmatrix}, \quad \mathbf{F}^V = \begin{bmatrix} 0 \\ h_{u_{n} n_{c} + h_{u_{v} n_{c}} } \\ h_{v_{n} n_{c} + h_{v_{n} n_{c}} } \end{bmatrix},
\]

\[
\mathbf{F}^I = \begin{bmatrix} uhn_{x} + vhn_{y} \\ u^{2} h + \frac{g}{2} \left( \xi^{2} + 2h_{\xi} \right) n_{c} + uvh_{n_{y}} \\ uvh_{n_{x}} + \left[ v^{2} h + \frac{g}{2} \left( \xi^{2} + 2h_{\xi} \right) n_{y} \right] \end{bmatrix},
\]

\[
\mathbf{H} = \begin{bmatrix} 0 \\ \frac{\tau_{ux}}{\rho} - \frac{g}{C_{f}} \sqrt{u^{2} + v^{2}} - g\xi S_{ox} + hv_{u} \\ \frac{\tau_{uy}}{\rho} - \frac{g}{C_{f}} \sqrt{u^{2} + v^{2}} - g\xi S_{oy} - hv_{v} \end{bmatrix},
\]

where \( g \) is the acceleration due to gravity, \( h \) is the total depth (\( h = h_{s} + \xi \)), \( \xi \) is the free surface elevation above still water level, \( h_{s} \) is the still water depth, \( S_{ox} \) and \( S_{oy} \) are the bed slopes in the x- and y-directions respectively, \( u \) and \( v \) are depth averaged velocities in the x- and y-directions respectively, \( \rho \) is water density, \( \tau_{ux} \) and \( \tau_{uy} \) are surface stresses in the x- and y-directions respectively, \( C \) is the Chézy coefficient and \( f \) is the Coriolis parameter. The velocity subscripts denote derivatives in the implied directions. Moreover, \( n_{x} \) and \( n_{y} \) denote the x and y components of the outward normal vector \( \mathbf{n} \). The bed shear stress terms are evaluated by the following equations

\[
\tau_{ux} = \rho C_{f} u \sqrt{u^{2} + v^{2}} \frac{1}{2}, \quad \tau_{uy} = \rho C_{f} v \sqrt{u^{2} + v^{2}} \frac{1}{2}
\]

where \( C_{f} \) is an empirical friction coefficient based on bed roughness. The expression used herein for calculating \( C_{f} \) is the Chézy friction law, \( C_{f} = g / C^{2} \), where \( C \) is the Chézy coefficient.

It is worth noting that vectors \( \mathbf{U}, \mathbf{F} \) and \( \mathbf{H} \) have the content described in equation (2) due to the fact that the involved free surface gradient terms are split in the
following way:
\[
gh \frac{\partial \zeta}{\partial x} = g \zeta S_{ax} + \frac{1}{2} gh \frac{\partial (\zeta^2 + 2 \zeta h_i)}{\partial x},
\]
\[
gh \frac{\partial \zeta}{\partial y} = g \zeta S_{ay} + \frac{1}{2} gh \frac{\partial (\zeta^2 + 2 \zeta h_i)}{\partial y}.
\]

This splitting and the consequent partial redistribution between flux gradient and source terms is based on the fact that the splitting of terms in Godunov-type models that use Roe’s approximate Riemann solver may cause numerical inaccuracy in cases where the bed is non-horizontal (see e.g. Nugic (1995)) while techniques for handling the inaccuracy in cases where the bed is non-horizontal (see e.g. Chippada et al. (1998), and Rogers et al. (2003)). The splitting of terms adopted herein not only provides a hyperbolic nature to the system of equations under consideration (given that a prerequisite for adopting the method of characteristics as a solution technique of a certain system of equations, is the hyperbolic nature of the system under consideration) but at the same time it subtracts the still water level from the column of water that is involved in the dynamic character of the test case under consideration. It is noted that other ways of splitting these terms are available in literature, see for example Anastasiou and Chan (1997), but they are unsuitable for non-uniform bed topographies, which is not the case for the adopted herein formulation, see Rogers et al. (2003).

3. The numerical solver

Values of the conserved variables in the present scheme are cell-centred. The three edges of each cell define a triangular control volume over which the integral form of the equations transforms into

\[
\frac{\partial \mathbf{U}_i \Omega_i}{\partial t} = - \oint_{\partial C_i} \mathbf{F} \cdot \mathbf{n} \, dS + \mathbf{H}_i \Omega_i
\]

\[
= \sum_{j = k(i)} F_{i,j} \Delta l_{i,j} + \mathbf{H}_i \Omega_i = - \text{RHS}(\mathbf{U}_i)
\]

(4)

where \( \mathbf{U}_i \) and \( \mathbf{H}_i \) are the average quantities of cell \( i \) stored at the cell centre, while \( \Omega_i \) and \( \partial C_i \) denote the area and boundary of cell \( i \), respectively. In the discrete form of the integral, \( k(i) \) is a list of the neighbouring cells to cell \( i \), \( F_{i,j} \) is the numerical flux through the interface of cells \( i \) and \( j \), and \( \Delta l_{i,j} \) is the length of the \( i,j \) edge, respectively.

3.1 Evaluation of inviscid fluxes

Roe’s approximate Riemann solver is used to evaluate the inviscid numerical fluxes from:

\[
F_{i,j} = \frac{1}{2} \left[ F^1(\mathbf{U}^+_{i,j}) + F^1(\mathbf{U}^-_{i,j}) - |\mathbf{A}|(\mathbf{U}^+_{i,j} - \mathbf{U}^-_{i,j}) \right] \quad (5)
\]

where \( |\mathbf{A}| = \mathbf{R} \cdot |\mathbf{A}| \cdot \mathbf{L} \) is the magnitude of the inviscid flux Jacobian matrix and \( \mathbf{U}^+_{i,j} \) and \( \mathbf{U}^-_{i,j} \) are the right and left states respectively of the interface between cells \( i \) and \( j \). The matrices \( \mathbf{R} \) and \( \mathbf{L} \) are the right and left eigenvector matrices of flux Jacobian \( \mathbf{A} = \frac{\partial F^1}{\partial \mathbf{U}} \), and \( |\mathbf{A}| \) = diag\([\lambda_1, \lambda_2, \lambda_3]\) is a diagonal matrix of the absolute values of the eigenvalues of \( \mathbf{A} \). The variables \( \zeta, u \) and \( v \) involved in the calculation of the inviscid fluxes are described using Roe’s average state, which is defined as:

\[
\zeta = \frac{(h_x + \zeta^+ + h_x + \zeta^-)}{2} - h_v,
\]

\[
u = \frac{u^+ \sqrt{(h_x + \zeta^+)^2 + u^- \sqrt{(h_x + \zeta^-)^2}}}{(h_x + \zeta^+)^2 + (h_x + \zeta^-)^2}
\]

(6)

Matrices \( \mathbf{A}, \mathbf{R} \) and \( \mathbf{L} \) used herein render identical to those of Anastasiou and Chan (1997) despite the different formulation of equations (2) used herein. Hirsch (1990) provides detailed information about the evaluation of eigenvalues and eigenvectors.

3.1.1 Second order spatial accuracy and control of local oscillations

An essential part of the numerical process is the determination of the left and right Riemann states at each interface. For a first-order space accurate Godunov-type scheme, the assumption is that values of the conserved variables within each cell are constant throughout the cell. That is \( \mathbf{U}_i = 0 \) where \( i \) is the cell under consideration. This means that for all edges, \( j \), of cell \( i \), \( \mathbf{U}^-_{i,j} = \mathbf{U}_i \) where \( \mathbf{U}_i \) is the vector of conservative variables at the centre of cell \( i \). To achieve second-order accuracy in space, a piecewise linear variation of the conserved variables vector is assumed within each cell, following the MUSCL or variable extrapolation approach. Hence, for a given cell with centre \( C \) the interior Riemann states are constructed according to the relation

\[
\mathbf{U}(x,y) = \mathbf{U}_C + \Phi \nabla \mathbf{U}_C \cdot \mathbf{r}
\]

(7)

where \( \mathbf{r} \) is the vector from cell centre \( C \) to any point \( (x, y) \) within the cell, \( \mathbf{U}_C \) is the vector of conserved variable cell-centre values, and \( \nabla \mathbf{U}_C \) is its vector gradient whose accuracy can be further improved by carrying out a weighted averaging process, see Pan and Cheng (1993) for the one adopted herein. Moreover, \( \Phi \) is a limiter used to control numerical oscillations at flow discontinuities. It is noted that this approach is second-order accurate for all non-zero values of \( \Phi \), while it is first-order accurate for zero values of \( \Phi \). The limiter used herein is:

\[
\Phi = \min(\max(\beta \mathbf{r}_j, 1), \min(r_j, \beta)),
\]

\[
j = 1, 2, 3
\]

(8)
with the variable $\beta$ taking values between 1 and 2 (in particular when $\beta = 1$, $\Phi$ becomes the “minmod” limiter, whilst when $\beta = 2$, $\Phi$ corresponds to Roe’s “superbee” limiter), and $r_i$ being evaluated according to Pan and Cheng (1993).

### 3.1.2 Spatial gradients calculation of conserved and source term variables
The spatial gradients of variables are calculated using a heuristic method based on the unit normal vector relative to the plane that is composed by the lines connecting the centres of the three neighbouring cells to the cell of interest. The method is outlined as follows: consider an arbitrary variable $B$, whose gradients are to be calculated in two-dimensions. The projection of the unit normal vector of $B$ yields the following Cartesian components of the gradient of $B$:

$$
\frac{\partial B}{\partial x} = -\frac{B_{cx}}{B_{cz}}, \quad \frac{\partial B}{\partial y} = -\frac{B_{cy}}{B_{cz}}
$$

(9)

in which $B_{cx}$, $B_{cy}$ and $B_{cz}$ are the local Cartesian components of $B$. This approach gave very accurate results for a number of simple test cases with known values of gradients. In present study it is used to calculate spatial gradients of components of the source term $H$ (i.e.: bathymetry) and to enhance the polynomial approximation of the conserved variables through equation (7).

### 3.2 Evaluation of viscous fluxes
The viscous fluxes $F_{ij}$ are evaluated at cell faces. The velocity gradients are determined from weighted area averaging (see figure 1):

$$
\nabla U_{\Gamma\Delta} = \frac{1}{\Omega_{\Gamma\Delta}} ((\Omega_{\Gamma BA} U_{A\Gamma\Delta} + \Omega_{A\Gamma\Delta} U_{BA\Gamma})
$$

(10)

where $\nabla U_{A\Gamma\Delta}$ and $\nabla U_{BA\Gamma}$ are the gradient vectors of paths $A \Gamma \Delta$ and $B \Gamma \Delta$, respectively, and $\Omega$ denotes area. For a boundary cell (e.g. $\Gamma \Delta E$), equation (10) is restricted to $\nabla U_{\Gamma\Delta} = \nabla U_{A\Gamma\Delta}$. The values at cell vertices (i.e. $\Gamma$ and $\Delta$) are calculated by a distance-weighted average of cell-centre values surrounding the vertex under consideration.

![Figure 1. Integration path for viscous stresses: (a) $A\Gamma\Delta$ for an internal cell, (b) $A\Gamma\Delta$ for a boundary cell.](image)

### 3.3 Initial and boundary conditions
Initial conditions are user-specified. Wall boundaries are either slip or no-slip. For a slip condition, $U_i^b = 0$, $i = 2$, $\partial U_i^b / \partial x_i = 0$, $i \neq 2$, where subscript $i$ refers to the three unknowns, namely $\xi$, $u(h_x + \xi)$, and $\phi(h_x + \xi)$. For a no-slip wall, $U_i^b = 0$, $i \neq 1$, $\partial U_i^b / \partial x_i = 0$, $i = 1$. At open boundaries, the Froude number $Fr = U/(gh)^{1/2}$ is used to determine whether the flow is subcritical or supercritical. For supercritical flows, $U_i$ is specified. For subcritical flows or where a radiation flux condition is applied $\partial U_i / \partial x_i = 0$. At a velocity boundary, if the flow is supercritical, then $\partial U_i / \partial x_i = 0$. Otherwise $h_b = (0.5 g^{-1/2}(U_{in} - U_b) + h_{in}^{1/2})^2$. At a water surface elevation boundary, if the flow is supercritical then $\partial U_i / \partial x_i = 0$, else $U_b = U_{in} + 2 g^{1/2}(h_{b2}^{1/2} - h_{b1}^{1/2})$. It should be noted that $U$ and $V$ refer to the depth averaged velocity components normal to each cell edge in the Cartesian system of coordinates, subscript $b$ refers to boundary values, while subscript $in$ refers to values in the internal flow field.

### 3.4 Time integration
Equation (4) is updated in time as follows:

$$
\frac{(\Omega U)^{n+1} - (\Omega U)^n}{\Delta t} = -[\alpha_t \text{RHS}(U_i^{n+1}) + (1 - \alpha_t) \text{RHS}(U_i^n)]
$$

(11)

where $U_i^{n+1}$ is the vector of conserved variables for cell $i$ at time level $n + 1$, $U_i^n$ is the known state at time level $n$, $\Delta t$ is the time step, $\Omega_i$ is the area of cell $i$ (not constant with respect to time for an adaptive scheme) and $\text{RHS}(U_i^n)$ is the right hand side of equation (4) evaluated as described previously. Constant $\alpha_t$ is user-specified and sets the character of the calculations that will take place in each numerical experiment (e.g. $\alpha_t = 0.0$ corresponds to first-order accurate explicit time calculations with $U_i^{n+1}$ being defined by $U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x} \text{RHS}(U_i^n)$, $\alpha_t = 1.0$ corresponds to first-order accurate implicit time calculations and $\alpha_t = 0.5$ corresponds to second-order accurate implicit time calculations). The time step $\Delta t$ is determined from

$$
\Delta t = \frac{1}{N} \sum_{i=1}^{N} \left\{ \min_{j=0} k(i)(C_i \left\{ \frac{S_{ij}}{2(c_i + c_{ij})} \right\} \right\}
$$

(12)

where $N$ is the total number of cells in the domain, $k(i)$ is a list of the three adjacent cells to cell $i$, $C_i$ is a user specified Courant-like number, $S_{ij}$ is the distance between the centres of the adjacent cells $i$ and $j$, $c_i$ is the wave celerity at cell $i$ which is equal to $(gh)^{1/2}$, and $c_{ij} = (u_i^2 + v_i^2)^{1/2}$. The time step therefore depends on the underlying mesh density and the flow regime, and is recalculated.
throughout the simulation immediately after the depth and velocity components are updated.

For implicit time integration ($\alpha_t \neq 0.0$), Newton’s linearisation method is used to solve iteratively equation (11), following Anastasiou and Chan (1997). The method is based on Taylor’s expansion theorem up to first-order for $N$ variables. Hence, the exact value of the unknowns in the particular system of equations can be approximated by a known value plus an error quantity, $\epsilon$. The linear (with respect to $\epsilon$) equations are then solved simultaneously for $\epsilon$. The approximated value is corrected and the process is repeated with the corrected value as the new approximation. The method states that a general function $f(X_1, X_2, X_3, \ldots) = 0$ can be expressed as:

$$f(x_1, x_2, x_3 \ldots) + \begin{bmatrix} \frac{\partial f}{\partial x_1} \epsilon_1 + \frac{\partial f}{\partial x_2} \epsilon_2 + \frac{\partial f}{\partial x_3} \epsilon_3 + \ldots \end{bmatrix} = 0 \quad (13)$$

where $X_k$ are the unknown variables, $x_k$ are known approximations to $X_k$ and $\epsilon_k$ are the corrections to $x_k$. The subscript notation after the right square bracket in equation (13) indicates that the unknown variables $X_k$ are replaced by the vector $x_k$ after the differentiation. The implicit formulation is now derived for a general triangular cell, 0. Following this notation, equation (11) gives

$$f = \frac{\Omega_0}{\Delta t} (U_{0}^{n+1} - U_0^n) + \alpha \text{RHS}(U_0^{n+1}) + (1 - \alpha) \text{RHS}(U_0^n) = 0 \quad (14)$$

where $U_0^n$ is the known state for cell 0 at the previous time step, whilst equations (4) and (5) are further decomposed in order to facilitate the application of Newton’s linearisation

$$\text{RHS}(U_0^{n+1}) = 0.5 \left[ F(U_0^{n+1}) + F(U_0^n) - [A_{0,1} (U_1^{n+1} - U_0^n)] \Delta U_{0,1} + 0.5 \left[ F(U_2^{n+1}) + F(U_1^n) - [A_{0,2} (U_2^{n+1} - U_0^n)] \Delta U_{0,2} + 0.5 \left[ F(U_3^{n+1}) + F(U_2^n) - [A_{0,3} (U_3^{n+1} - U_0^n)] \Delta U_{0,3} - H_0 \Omega_0 \right] \right] \right.$$

$$\left. \right.$$

$$\text{where 0 denotes the particular cell and 1, 2, 3 denote its adjacent cells and subscripts i, j refer to the interface between cells i and j. } \text{H}_0 \text{ is defined in equation (2).}

Let $U_0^{n+1}$ be approximated by a known state $U_0^n$ plus an error quantity $\Delta U_0^n$, i.e. $U_0^{n+1} = U_0^n + \Delta U_0^n$, where $s$ is a sub-iteration index. Newton’s linearisation of equation (14) then yields

$$\begin{align*}
[\Omega_0 & \Delta t \left( U_0^s - U_0^n \right) + \alpha \text{RHS}(U_0^s) + (1 - \alpha) \text{RHS}(U_0^n) \\
& + \left[ \frac{\partial f}{\partial U_0^{n+1}} \right]_{U_0^{n+1} = U_0^n} \Delta U_0^n + \left[ \frac{\partial f}{\partial U_1^{n+1}} \right]_{U_1^{n+1} = U_1^n} \Delta U_1^n + \left[ \frac{\partial f}{\partial U_2^{n+1}} \right]_{U_2^{n+1} = U_2^n} \Delta U_2^n + \left[ \frac{\partial f}{\partial U_3^{n+1}} \right]_{U_3^{n+1} = U_3^n} \Delta U_3^n = 0 \\
\end{align*} \quad (16)$$

Recalling that $J = \partial F/\partial U$ is the flux Jacobian, equation (16) gives

$$\begin{align*}
- \left[ \frac{\Omega_0}{\Delta t} (U_0^s - U_0^n) + & \alpha \text{RHS}(U_0^s) + (1 - \alpha) \text{RHS}(U_0^n) \\
& + \left[ \frac{\partial f}{\partial U_0^{n+1}} \right]_{U_0^{n+1} = U_0^n} \Delta U_0^n + \left[ \frac{\partial f}{\partial U_1^{n+1}} \right]_{U_1^{n+1} = U_1^n} \Delta U_1^n + \left[ \frac{\partial f}{\partial U_2^{n+1}} \right]_{U_2^{n+1} = U_2^n} \Delta U_2^n + \left[ \frac{\partial f}{\partial U_3^{n+1}} \right]_{U_3^{n+1} = U_3^n} \Delta U_3^n \right] \\
& \left. \right]_{U_0^{n+1} = U_0^n} \Delta U_0^n \\
& \left. \right]_{U_1^{n+1} = U_1^n} \Delta U_1^n \\
& \left. \right]_{U_2^{n+1} = U_2^n} \Delta U_2^n \\
& \left. \right]_{U_3^{n+1} = U_3^n} \Delta U_3^n = 0 \\
\end{align*} \quad (17)$$

Equation (16) demonstrates clearly the cell-coupling effect of the implicit scheme. Application of equation (17) to all cells in the domain results in a matrix equation of the type $AX = B$ where $A$ is a $N \times N$ matrix, with only four non-zero and not necessarily consecutive entries per row, and $X, B$ are $N \times 1$ matrices ($N$, the number of cells in the domain). Each of the elements in $A, X$ and $B$ is a $3 \times 3$ matrix. Direct inversion of $A$ is computationally expensive, and so the fully discretised implicit time integration equation is written as $(D_{\text{diag}} + L_{\text{low}} + U_{\text{up}}) \Delta U = \text{RHS}$, where $D_{\text{diag}}$ is a block diagonal matrix, $L_{\text{low}}$ is a block lower triangular matrix, and $U_{\text{up}}$ is a block upper tridiagonal matrix, with each of the elements in these matrices being a $3 \times 3$ matrix. An approximate LU factorisation (ALU) scheme after Pan and Lomax (1988) is then adopted to convert the discretised implicit time integration equation into the form $(D_{\text{diag}} + L_{\text{low}}) D_{\text{diag}}^{-1}$.
(D_{\text{diag}} + U_{\text{up}}) \Delta U^t = \text{RHS} \text{ where the factorisation error is } L_{\text{low}} D_{\text{diag}}^{-1} U_{\text{up}} \Delta U^t. \text{ } \Delta U^t \text{, may now be obtained in a straightforward manner. Sub-iterations within each time step are commenced by setting } U^t_{\text{old}} = U^t_{\text{new}} \text{ and terminated when the } L2 \text{-norm of the iteration process,}

\begin{equation}
L2 = \left[ \frac{1}{N} \left( \sum_{i=1}^{N} (\Delta U_i^t)^2 \right) \right]^{1/2}
\end{equation}

is less than a specified tolerance value for all variables } u \text{ of matrix } U. \text{ As } \Delta U^t \text{ tends to zero, } U^t_{\text{old}} \text{ tends to the accurate } U^t_{\text{new}} \text{ as the ALU factorisation error tends to zero. The number of sub-iterations required per time step is internally decided and has to be small in order for a simulation to be efficient, while the time step value for the implicit scheme can be one or two orders of magnitude larger than the feasible time step for explicit time integration.}

3.5 The use of dynamically-adaptive triangular grids

Dynamically-adaptive unstructured grids are used to resolve flow features at different scales. The initial grid consists of triangular cells that have been generated using an advancing front algorithm on the plan-form shallow flow domain. At fixed intervals, values of certain computed flow variables are compared against adaptation criteria for each cell and its neighbours in turn. If the adaptation criteria are satisfied, then mesh enrichment is implemented by cell subdivision into further triangles. The overall procedure is:

- Initially a coarse grid } T_0 \text{ is constructed representing the shallow flow geometry.}
- The index } k \text{ is set to zero, with } k \text{ denoting the number of times that } T_k \text{ has been altered.}
- The discrete problem is solved on } T_k.
- For each cell in } T_k \text{ the adaptation criterion is computed. If the criterion does not hold for a cell, then it is left unaltered and the next cell is examined. If the criterion does hold, then mesh enrichment takes place.}
- After all cells have been checked, the new adapted mesh } T_{k+1} \text{ has been constructed.}
- } k \text{ is then replaced by } k + 1 \text{ and the procedure returns to the third step above.}

Mesh enrichment is implemented as follows. For any cell that satisfies the adaptation criterion, the neighbouring cells are also checked as to whether they too satisfy the criterion. If the criterion holds for all the neighbours as well as the cell of interest, then } h\text{-refinement is applied to it. If the adaptation criterion does not hold for a cell but holds for two of its neighbours, then the cell is } h\text{-refined. If the adaptation criterion holds only for one neighbouring cell but not for the cell itself, then half-refinement is implemented on the cell. This is achieved by connecting the new node inserted at the interface between the cell and its neighbour cell that satisfies the criterion (and so is } h\text{-refined) with the opposite vertex of the cell of interest. The } h\text{-refinement adopted herein (see Bank (1983)) involves creating more cells by connecting lines between mid-points of adjacent sides of a triangular cell. Hence, a parent cell is subdivided into four triangular cells at the first level of refinement. The present scheme permits up to two levels of local refinement beyond the base grid under consideration. Figure 2 depicts a refinement example where the refinement criterion holds for triangles } B\Gamma\Theta \text{ and } \Gamma Z\Theta \text{ but does not hold for the other triangles } (A\Theta B, \Delta \Theta\Gamma, \Gamma \Delta E, \Gamma E Z, Z \Theta\Theta). \text{ The mesh enrichment approach used here is designed to achieve fairly uniform integration properties over the computational domain while avoiding numerical errors related to the presence of hanging nodes.}

4. Results

4.1 Oblique hydraulic jump

The inviscid formulation of the present solver is first validated for the benchmark case of an oblique hydraulic jump for which an analytical solution is available, see Hager et al. (1994). This test case has also been chosen as the basis for testing the effect that the mesh density has on numerical results and to highlight the advantages to be gained by using mesh adaptation. Various combinations of time integration approaches and different values of the tolerance parameter } L2 \text{ are considered in order to investigate the effect of implicit time integration on the accuracy and efficiency of the numerical scheme. The case involves a converging wall deflecting a supercritical inflow to create an oblique hydraulic jump. The upstream water depth is } 1 \text{ m and inflow velocity is } 8.57 \text{ m/s corresponding to an inflow Froude number of } 2.73619. \text{ The side wall convergence is at an angle, } \theta = 8.95^\circ. \text{ At the wall convergence, a shock-like oblique hydraulic jump forms at an angle } \beta_1 \text{ that is different to that of the wall } \theta. \text{ Hager et al.’s (1994) analytical approach predicts the jump angle } (\beta_1 = 30^\circ), \text{ the downstream water depth } (1.49984 \text{ m}), \text{ and the downstream velocity } (7.95308 \text{ m/s}). \text{ In the numerical model, the inflow boundary condition is set to supercritical and no-slip wall conditions are applied.}

Figure 2. (a) Initial triangulated domain; (b) refinement when adaptation criterion holds for triangles } B\Gamma\Theta \text{ and } \Theta\Gamma Z\Theta \text{ only.
Adaptive implicit solution

Figure 3. Computational grids and steady-state free surface contours for oblique hydraulic jump: (a) coarse grid 1,424 cells; (b) medium grid 5,696 cells; (c) dynamically adapted grid; and (d) fine grid 91,136 cells.
at the walls. In all cases examined (see tables 1–3), a “minmod” slope limiter is used. The initial conditions are $z_0 = 0.00$ m, $u_0 = 8.57$ m/s and $v_0 = 0.00$ m/s.

Flow simulations are undertaken using a fixed coarse grid of 1,424 cells, a fixed medium grid of 5,696 cells, a fixed fine grid of 91,136 cells and a dynamically adaptive grid. Figure 3 shows the fixed and evolved grids and steady-state free surface elevation contours obtained. The adaptive grid results are obtained based on an initial grid identical to the medium grid depicted in figure 3(b), while the number of cells at steady-state reached 16,078. The grid adaptation criterion is based on the magnitude of the free surface gradient

$$\xi = \left( \frac{\partial h}{\partial x} \right)^2 + \left( \frac{\partial h}{\partial y} \right)^2 \right)^{1/2} \tag{20}$$

with mesh enrichment being invoked for values of $\xi > 0.01$.

Table 1 presents results obtained on the fixed medium grid using three different time integration schemes, corresponding to: $a_t = 0.0$, first-order explicit; $a_t = 1.0$, first-order implicit; and $a_t = 0.5$, second-order implicit. For the implicit schemes, the Courant number was set to 2 and the $L_2$ tolerance to 1e-2. For the explicit scheme the Courant number was set equal to unity. All schemes give predictions that are almost identical, and in reasonable agreement with the analytical solution, though the implicit schemes use a time step that is about a factor of ten larger than that required by the explicit scheme. On the other hand, the CPU time that the implicit schemes need is bigger than that of the explicit one, due to the iteration procedures involved.

Table 2 shows the results obtained on the fixed and dynamically adaptive grids using the second-order implicit scheme. As would be expected, the denser the fixed grid the greater the accuracy of the numerical predictions and the computational effort required. Furthermore, the predictions based on the adaptive grid are of the same order of accuracy as those on the fine grid, even though the number of cells required is more than five times smaller. It is noted that in constructing the velocity-related results, it has been assumed that the free surface elevation data had to be over 1.48 m as the width of the free surface discontinuity varied according to the underlying mesh density under consideration, see figures 3 and 4. This variation also indicated that the results’ quality could be measured by the percentage of the computational domain involved in the jump. Hence, the plan area occupied by the jump is taken to be that occupied between the water surface elevation of 1.48 m and its maximum value. In table 2, the jump area is expressed as a percentage of the domain area, and can be used as a further measure of the quality of the results. Excellent agreement is achieved between the fine grid prediction and the analytical value of the percentage jump area.

Table 3 investigates the model sensitivity to the choice of the tolerance $L_2$ on the coarse fixed grid using the second-order implicit time-stepping scheme. The effect of reducing the tolerance $L_2$ by three orders of magnitude appears to be limited in terms of accuracy, but requires a factor of two increase in CPU time. Further to the data included in table 3, it is noted that the initial subiterations number within each time step for $L_2 = 1e-3$ is 7 at the beginning of the calculations while at the end of them the number drops to zero. Moreover, for this test case the CPU time becomes 551 s while for all smaller than 1e-5 values of $L_2$ tested, the initial required number of subiterations was prohibitively expensive for the solution to proceed.

Table 1. Analytical and numerical results: different time integrators, oblique hydraulic jump.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Analytical solution</th>
<th>$a_t = 0.0$</th>
<th>$a_t = 1.0$</th>
<th>$a_t = 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$ (m)</td>
<td>1.49984</td>
<td>1.498392</td>
<td>1.498392</td>
<td>1.498392</td>
</tr>
<tr>
<td>$\tau$ (m/s)</td>
<td>7.95308</td>
<td>7.884612</td>
<td>7.884614</td>
<td>7.884614</td>
</tr>
<tr>
<td>Froude No</td>
<td>2.07338</td>
<td>2.056531</td>
<td>2.056531</td>
<td>2.056531</td>
</tr>
<tr>
<td>$\Delta t$ (s)</td>
<td>–</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>–</td>
<td>1650</td>
<td>1650</td>
<td>1650</td>
</tr>
<tr>
<td>Jump area (%)</td>
<td>16.73</td>
<td>10.81</td>
<td>14.74</td>
<td>16.72</td>
</tr>
</tbody>
</table>

Table 2. Analytical and numerical results: grid sensitivity, oblique hydraulic jump.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Analytical solution</th>
<th>Fixed coarse grid</th>
<th>Fixed medium grid</th>
<th>Fixed fine grid</th>
<th>Final adapted grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$ (m)</td>
<td>1.49984</td>
<td>1.495828</td>
<td>1.498392</td>
<td>1.498687</td>
<td>1.498687</td>
</tr>
<tr>
<td>$\tau$ (m/s)</td>
<td>7.95308</td>
<td>7.804279</td>
<td>7.884612</td>
<td>7.941130</td>
<td>7.941130</td>
</tr>
<tr>
<td>Froude No</td>
<td>2.07338</td>
<td>2.037343</td>
<td>2.056531</td>
<td>2.071061</td>
<td>2.071061</td>
</tr>
<tr>
<td>$\Delta t$ (s)</td>
<td>–</td>
<td>0.17</td>
<td>0.04</td>
<td>0.015</td>
<td>0.015</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>–</td>
<td>441</td>
<td>1650</td>
<td>54764</td>
<td>6220</td>
</tr>
<tr>
<td>Jump area (%)</td>
<td>16.73</td>
<td>10.81</td>
<td>14.74</td>
<td>16.72</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
4.2 Jet-forced flow in a flat-bottomed circular reservoir

The test case of a jet-forced flow in a flat-bottomed circular reservoir has been selected to demonstrate further the dynamic adaptive capability of the numerical scheme and to verify that it can correctly simulate separated recirculating viscous flows. The circular reservoir has radius 0.75 m and has inlet and outlet stems diametrically opposite to each other. The stems are of width $b = 0.156$ m and length $l = 0.75$ m. The mean inflow velocity $U_b = 0.1$ m/s and the eddy viscosity coefficient is $0.00078$ m$^2$/s, corresponding to an inlet Reynolds number ($Re = 0.5Ub/b$) of 10. The still water depth is 0.1 m. Wind and bed stress components are set to zero in order for the simulation to be virtually two-dimensional. No-slip wall boundary conditions and the “minmod” slope limiter are applied. Time integration was undertaken using the second-order implicit scheme.

Figure 5 shows the fixed coarse grid of 1,792 cells and predicted velocity field at $t = 100$ s. The corresponding results obtained on a fixed fine grid of 3,328 cells are depicted in figure 6. There is a throughflow from inlet to outlet separating two counter-rotating gyres. The steady-state flow pattern on the fine grid is very similar to those obtained by Dennis (1974) and Borthwick and Barber (1990) who solved the two-dimensional Navier–Stokes equations for steady laminar flow in a circle with single inlet and outlet. A dynamically adaptive grid simulation has also been carried out, commencing with the coarse 1,792 cell grid on which $h$-refinement was achieved

<table>
<thead>
<tr>
<th>Variables</th>
<th>Analytical solution</th>
<th>$L2 = 1e-2$</th>
<th>$L2 = 1e-5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$ (m)</td>
<td>1.49984</td>
<td>1.498392</td>
<td>1.498392</td>
</tr>
<tr>
<td>$\bar{v}$ (m/s)</td>
<td>7.95308</td>
<td>7.884614</td>
<td>7.884614</td>
</tr>
<tr>
<td>Froude</td>
<td>2.07338</td>
<td>2.056531</td>
<td>2.056531</td>
</tr>
<tr>
<td>$\Delta t$ (s)</td>
<td>–</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>–</td>
<td>441</td>
<td>898</td>
</tr>
<tr>
<td>No of initial iterations</td>
<td>–</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>No of final iterations</td>
<td>–</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
according to criteria based on the following vorticity-related parameter:

\[
\xi = \frac{b}{U_b} \left| \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \right|.
\]

(21)

Mesh adaptation was implemented for \(\xi > 0.234\), a value that kept the overall number of cells created close to the initial one but at the same time the accuracy of the numerical computations was increased. Figure 7 depicts the dynamically-adapted grids and the associated velocity vector distribution at \(t = 15\) s, \(t = 30\) s, \(t = 50\) s and \(t = 100\) s as the flow evolves toward steady-state conditions. Figure 8 indicates the rapid growth in the number of cells as the mesh initially adapts to the separating flow at the inlet and the consequent spread of vorticity as lobes into the interior of the reservoir. After about 45 s, no more changes occur to the dynamically-adaptive grid. The final overall flow pattern is very similar to that obtained on the fixed fine grid, in particular the positions of the rotational centres. The final adapted grid contains 2,524 cells, a saving on the mesh storage requirement. This confirms that dynamic mesh adaptation can retain accuracy while providing savings in computational overhead in cases where the flow contains structures at different scales or variable intensity (e.g. sheared flows). This increase of accuracy, however, seems to be bound with the nature of the imposed case-specific adaptivity criterion as a second adaptive simulation of the present test case indicated: if the adaptation criterion becomes \(\xi = \frac{b}{U_b} |(\partial u/\partial y) + (\partial v/\partial x)|\), the resulting final mesh resolution, focuses on the centre-line of the reservoir connecting its inlet and outlet (because of the magnitude of the local underlying velocity gradients) without it being able to enhance the overall accuracy. Hence, adaptive solvers can contribute towards numerical accuracy as along as the imposed adaptivity criteria are capable of describing the dominant characteristic of the computed flow field.

4.3 Wind induced circulation in a circular basin of uneven bathymetry

The present test considers the simulation of topographic gyres in a shallow circular basin due to a steady uniform wind field, originally considered by Kranenburg (1992).
This test is useful for checking the numerical model's ability to account correctly for non-uniform bed topography, surface stresses and viscous effects.

Kranenburg fitted an analytical stream function solution to a reduced rigid-lid form of the non-linear shallow water equations, and derived the following axially-symmetric expression for the still water depth:

\[ h_s = \frac{0.5 + (0.5 - 0.5r/R)^{1/2}}{1.3} \] (22)
where $r$ is the radial distance from the centre of the lake and $R$ is its radius. Kranenburg’s streamline pattern consisted of a pair of gyres, rotating such that the direction of the depth-averaged velocity in the deepest central part of the basin was opposite that of the wind. Kranenburg assumed that the flow was steady, non-linear advection terms could be ignored, the bed roughness height was proportional to the water depth, and the eddy viscosity had a parabolic distribution with depth. Kranenburg derived the following expression for the depth-averaged velocity component perpendicular to the wind direction, across the centre of the basin,

$$U_o = \frac{u_*}{\kappa} \left( \frac{h}{H} - 1 \right) \ln Z$$

where $u_* = (\tau_w/\rho)^{1/2}$, $\tau_w$ is the wind shears stress, $\kappa \approx 0.4$ is the von Kármán constant, $Z = H/z_o$, $H$ is the weighted mean water depth, and $z_o$ is the bed roughness height.

For the numerical experiment the following physical parameters have been used: $R = 192$ m; $H = 0.769$ m; $\tau_w = \tau_{wx} = 0.02$ N/m$^2$; $C = 29.53$ m$^{1/2}$/s; $\nu_r = 0.00012$ m$^2$/s; $f = 0$. Slip wall conditions are applied at the outer boundary of the basin. It is noted that the Chezy coefficient selected corresponds to laminar flow of mean depth 0.769 m over a bed where $z_o = 0.0065$ m (noting that $C = (g/C_D)^{1/2}$, and the drag coefficient $C_D = [0.4/(1 + \ln(z_o/h))]^2$, see Soulsby (1998)). The resulting time step for both explicit and implicit time integration simulations is 0.8 s with the $L^2$ tolerance being $10^{-6}$. The corresponding relative CPU times are 939 and 2,092 s respectively. Model predictions of the free surface elevation indicate that Kranenburg’s rigid-lid approximation is reasonable. Figure 9 shows the computational mesh of 2,194 cells, while figure 10 depicts the steady-state velocity field at 5,000 s. It is noted that almost identical results were obtained using the explicit and implicit time integration schemes, see figure 11 where the predicted and analytical steady-state non-dimensional velocity profiles across the basin (perpendicular to the wind direction) are shown.

### 4.4 Toro’s circular dam break

Using an extremely fine mesh with 1,000 cells in the radial direction, Toro (2001) has presented almost exact predictions of the evolving flow field (water depth and velocity profiles) due to an idealised circular dam break in a $40 \times 40$ m square basin with a horizontal bed. The initial water depth of the 5 m diameter water column is 2.5 m while the depth outside the column is 0.5 m. This case stands as a challenging unsteady case for verifying...
the accurate character of the present solver and its grid adaptation module for implicit and explicit time integration as well as for demonstrating clearly its time saving capabilities. Hence, all four test cases described in table 4 were computed with $L_2 = 1e-2$ being applied for the time implicit calculations. Moreover, for all test cases the bed and wind stresses, the eddy viscosity and the Coriolis parameter are set to zero, while the “superbee” limiter is applied.

Figure 12 depicts the time dependent behaviour of the free surface elevation in the two- and three-dimensional space as well as the velocity profile across the basin that are in excellent agreement with Toro’s results. It is noted that these results are based on a steady mesh that has $\sim$130 cells in the radial direction (16,384 in total) and implicit time integration has been utilised. Figure 13 presents the free surface elevation at $t = 3.0$ s for all cases described in table 4 and demonstrates that implicit time integration gives a slightly smoother character to the solution for all adaptive and non-adaptive results shown, see the edges of the radial bore. On the other hand, implicit time integration is more time consuming than its explicit counterpart, see table 4, despite the fact that it employs a bigger $\Delta r$. Figure 14 presents the mesh configuration and the free surface contours of the time explicit adaptive case of table 4 at certain times chosen to demonstrate indicative changes in the mesh configuration which adapts not only to the moving radial bore at the front of the dam break but to the free surface leveling and spreading of the initial water column as well. The number of cells for this case evolves as shown in figure 15, while mesh adaptation for both adaptive cases of table 4 occurs for values of $\xi$ bigger than 0.22 ($\xi$ being described by equation (20)). Table 4 indicates a clear time saving ability of the adaptive scheme which enhances the grid in areas that demonstrate a steep free surface profile according to the adaptation criterion being utilised.

5. Conclusions

A Godunov-type scheme that solves a hyperbolic set of the shallow water equations has been presented. The mathematical formulation of the shallow water equations adopted has been proved capable of accounting for non-uniform bed topographies. This ability of the formulation seems to be well supported by a simple vector-based technique adopted for the calculation of the source term involved gradients. Time integration in the present scheme has been achieved through explicit and implicit formulations of varying order of accuracy. The fluxes have been evaluated using Roe’s approximate Riemann solver, as applied on dynamically-adaptive unstructured triangular grids while the overall robustness of the scheme is enhanced through the global polynomial enhancement of the conserved variables’ slope within each cell. It has been found that grid adaptation leads to enhanced computational efficiency and accuracy — as long as the adaptivity criterion used is capable of depicting the main flow characteristics — while implicit time integration little contributes towards smoothness of the results but increases the stability of the scheme as the code iterates within each time step targeting convergence. Results presented for an oblique hydraulic jump, a jet-forced flow in a circular flat-bottomed basin, wind-induced circulation in a basin of uneven bathymetry and the collapse of a circular dam are in very close agreement with analytical solutions and alternative numerical predictions in the literature.
Figure 12. Toro's circular dam break for the time implicit non-adaptive case of table 4 on a mesh of ~130 cells in the radial direction: water depth, velocity profiles across the basin and 3-D representation of the free surface at times: (a) $t = 0.4$ s; (b) $t = 0.7$ s; (c) $t = 1.4$ s; (d) $t = 3.5$ s; and (e) $t = 4.7$ s.
Figure 13. Toro’s dam break: free surface elevation at $t = 3.0$ s for all cases of table 4, i.e.: (a) time explicit non-adaptive simulation; (b) time implicit non-adaptive simulation; (c) time explicit adaptive simulation and (d) time implicit adaptive simulation.

Figure 14. Toro’s dam break: mesh configuration and free surface contours at times: (a) $t = 0.02$ s; (b) $t = 1.50$ s; (c) $t = 2.00$ s and (d) $t = 2.50$ s for the time explicit adaptive simulation of table 4.
Figure 15. Toro’s dam break: number of cells involved in the computational field for the time explicit adaptive case of table 4.

Table 4. Numerical results: different time integrators for adaptive and non-adaptive computations, Toro’s circular dam break.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Non-adaptive results</th>
<th>Adaptive results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a_i = 0$</td>
<td>$a_i = 0.5$</td>
</tr>
<tr>
<td>No of initial cells</td>
<td>16,384</td>
<td>16,384</td>
</tr>
<tr>
<td>No of final cells</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>$\Delta t$ (s)</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>1470</td>
<td>320</td>
</tr>
<tr>
<td>No of initial and final</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>iterations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Acknowledgements

This work has been partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through grant GR/L92907 and co-investigated by Dr K. Anastasiou of Imperial College, University of London, as well as Professor P.H. Taylor and Dr B. Rogers of Oxford University. Moreover, Dr C.T. Chan’s (Imperial College, University of London) support and encouragement are kindly acknowledged.

References


