INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2009; **59**:459–474
Published online 2 June 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1827

A hybrid building-block and gridless method for compressible flows

Hong Luo[1,*,†,‡], Joseph D. Baum[2,§] and Rainald Löhner[3,¶]

[1]*Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, U.S.A.*
[2]*Center for Applied Computational Sciences, Science Applications International Corporation, McLean, VA 22102, U.S.A.*
[3]*School of Computational Sciences, George Mason University, Fairfax, VA 22030, U.S.A.*

SUMMARY

A hybrid building-block Cartesian grid and gridless method is presented to compute unsteady compressible flows for complex geometries. In this method, a Cartesian mesh based on a building-block grid is used as a baseline mesh to cover the computational domain, while the boundary surfaces are represented using a set of gridless points. This hybrid method combines the efficiency of a Cartesian grid method and the flexibility of a gridless method for the complex geometries. The developed method is used to compute a number of test cases to validate the accuracy and efficiency of the method. The numerical results obtained indicate that the use of this hybrid method leads to a significant improvement in performance over its unstructured grid counterpart for the time-accurate solution of the compressible Euler equations. An overall speed-up factor from six to more than one order of magnitude and a saving in storage requirements up to one order of magnitude for all test cases in comparison with the unstructured grid method are demonstrated. Copyright © 2008 John Wiley & Sons, Ltd.

Received 14 October 2007; Revised 8 March 2008; Accepted 24 March 2008

KEY WORDS: Cartesian grid methods; gridless methods; building-block methods; Euler equations; unsteady compressible flows; shock waves

*Correspondence to: Hong Luo, Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, U.S.A.
†E-mail: hong_luo@ncsu.edu
‡Associate Professor.
§Director.
¶Professor.

Contract/grant sponsor: Defense Threat Reduction Agency

Copyright © 2008 John Wiley & Sons, Ltd.

1. INTRODUCTION

Over the course of last decade, significant progress has been made on developing numerical methods for the solution of the compressible Euler and Navier–Stokes equations. In general, these numerical methods can be classified by the mesh they use as structured grid methods, unstructured grid methods, Cartesian grid methods, and gridless methods. Each of these methods is advocated, promoted, developed, and used by their respective supporters. Since each method has its own advantages and disadvantages, the answer to which method is preferable depends on the problem to be solved. The structured grid methods [1–3] have a disadvantage in mesh generation for complex geometries. The main advantage of the unstructured grid methods [4–6] is the ease of grid generation for complex configurations. However, the computational costs and memory requirements are generally higher than their structured grid counterparts. The advantages of the Cartesian grid methods [7–9] include ease of grid generation, lower computational storage requirements, and significantly less operational count per cell. However, the main challenge in using Cartesian methods is how to deal with arbitrary boundaries, as the grids are not body-aligned. The cells of a Cartesian mesh near the body can extend through surfaces of boundaries. Accurate means of representing boundary conditions in cells that intersect surfaces are essential for successful Cartesian methods. Lately, gridless methods [10–15] came into focus. This class of methods is essentially propelled by the fact that there exists a perceived difficulty in generating volume filling grids for complex geometries in spite of significant progress in the theory and practice of mesh generation over the last decade. However, global conservation of mass, momentum, and energy for these methods is not necessarily assured. Furthermore, these gridless methods are generally slower than their mesh-based counterparts.

Recently, a new approach [16] named building-cube method has been developed for large-scale high-resolution flow computations around complex geometries. In this method, a flow field is divided into a number of cubes (squares in two dimension) of various sizes. Each cube is a computational sub-domain with a Cartesian mesh of equal spacing and equal number of nodes. The geometrical size of each cube is determined by adapting to geometry and flow features so as to cope with broadband characteristic lengths of the flow. However, in this approach, the wall boundary is defined by a staircase representation in order to keep the simplicity of the algorithm and to minimize the memory requirement per node. To keep the geometrical accuracy by the staircase representation of wall boundaries, very fine mesh that resolves viscous sub-layers in the boundary layers is used. It is argued that with this mesh size, the staircase representation is accurate enough to approximate the curved wall boundary. Unfortunately, it is not appropriate, convincing, and practical to use such a staircase representation to approximate the curved boundaries where one cannot afford such highly dense grids.

The objective of the efforts presented in this paper is to develop a fast and efficient numerical method for time-accurate computation of the compressible Euler equations using a hybrid building-block and gridless method for time-accurate computation of the unsteady compressible Euler equations. The development of such a numerical method is strongly motivated by the need to be able to simulate blast and shock waves for complex geometries on a personal computer platform in a reasonable time and accuracy. Simulation of high explosive detonation, blast propagation, and shock wave diffraction plays an important role in determining and assessing target vulnerability and weapon lethality. The idea is to combine the advantages of both gridless and building-block Cartesian methods in an attempt to develop a fast, low storage method for complex geometries. This method uses a gridless method to address the boundary or interface while a building-block

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

Cartesian grid method is used everywhere else. This method can be regarded as a natural extension of the so-called hybrid Cartesian and gridless methods [17–20] to treat non-uniform grids. Since a majority of computational cells are solved using the building-block Cartesian grid method, the resulting method is very efficient in terms of both computational cost and storage requirements. Since the gridless method is used to solve computational cells in the boundary and interface, the developed method can be easily used for complex geometries without a need to generate a computational mesh. The developed method has been used to compute a number of test cases to validate the accuracy and efficiency of the method. The numerical results obtained indicate that the use of this hybrid method leads to a significant increase in performance over its unstructured grid counterpart. An overall speed-up factor from six to more than one order of magnitude and a saving in storage requirements up to one order of magnitude for all test cases in comparison with the unstructured grid method are demonstrated.

2. GOVERNING EQUATIONS

The Euler equations governing unsteady compressible inviscid flows can be expressed in conservative form as

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}+\frac{\partial \mathbf{H}}{\partial z}=0 \tag{1}$$

where the conservative state vector $\mathbf{U}$ and the inviscid flux vectors $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ are defined as

$$\mathbf{U}=\begin{pmatrix}\rho\\ \rho v_x\\ \rho v_y\\ \rho v_z\\ \rho e\end{pmatrix},\quad \mathbf{F}=\begin{pmatrix}\rho v_x\\ \rho v_x^2+p\\ v_x\rho v_y\\ v_x\rho v_z\\ v_x(\rho e+p)\end{pmatrix},\quad \mathbf{G}=\begin{pmatrix}\rho v_y\\ \rho v_y v_x\\ \rho v_y^2+p\\ \rho v_y v_z\\ v_y(\rho e+p)\end{pmatrix},\quad \mathbf{H}=\begin{pmatrix}\rho v_z\\ \rho v_z v_x\\ \rho v_z v_y\\ \rho v_z v_z+p\\ v_z(\rho e+p)\end{pmatrix} \tag{2}$$

where ρ, p, and e denote the density, pressure, and specific total energy of the fluid, respectively, and v_x, v_y, and v_z are the velocity components of the flow in the coordinate direction x, y, z, respectively. This set of equations is completed by the addition of the equation of state

$$p=(\gamma-1)\rho(e-\tfrac{1}{2}(v_x^2+v_y^2+v_z^2)) \tag{3}$$

which is valid for perfect gas, where γ is the ratio of the specific heats.

3. NUMERICAL METHOD

3.1. Basic approach

The basic idea behind the present hybrid method is to combine a building-block method and a gridless method in an attempt to develop a fast, low storage method for time-accurate computation of the unsteady Euler equations for complex geometries. In this method, body geometries are first defined and the desired grid sizes in the flow field are specified via a background mesh and/or sources. A baseline block (root block) covering the computational domain is then refined to create

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

eight blocks of equal size. In turn, these blocks are refined again until a suitable block size is achieved, which is determined by the specified number of cells in x-, y-, and z-directions in a block and the prescribed grid size. Each block is a sub-domain of the flow computation, where a uniform-spacing Cartesian mesh is used. All blocks have the same number of Cartesian cells in all three directions, so that the local computational resolution is determined by the block size. In order to minimize the numerical errors between blocks, a smoothing of the size differences among blocks is performed, allowing only the size difference between two adjacent blocks to two. An unstructured data structure is used to manage the blocks and communicate with each other. Meanwhile, a triangulation of the body geometries is performed using the advancing front method based on the representation of the body geometries and the prescribed grid size. The center points of triangles are chosen as a set of boundary points, which will be updated using the gridless method. Note that such chosen points have a well-defined normal direction, therefore avoiding ambiguity of normal definitions for surface singularities such as the tip of an airfoil trailing edge [16–18]. Since a majority of computational cells are solved using the building-block Cartesian method, the resulting method is very efficient in terms of both computational cost and storage requirements. Since the gridless method is used to solve computational cells in the boundary and interface, the developed method can easily be used for complex geometries without a need to generate a computational mesh. Note that when the number of cells in all three directions is set to one in each block, a classical octree-based Cartesian grid will be generated. The underlying numerical method becomes the unstructured Cartesian grid method, which can be used to serve as a good reference in assessing the efficiency of the present hybrid method.

3.2. Determination of gridless points

As the solid boundary points are immersed in the building-block Cartesian grid, three types of nearby cells can be identified as shown in Figure 1. Cartesian cells are cells whose solutions can be computed using a Cartesian grid method, since the values of all their neighbors are known and the stencil for the Cartesian grid method is complete. Cut-off cells are cells that contain one or more boundary points or embed within a body. These cut-off cells are omitted from the computation. Transitional cells are cells that have cut-off cells as its neighbors. The solution for these cells and the boundary points is obtained using a gridless method. An unstructured data structure is used to manage the solution of these transitional cells and boundary points, which will be termed gridless points hereafter. The procedure for the determination of cut-off, Cartesian, and transitional cells is simple and straightforward, which is listed as follows:

1. All the cells are first marked as Cartesian cells.
2. The cells that have at least one node out of computational domain are marked as cut-off cells. This is achieved using the alternate digital tree method [21] and a ray-casting approach.
3. The neighboring cells to the cut-off cells are then marked as transitional cells.

3.3. Determination of cloud points for gridless points

A gridless method requires the construction of a local cloud of points for every gridless point. How to construct these local clouds is crucial to the success of the gridless method. In this study, the local clouds for the surface gridless points are constructed first. For each surface gridless point, a host Cartesian cell, which contains this point, is first determined. Those adjacent Cartesian cells having at least one shared vertex with the host cell along with the neighboring gridless points (these

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

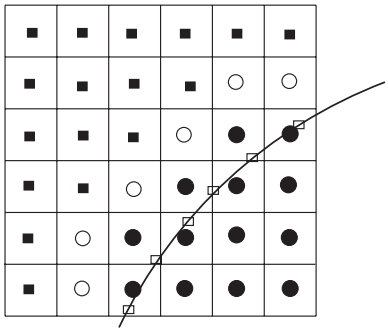

Figure 1. Classification of grid cells: ■, Cartesian cells; ○, transitional cells; ●, cut-off cells; and □, boundary points.

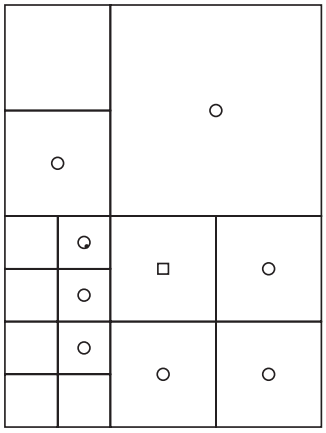

Figure 2. Cloud points for a gridless point: □, host cell and ○, cloud points.

surface triangles having at least one shared vertex with this gridless triangle) are then considered as cloud points as shown in Figure 2. Note that this choice of cloud points leads to $3 \times 3 \times 3$ stencil for a host cell whose neighboring cells have the same size as the host cell. Among these cells, the cut-off cells will be removed. Those cells located in the opposite direction of this point will also removed in order to avoid choosing cells from the wrong side of the profile at thin surfaces. Next, if this surface gridless point has any transitional gridless points in its local clouds, this surface gridless point will reciprocally be added to the local clouds of these transitional gridless points. These surface gridless points along with the neighboring Cartesian cells, determined in a similar way as for a surface gridless point, will construct the local clouds for the transitional gridless points. Finally, a local Delaunay triangulation [14] is performed to further optimize the local clouds for the transitional gridless points.

3.4. Gridless method

For all the gridless points, the governing equation (1) is discretized using a gridless method. In recent years, significant progress has been made in developing gridless methods for the compressible

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

Euler and Navier–Stokes equations [10–15]. Gridless methods are attractive due to the fact that they are free of generating a computational mesh, which is still a daunting challenge for complex geometries. Almost all of gridless methods make use of a least-square formulation and they differ from one another in the way they introduce artificial dissipations, which are essential and necessary for the governing hyperbolic equations. In this study, the dual least-squares approximation, a variant of the least-squares approximation, is used to compute the inviscid fluxes. This dual least-squares approximation provides a natural way to use upwind-type discretization for the inviscid fluxes of the Euler equations. Assume that C_i is the set of cloud points for a given point i. Let f_{ij} denote the value of any flux functions f at the mid-point of the edge **ij**, where $j \in C_i$. Assuming that the solution varies linearly along an edge **ij** and using Taylor's formula about the point i, one obtains

$$\left.\frac{\partial f}{\partial x}\right|_i (x_{ij}-x_i)+\left.\frac{\partial f}{\partial y}\right|_i (y_{ij}-y_i)+\left.\frac{\partial f}{\partial z}\right|_i (z_{ij}-z_i)=f_{ij}-f_i \tag{4}$$

where x_{ij}, y_{ij}, and z_{ij} are the coordinates for the mid-point of the edge **ij**. Similar equations could be expressed for all cloud points associated with point i, subject to an arbitrary weighting factor w_i. This yields the following non-square matrix:

$$\begin{pmatrix} w_1(x_{i1}-x_i) & w_1(y_{i1}-y_i) & w_1(z_{i1}-z_i) \\ \vdots & \vdots & \vdots \\ w_n(x_{in}-x_i) & w_1(y_{in}-y_i) & w_1(z_{in}-z_i) \end{pmatrix} \begin{pmatrix} \left.\frac{\partial f}{\partial x}\right|_i \\ \left.\frac{\partial f}{\partial y}\right|_i \\ \left.\frac{\partial f}{\partial z}\right|_i \end{pmatrix} = \begin{pmatrix} w_1(f_{i1}-f_i) \\ \vdots \\ w_n(f_{in}-f_i) \end{pmatrix} \tag{5}$$

where n is the number of cloud points for the point i. This formulation provides a freedom in the choice of weighting coefficients w_j. These weighting coefficients can be selected as a function of the geometry and/or solution. Classical approximations in one dimension can be recovered by choosing geometrical weights of the form $w_j = 1.0/|\mathbf{r}_{ij}-\mathbf{r}_i|^{\beta}$ for values of $\beta = 0, 1, 2$, where $\mathbf{r}_{ij} = (x_{ij}-x_i, y_{ij}-y_i, z_{ij}-z_i)$ and $\mathbf{r}_i = (x_i, y_i, z_i)$. The numerical computations shown in the next section are performed using $\beta = 0$.

Equation (5) can be solved using the least-squares method and its solution can be written as

$$\left.\frac{\partial f}{\partial x}\right|_i = \sum_{j\in C_i} a_{ij}(f_{ij}-f_i), \quad \left.\frac{\partial f}{\partial y}\right|_i = \sum_{j\in C_i} b_{ij}(f_{ij}-f_i), \quad \left.\frac{\partial f}{\partial z}\right|_i = \sum_{j\in C_i} c_{ij}(f_{ij}-f_i) \tag{6}$$

If the function values at the mid-points are simply taken to be the average values of its two end points $f_{ij} = (f_i + f_j)/2$, then the standard least-squares approximation of the derivatives is recovered:

$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{1}{2}\sum_{j\in C_i} a_{ij}(f_j-f_i), \quad \left.\frac{\partial f}{\partial y}\right|_i = \frac{1}{2}\sum_{j\in C_i} b_{ij}(f_j-f_i), \quad \left.\frac{\partial f}{\partial z}\right|_i = \frac{1}{2}\sum_{j\in C_i} c_{ij}(f_j-f_i) \tag{7}$$

Substituting the inviscid flux in Equation (1) by the dual least-squares approximation given by Equation (6), one obtains a semi-discrete form of the Euler equations at point i

$$\frac{\partial \mathbf{U}_i}{\partial t} + \sum_{j\in C_i} (a_{ij}(\mathbf{F}_{ij}-\mathbf{F}_i)+b_{ij}(\mathbf{G}_{ij}-\mathbf{G}_i)+c_{ij}(\mathbf{H}_{ij}-\mathbf{H}_i)) = 0 \tag{8}$$

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

In this dual least-squares formulation, fluxes are computed at the mid-points, which are not known *a priori* and have to be reconstructed in some way. It is by defining these numerical fluxes in a consistent and upwind manner that upwinding can be naturally introduced into the gridless methods and specified accuracy can be obtained. If the numerical fluxes at the mid-points are evaluated using the simple arithmetic averages of conservative variables at the two end points, the resulting numerical scheme is equivalent to the central differencing method. It is well known that such discretizations lead to unstable schemes, and must be augmented by stabilizing terms. This can be achieved either by adding directly second-, forth-, or higher-order damping, or by using upwinding schemes. Over the last two decades characteristic-based upwind methods have established themselves as the methods of choice for prescribing the numerical fluxes for compressible Euler equations. The upwinding schemes seem especially attractive in the present context, as they do not require any intrinsic measure of length. In this study, the numerical fluxes are approximated using the HLLC approximate Riemann solver [22], which has been successfully used to compute compressible viscous and turbulent flows on both structured grids [2] and unstructured grids [6]. HLLC scheme is known to be very robust in terms of positivity, entropy consistency, and convergence history. If the Riemann fluxes are evaluated using the flow variables $\mathbf{U}_i$ and $\mathbf{U}_j$, this scheme is equivalent to the first-order upwind scheme. Many different ways exist to achieve higher-order accuracy. In this study, a scheme of higher-order accuracy is obtained by using upwind-biased interpolations of the solution $\mathbf{U}$, via MUSCL approach [23]. The upwind-biased interpolations for $\mathbf{U}_i^+$ and $\mathbf{U}_j^-$ are defined as

$$\mathbf{U}_i^+ = \mathbf{U}_i + \tfrac{1}{4}[(1-k)\Delta_i^- + (1+k)(\mathbf{U}_j - \mathbf{U}_i)] \tag{9}$$

$$\mathbf{U}_j^- = \mathbf{U}_j - \tfrac{1}{4}[(1-k)\Delta_j^+ + (1+k)(\mathbf{U}_j - \mathbf{U}_i)] \tag{10}$$

where the forward and backward difference operators are given as

$$\Delta_i^- = \mathbf{U}_i - \mathbf{U}_{i-1} = 2(\nabla\mathbf{U})_i \cdot \Delta\sigma - (\mathbf{U}_j - \mathbf{U}_i) \tag{11}$$

$$\Delta_j^+ = \mathbf{U}_{j+1} - \mathbf{U}_j = 2(\nabla\mathbf{U})_j \cdot \Delta\sigma - (\mathbf{U}_j - \mathbf{U}_i) \tag{12}$$

where σ denotes a length coordinate along the edge nodes i and j of the grid and $\Delta\sigma = \sigma_j - \sigma_i$ is the length of this edge. The gradients $(\nabla\mathbf{U})_i$ and $(\nabla\mathbf{U})_j$ are computed using the standard least-squares approximation (7).

The parameter k in Equations (9) and (10) can be chosen to control a family of difference schemes in the interpolation. On structured meshes it is easy to show that $k=-1$ yields a fully upwind scheme, $k=0$ yields semi-upwind approximation (Fromm's scheme), and $k=1$ yields central differencing. The value $k=\frac{1}{3}$ leads to a third-order-accurate upwind-biased scheme, although third-order-accuracy is strictly correct only for one-dimensional calculations. Nevertheless, $k=\frac{1}{3}$ was used in the calculations presented herein. With higher-order spatial accuracy, spurious oscillations in the vicinity of shock waves are expected to occur. Some form of limiting is usually required to eliminate these numerical oscillations of the solution and to provide some kind of monotonicity property. The flux limiter modifies the upwind-biased interpolation $\mathbf{U}_i$ and $\mathbf{U}_j$, replacing them by

$$\mathbf{U}_i^+ = \mathbf{U}_i + \frac{s_i}{4}[(1-ks_i)\Delta_i^- + (1+ks_i)(\mathbf{U}_j - \mathbf{U}_i)] \tag{13}$$

$$\mathbf{U}_j^- = \mathbf{U}_j - \frac{s_j}{4}[(1-ks_j)\Delta_j^+ + (1+ks_j)(\mathbf{U}_j - \mathbf{U}_i)] \tag{14}$$

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

where s_i and s_j are the flux limiters. The van Albada limiter is employed in this study. It acts in a continuously differentiable manner and is defined by

$$s_i = \max\left\{0, \frac{2\Delta_i^-(\mathbf{U}_j - \mathbf{U}_i) + \varepsilon}{(\Delta_i^-)^2 + (\mathbf{U}_j - \mathbf{U}_i)^2 + \varepsilon}\right\} \tag{15}$$

$$s_j = \max\left\{0, \frac{2\Delta_j^+(\mathbf{U}_j - \mathbf{U}_i) + \varepsilon}{(\Delta_j^+)^2 + (\mathbf{U}_j - \mathbf{U}_i)^2 + \varepsilon}\right\} \tag{16}$$

where ε is a very small number to prevent division by zero in smooth regions of the flow. Three options exist concerning the choice of interpolation variables: conservative variables, primitive variables, and characteristic variables. Using limiters on characteristic variables seems to give the best results. However, the primitive variables are used in this study for the sake of computational efficiency.

3.5. Cartesian grid method

For all the Cartesian cells, the governing equation (1) is discretized using a standard cell-centered finite volume formulation, where the control volume is taken as the cell of the Cartesian grid itself, and the cell-averaged variables are stored at the center of the cell. The finite volume approximation of the governing equations, applied to the cell (i, j, k), becomes

$$V\frac{\partial \mathbf{U}_{i,j,k}}{\partial t} + \int_{\Gamma_{i,j,k}} (\mathbf{F}n_x + \mathbf{G}n_y + \mathbf{H}n_z)\,\mathrm{d}\Gamma = 0 \tag{17}$$

where V is the cell volume, $\Gamma_{i,j,k}$ is the boundary of the cell (i, j, k), and $\mathbf{n} = (n_x, n_y, n_z)$ denotes the unit outward normal vector to the boundary of the cell. The flux integral in Equation (17) is evaluated by summing all the contributions over the cell interfaces between the cell (i, j, k) and its neighboring cells. Equation (17) can then be rewritten in a compact form as

$$\frac{\mathrm{d}\mathbf{U}_{i,j,k}}{\mathrm{d}t} = \mathbf{R}_{i,j,k} \tag{18}$$

where $\mathbf{R}_{i,j,k}$ is the right-hand side residual,

$$\mathbf{R}_{i,j,k} = -\left(\frac{\mathbf{F}_{i+1/2,j,k} - \mathbf{F}_{i-1/2,j,k}}{\Delta x} + \frac{\mathbf{G}_{i,j+1/2,k} - \mathbf{G}_{i,j-1/2,k}}{\Delta y} + \frac{\mathbf{H}_{i,j,k+1/2} - \mathbf{H}_{i,j,k-1/2}}{\Delta z}\right) \tag{19}$$

where $\Delta x, \Delta y$, and Δz are the cell size in the x-, y-, z-directions, respectively.

As for the gridless points, the numerical fluxes at the interface in Equation (19) are computed using the HLLC approximate Riemann solver. Second-order accuracy is achieved using the reconstruction scheme. The van Albada limiter is used to suppress oscillations in the vicinity of discontinuities.

3.6. Temporal discretization

Equations (8) and (18) represent the time evolution of the unknown vector, which can be expressed as the following semi-discrete form for both Cartesian cells and gridless points:

$$\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} = \mathbf{R}_i \tag{20}$$

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

where $\mathbf{R}_i$ is the residual vector. Assuming that the unknown vector $\mathbf{U}_i^n$ is known at time t_n, the solution is advanced over a time step Δt, to time t_{n+1}, by an explicit multi-stage Runge–Kutta time-stepping scheme given as

$$\begin{aligned}
\mathbf{U}_i^{(0)} &= \mathbf{U}_i^n \\
&\vdots \\
\mathbf{U}_i^{(p)} &= \mathbf{U}_i^{(0)} - \alpha_p \Delta t \mathbf{R}_i(\mathbf{U}_i^{(p-1)}), \quad p = 1, 2, \ldots, m \\
&\vdots \\
\mathbf{U}_i^{n+1} &= \mathbf{U}_i^{(m)}
\end{aligned} \tag{21}$$

with the parameters α_p assigned appropriate values. For the four-stage Runge–Kutta scheme used in this study, $\alpha_1 = \frac{1}{4}$, $\alpha_2 = \frac{1}{3}$, $\alpha_3 = \frac{1}{2}$, and $\alpha_4 = 1$.

4. COMPUTATIONAL RESULTS

All of the computational results to be presented in this section are performed on a Dell Precision M70 laptop computer (2.13 GHz Pentium M CPU with 2 GBytes memory) running the Suse 10.1 Linux operating system. A cell-vertex finite volume code [5, 6] is used as a reference to demonstrate the numerical accuracy and computational efficiency of the hybrid building-block Cartesian grid and gridless method. All computations use an explicit four-stage Runge–Kutta time-stepping scheme to advance the solution in time.

4.1. Test case 1: steady two-dimensional oblique shock wave

The first example is the simulation of a two-dimensional shock wave reflecting from a rigid surface. This test case is chosen to validate and verify the baseline building-block Cartesian grid method, which is the foundation of the proposed hybrid method. The computational domain is a rectangle of length 4.1 and height 1. The boundary conditions are that of a reflecting surface along the bottom surface, supersonic outflow along the right surface, and prescribed fixed values on the other two sides, which are

$$(\rho, u, v, p)_{(0,y,t)} = (1, 2.9, 0, 1/1.4)$$

$$(\rho, u, v, p)_{(x,1,t)} = (1.69997, 2.61934, -0.50632, 1.52819)$$

The boundary conditions produce an incident shock angle of 29° and the free stream Mach number M_∞ is 2.9. Figure 3 shows the unstructured boundary surface mesh generated using a specified uniform grid size of 0.05. Figure 4 displays the computed density contours in the flow field for the shock reflection problem. The computed density distribution is compared with the exact solution at $y = 0.5$ in Figure 5. When the numbers of cells in x-, y-, and z-directions are specified as $82 \times 20 \times 20$, a single block will be generated automatically. When these numbers are halved, eight blocks will be created. When the numbers of cells in all three directions are set to one, 32 800 ($82 \times 20 \times 20$) blocks will then be generated. The central processing unit (CPU) cost

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

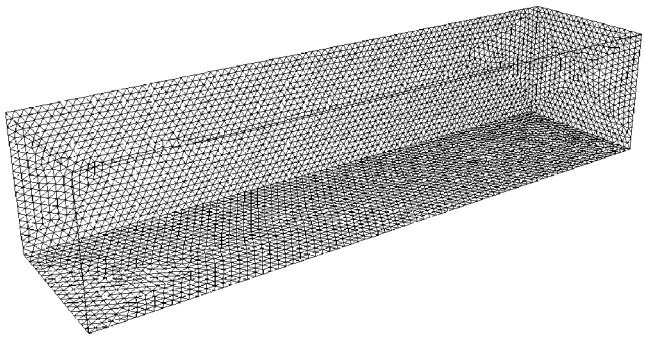

Figure 3. Unstructured boundary surface mesh used for the representation of gridless points for the stationary shock reflection problem (nface = 16548, nboun = 8276).

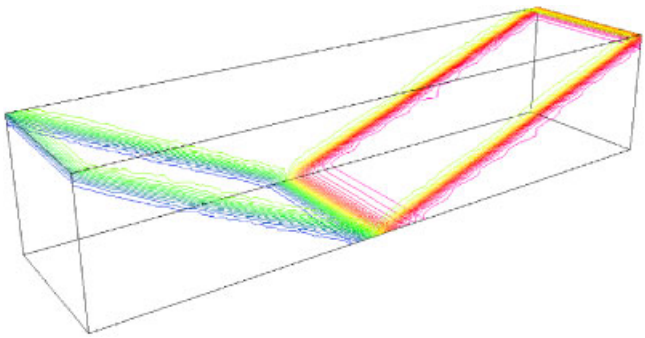

Figure 4. Computed density contours in the flow field for the stationary shock reflection problem.

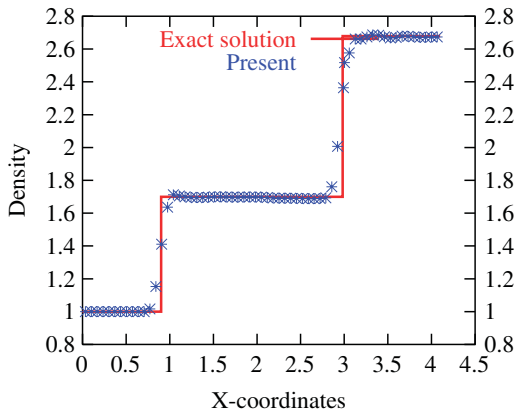

Figure 5. Comparison of the computed density profile with the exact solution at $y = 0.5$ for the stationary shock reflection problem.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

Table I. Computing cost analysis for the building-block Cartesian grid and unstructured grid methods.

Method	FEFLO	Building-block (32 800)	Building-block (8)	Building-block (1)
CPU (s)	956	197	135	132
No. of faces (edges)	343 780	102 080	102 080	102 080
Speed-up (fluxes)	1	3.37	3.37	3.37
Speed-up (others)	1	1.44	2.10	2.15
Speed-up (total)	1	4.85	7.08	7.24

between the building-block Cartesian grid method and the unstructured grid method for 500 time steps is compared in Table I, where the number of faces for the building-block Cartesian grid and the number of edges for the unstructured grid are also given. The unstructured tetrahedral grid consists of 283 359 elements, 52 148 points, and 8276 boundary points. Note that computing cost for the unstructured grid method is proportional to the number of edges, while that of the building block Cartesian grid method is proportional to the number of faces. In both algorithms evaluation of the Riemann fluxes consumes most of the CPU time. On the one hand, when a single block is used, the building-block Cartesian grid method becomes the traditional Cartesian grid method. A total speed-up factor of 7.24 is observed for this case, where a speed-up factor of two is achieved due to the saving in both indirect addressing and fewer operations (named other in the table), representing the best scenario of the building-block Cartesian grid method. On the other hand, when 32 800 blocks are used, the building-block Cartesian grid method becomes the traditional unstructured Cartesian grid method. A speed-up factor of 4.85 is still observed for this case mainly due to the lower number of Riemann flux evaluations, representing the worst scenario of the building-block Cartesian grid method, although the speed-up factor due to the indirect addressing and fewer operations is reduced, as expected. In addition, a larger time step can be used in the building-block Cartesian grid method due to the larger size of cells, leading to another speed-up factor of 1.8. As a result, the building-block Cartesian grid method is 13 times faster using a single block and 8 times faster using 32 800 blocks than its unstructured grid counterpart.

4.2. Test case 2: an incident shock past a cube

In this case, an incident shock $M_s = 2$ past a unit cube in a channel is computed. The computational domain is a block of $5 \times 3 \times 5$. This test case is chosen to validate and verify the hybrid building-block Cartesian grid and gridless method and to compare the accuracy and efficiency between the present hybrid and unstructured grid methods. Figure 6 shows the unstructured boundary surface mesh generated using a uniform grid size of 0.08, which serves as a set of gridless points to our hybrid method for the representation of embedded boundaries. The underlying unstructured tetrahedral grid used for our unstructured grid method, consists of 1 268 020 elements, 224 833 points, and 20 219 boundary points. Figure 7 displays the computed density contours in the flow field at two different times obtained using the hybrid method. Figure 8 shows a comparison of pressure and impulse time histories at two different locations between the present hybrid method and the unstructured grid method, where one can see that both methods produce virtually identical results. The computation is performed by the hybrid building-block Cartesian grid and gridless method using 1 block, 8 blocks, and 146 072 ($62 \times 38 \times 62$) blocks, respectively. The number of

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

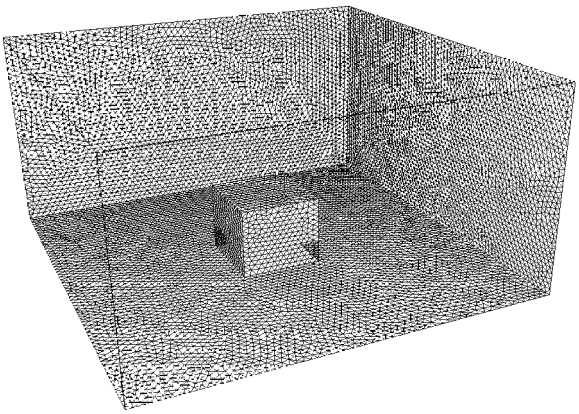

Figure 6. Unstructured boundary surface mesh used for the representation of gridless points for an incident shock $M_s = 2$ past a cube (nface = 40 434, nboun = 20 219).

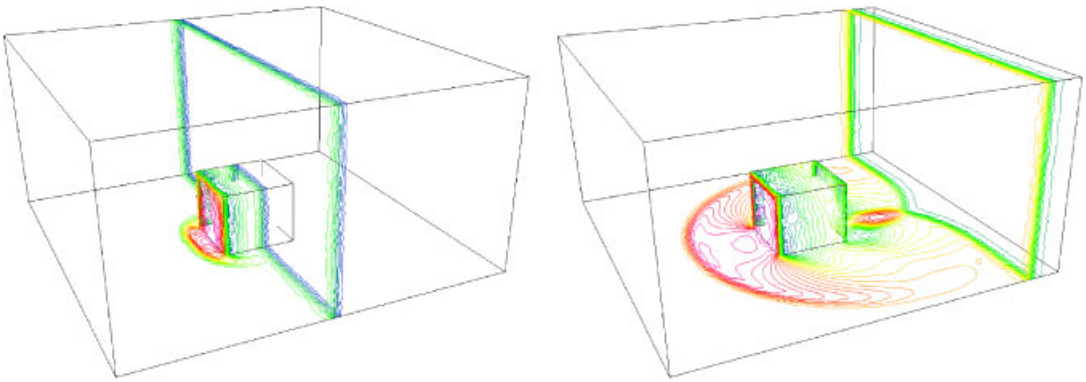

Figure 7. Computed density contours for an incident shock $M_s = 2$ past a cube at different times using the hybrid building-block Cartesian grid and gridless method.

gridless points is 2718 and the number of Cartesian cells is 143 524 in this case. An analysis of an execution profile indicates that the CPU time spent on gridless points is between 2 and 4% of the total CPU time, depending on the number of blocks used in the computation, since the number of gridless points is extremely small as compared with overall Cartesian cells. The average number of cloud points for each gridless point is about 17. The CPU cost between the building-block Cartesian grid method and the unstructured grid method for 180 time steps is compared in Table II in order to demonstrate the superior performance of the present hybrid building-block Cartesian grid and gridless method over the unstructured grid method for a typical computation of the shock wave propagation and diffraction problem. An overall speed-up factor from 4.9 to 8.4, depending on the number of blocks, in comparison with the unstructured grid method is observed for this computation. Considering that a larger time step can be used in the present hybrid method due to

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

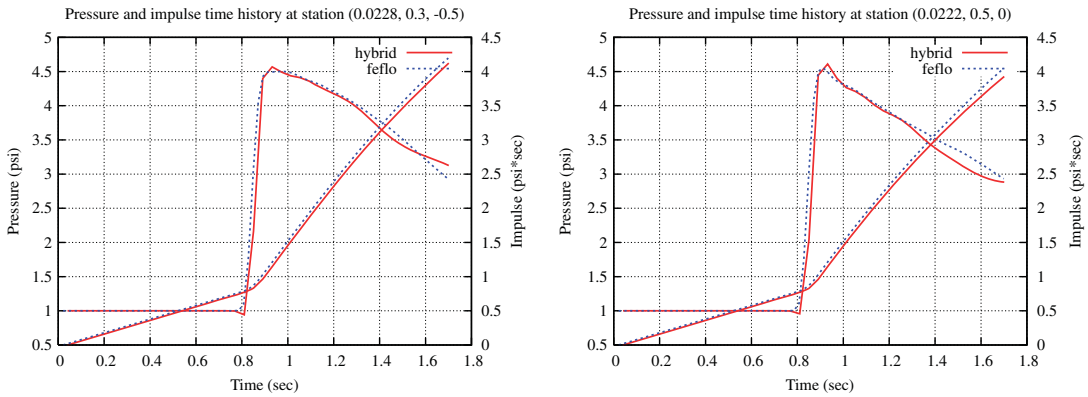

Figure 8. Comparison of pressure and impulse time history at different locations obtained using the hybrid method and the unstructured grid method.

Table II. Computing cost analysis for the building-block Cartesian grid and unstructured grid methods.

Method	FEFLO	Building-block (146 072)	Building-block (8)	Building-block (1)
CPU (s)	1514	307	185	180
No. of faces (edges)	1 513 069	441 156	441 156	441 156
Speed-up (fluxes)	1	3.4	3.4	3.4
Speed-up (others)	1	1.45	2.40	2.47
Speed-up (total)	1	4.93	8.18	8.41

the larger size of cells, this transfers to an overall speed-up factor of 5.5–12 over its unstructured grid counterpart.

4.3. Test case 3: a blast wave past a multi-buildings

As an example of application, the developed method is used for the simulation of a 50 kg TNT blast wave past a multi-building configuration. Figure 9 shows the automatically generated building-blocks used for the representation of Cartesian cells and unstructured boundary surface mesh used for the representation of gridless points based on the geometric definition of the configuration and specified grid size, respectively. This example is chosen to demonstrate the superior performance of the present hybrid building-block Cartesian grid and gridless method over the unstructured grid method for a typical simulation of blast waves past complex configurations. Figure 10 displays the computed pressure contours in the flow field at four different times obtained using the hybrid method. The number of blocks in this case is 106, the numbers of Cartesian cells in x-, y-, and z-direction inside a block are 13, 13, and 13. The number of Cartesian cells and the number of gridless points are 223 666 and 13 198, respectively. The unstructured tetrahedral grid with the same geometric definition file and specified mesh size, used for our unstructured grid method, consists of 1 559 837 elements, 273 847 points, and 16 124 boundary points. An overall speed-up factor about five in comparison with the unstructured grid method is observed for this computation.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

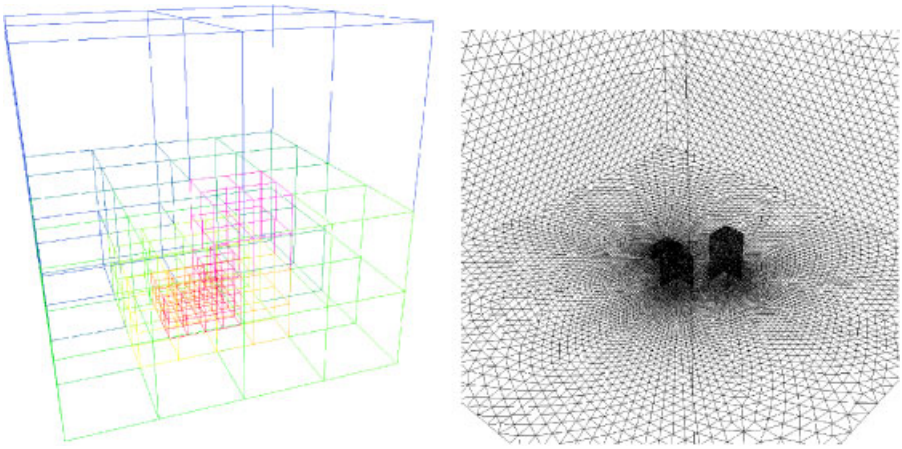

Figure 9. Automatically generated building-blocks used for the representation of Cartesian cells (left) and unstructured boundary surface mesh used for the representation of gridless points (right) based on the geometric definition of the configuration.

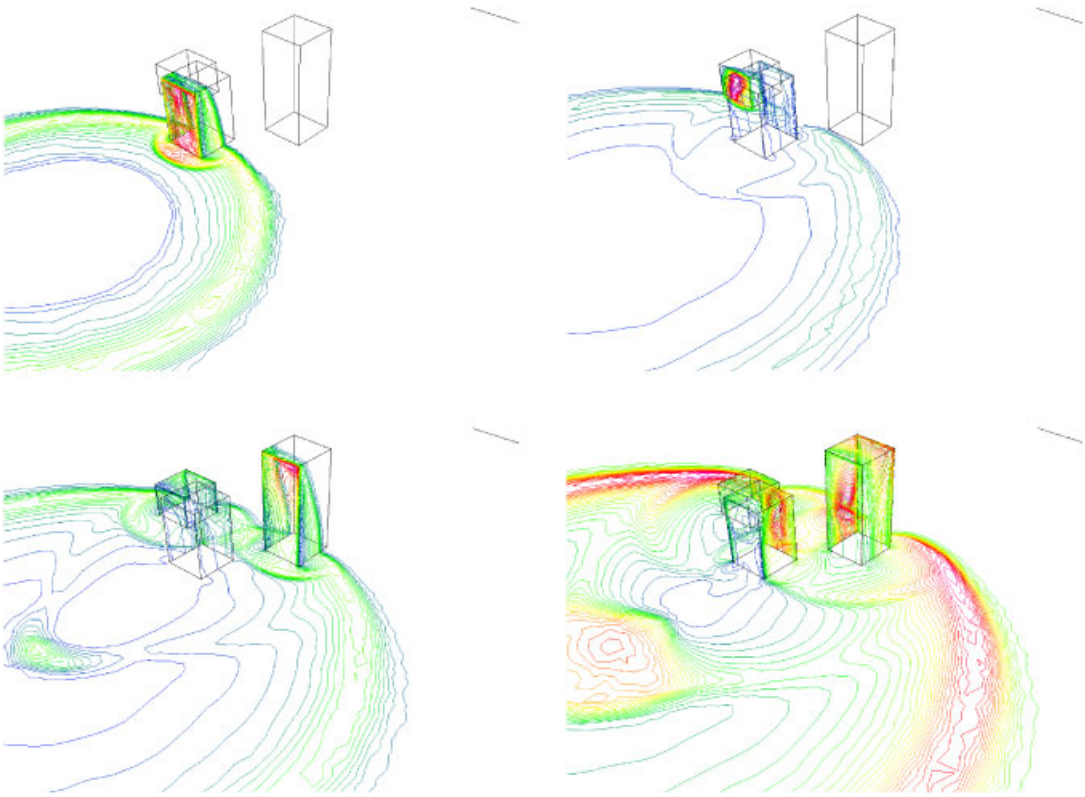

Figure 10. Computed pressure contours at different times for the simulation of a 50 kg TNT blast wave past a multi-building configuration using the hybrid method.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

5. CONCLUSIONS

A hybrid building-block Cartesian grid and gridless method has been developed for solving the unsteady compressible Euler equations. The developed method combines the efficiency of a Cartesian method and the flexibility of a gridless method for the complex geometries. Extensive numerical tests have been performed to demonstrate the accuracy, efficiency, robustness, and versatility of the proposed method. The numerical results obtained indicate that the use of this hybrid method leads to a significant improvement in performance over its unstructured grid counterpart without compromising solution accuracy, demonstrating the great potential and benefits of this hybrid method for the simulation of transient shock interaction problems. An overall speed-up factor from 6 to 12 and a saving in storage requirements up to one order of magnitude for a typical three-dimensional simulation in comparison with the unstructured grid method are obtained.

ACKNOWLEDGEMENTS

This research was partially sponsored by the Defense Threat Reduction Agency. Drs Darren Rice and Michael E. Giltrud served as the technical program monitor.

REFERENCES

1. Jameson A, Yoon S. Lower–upper implicit schemes with multiple grids for the Euler equations. *AIAA Journal* 1987; **25**:929–935.
2. Batten P, Leschziner MA, Goldberg UC. Average-state Jacobians and implicit methods for compressible viscous and turbulent flows. *Journal of Computational Physics* 1997; **137**:38–78.
3. Woodward PR, Colella P. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics* 1984; **54**:115–173.
4. Jameson A, Mavriplis D. Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. *AIAA Journal* 1986; **24**:611–618.
5. Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grids. *Journal of Computational Physics* 1998; **146**(2):664–690.
6. Luo H, Baum JD, Löhner R. High-Reynolds number viscous flow computations using an unstructured-grid method. *Journal of Aircraft* 2005; **42**(2):483–492.
7. Coirier WJ, Powell KG. An accuracy assessment of Cartesian mesh approaches for the Euler equations. *Journal of Computational Physics* 1995; **117**:121–131.
8. Berger MJ, LeVeque RJ. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. *AIAA Paper 1989-1930*, 1989.
9. Clarke DK, Salas MD, Hassan HA. Euler calculations for multielement airfoils using Cartesian grids. *AIAA Journal* 1986; **24**:353–358.
10. Batina JT. A gridless Euler/Navier–Stokes solution algorithm for complex aircraft applications. *AIAA Paper 1993-0333*, 1993.
11. Morinishi K. An implicit gridless type solver for the Navier–Stokes equations. *International Symposium on CFD*, Bremen, 1999.
12. Morinishi K. Effective accuracy and conservation consistency of gridless type solver. *Proceedings of the First International Conference on Computational Fluid Dynamics*, Kyoto, Japan, 10–14 July 2000.
13. Chandrashekar P, Deshpande SM. A new grid-free method for conservation laws. *Proceedings of the Second International Conference on Computational Fluid Dynamics*, Sydney, Australia, 15–19 July 2002.
14. Löhner R, Sacco S, Onate E, Idelsohn S. An finite point method for compressible flow. *International Journal of Numerical Methods in Engineering* 2002; **53**:1765–1779.
15. Sridar D, Balakrishnan N. An upwind finite difference scheme for meshless solvers. *Journal of Computational Physics* 2003; **189**:1–29.
16. Nakahashi K, Kim LS. Building-cube method for large-scale, high resolution flow computations. *AIAA Paper 2004-0423*, 2004.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

17. Koh EP, Tsai HM, Liu F. Euler solution using Cartesian grid with least squares technique. *AIAA Journal* 2005; **43**(2):246–255.
18. Kirshman DJ, Liu F. Gridless boundary condition treatment for a non-body-conforming mesh. *Journal of Computational Physics* 2004; **201**(1):119–147.
19. Kirshman DJ, Liu F. Cartesian grid solution of the Euler equations using a gridless boundary condition treatment. *AIAA Paper 2003-3974*, 2003.
20. Luo H, Baum JD, Löhner R. A hybrid Cartesian grid and gridless method for compressible flows. *Journal of Computational Physics* 2006; **214**:618–632.
21. Bonet J, Peraire J. An alternating digital tree (ADT) algorithm for geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering* 1991; **31**(1):1–17.
22. Toro EF, Spruce M, Speares W. Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves* 1994; **4**:25–34.
23. van Leer B. Towards the ultimate conservative difference scheme, II. Monotonicity and conservation combined in a second order scheme. *Journal of Computational Physics* 1974; **14**:361–370.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld