COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **2**, No. 6, pp. 1199-1219

Commun. Comput. Phys.
December 2007

A Cartesian Embedded Boundary Method for Hyperbolic Conservation Laws

Björn Sjögreen* and N. Anders Petersson

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA.

Received 4 January 2007; Accepted (in revised version) 11 April 2007

Available online 2 August 2007

Abstract. We develop an embedded boundary finite difference technique for solving the compressible two- or three-dimensional Euler equations in complex geometries on a Cartesian grid. The method is second order accurate with an explicit time step determined by the grid size away from the boundary. Slope limiters are used on the embedded boundary to avoid non-physical oscillations near shock waves. We show computed examples of supersonic flow past a cylinder and compare with results computed on a body fitted grid. Furthermore, we discuss the implementation of the method for thin geometries, and show computed examples of transonic flow past an airfoil.

AMS subject classifications: 65M06, 65M50, 76J20

Key words: Embedded boundary, hyperbolic conservation law, finite difference scheme, shock wave.

1 Introduction

This paper describes an embedded boundary finite difference method for solving the time-dependent compressible Euler equations external to a two- or three-dimensional object. In an embedded boundary approach the computational domain is discretized on a regular Cartesian grid and the boundary intersects the grid in an arbitrary fashion. Compared with boundary fitted structured or unstructured grid approaches, the biggest advantages of the embedded boundary method are the simplicity by which the grid can be generated as well as the efficiency and simplicity of the numerical method due to the Cartesian grid. The main challenge with the embedded boundary method is to accurately satisfy the boundary conditions while retaining stability of the resulting scheme. The proposed method is based on the second order accurate node-based discretization

*Corresponding author. *Email addresses:* sjogreen2@llnl.gov (B. Sjögreen), andersp@llnl.gov (N. A. Petersson)

technique for the wave equation in second order differential form subject to Dirichlet or Neumann boundary conditions [9–11]. Of particular interest for practical purposes is that these methods are explicit in time, but do not suffer from small-cell stiffness.

For the Euler equations, zero flux conditions are enforced on solid boundaries by combining Dirichlet and extrapolation conditions on different components of the solution. The Dirichlet components could in principle be approximated by the boundary condition in [9]. However, to avoid unphysical oscillations near shock waves we combine that technique with slope limiters to obtain a new zero flux boundary condition for embedded boundaries. The resulting method is formally second order accurate at the embedded boundary away from shock waves and smooth extrema, uses a finite difference formulation for the spatial discretization and is explicit in time, where the stability limit on the time step is based on the grid size away from the boundary.

Most previous work on embedded boundary methods for the compressible Euler equations are based on the finite volume formulation. At the embedded boundary, a naive finite volume discretization leads to an explicit time step that is limited by the smallest cell cut by the boundary. To overcome this so called "small cell problem", the method in [14] uses a modified non-conservative approximation at the boundary combined with a mass redistribution procedure after each time step [4] to achieve global conservation. Another way to overcome the small cell problem is provided by the h-box method, which is described in [1] and extended to multi-dimensional problems in [7,8]. An h-box is a larger control volume which is used for computing the flux on the side of a small cell. The one-dimensional h-box method is shown in [1] to be conservative, second order accurate, and having a time step which is not affected by small cut cells. For a simplified but less accurate approach, also see [2]. A third finite volume embedded boundary approach avoids the small cell problem by introducing uncut ghost cells around the boundary [5]. This method is second order accurate, but conservation has not been established. A fourth way of overcoming the small cell stiffness is provided by merging small cells at the embedded boundary with larger neighboring cells [3].

There is a large literature on embedded or immersed boundary methods for incompressible flow problems, see for example [12, 15] and the references therein. In these methods the immersed boundaries are often evolving material interfaces. Some of the boundary interpolation techniques are similar to what is used for compressible flows, but the incompressible problem is somewhat easier due to the absence of shock waves.

The remainder of the paper is organized as follows. The discretization of the Euler equations on a Cartesian grid is described in Section 2, and the discretization of the boundary conditions is developed in Section 3. In Section 4, we evaluate the performance of the method on several external flow problems. In the first numerical example we compare the accuracy of the computed solution at the embedded boundary with results obtained on a body fitted grid. Issues with the sharp trailing edge of an airfoil are discussed in Section 4.1 and the conservation properties of our method are investigated in Section 4.2. The embedded boundary discretization is extended to three space dimensions in Section 4.3 and conclusions are given in Section 5.

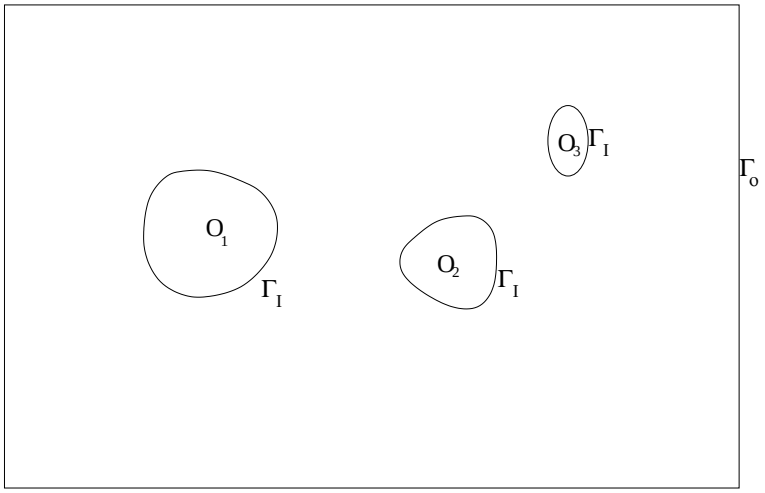

Figure 1: Computational domain is exterior of the objects O_i.

2 Numerical method

We consider a system of strongly hyperbolic conservation laws for $\mathbf{u}=\mathbf{u}(x,y,t)\in R^m$ on a two-dimensional domain Ω between an outer rectangular boundary Γ_O and one or more internal objects with boundary Γ_I, see Fig. 1,

$$\begin{aligned}&\mathbf{u}_t+\mathbf{F}(\mathbf{u})_x+\mathbf{G}(\mathbf{u})_y=\mathbf{0}, \quad (x,y)\in\Omega, \quad t>0,\\ &\mathbf{u}(x,y,0)=\mathbf{u}_0(x,y), \quad (x,y)\in\Omega.\end{aligned} \tag{2.1}$$

On the outer boundary, we have boundary conditions

$$\begin{aligned}&L_0\mathbf{u}(0,y,t)=\mathbf{c}_0(y,t), \quad L_1\mathbf{u}(l_x,y,t)=\mathbf{c}_1(y,t), \quad 0\le y\le l_y, \quad t>0,\\ &L_2\mathbf{u}(x,0,t)=\mathbf{c}_2(x,t), \quad L_3\mathbf{u}(x,l_y,t)=\mathbf{c}_3(x,t), \quad 0\le x\le l_x, \quad t>0,\end{aligned} \tag{2.2}$$

and on the boundary of the internal objects, we impose

$$L_{EB}\mathbf{u}(x,y,t)=\mathbf{c}_{EB}(x,y,t), \quad (x,y)\in\Gamma_I, \quad t>0.$$

Here, L_{EB} and L_i, $i=0,1,2,3$, are $m\times m$ matrices that can depend on $\boldsymbol{u}$. Inhomogeneous boundary data is prescribed through the vectors $\mathbf{c}_{EB}$ and $\mathbf{c}_i$, such that they belong to the range space of L_{EB} and L_i respectively. Let $\boldsymbol{n}=(n^{(x)},n^{(y)})^T$ be the unit normal of the boundary directed into Ω. Denote the flux normal to the boundary and its Jacobian matrix by

$$\mathbf{F}_\perp(\boldsymbol{u})=n^{(x)}\mathbf{F}(\boldsymbol{u})+n^{(y)}\mathbf{G}(\boldsymbol{u}), \quad A_\perp=\frac{\partial\mathbf{F}_\perp}{\partial\boldsymbol{u}},$$

respectively. Since (2.1) is strongly hyperbolic, all eigenvalues of $A_\perp$ are real and there is a complete set of eigenvectors,

$$A_\perp R=R\Lambda, \quad \Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_m).$$

Let there be r positive eigenvalues, $0 \le r \le m$. This number equals the number of ingoing characteristic variables, so the boundary condition matrices L_{EB} and L_i, i=0,1,2,3, should have rank r. Note that r depends on the solution, so it can change along the boundary. Let the free-stream state be $\boldsymbol{u}_\infty$. On the outer boundary Γ_O, we assign Dirichlet data for the ingoing characteristic variables by imposing

$$D^{in}R^{-1}\boldsymbol{u} = D^{in}R^{-1}\boldsymbol{u}_\infty, \quad \text{i.e.,} \quad L_i = D^{in}R^{-1}, \ \mathbf{c}_i = D^{in}R^{-1}\boldsymbol{u}_\infty, \ i=0,1,2,3. \tag{2.3}$$

Here, D^{in} is a diagonal $m \times m$ matrix with elements $D^{in}_{kk} = 1$, if $\lambda_k > 0$, otherwise $D^{in}_{kk} = 0$. Note that in general, all variables are not assigned boundary values on all boundaries.

The remainder of this section describes the numerical approximation of (2.1) and (2.2). The discretization at the internal boundary, Γ_I, is described in Section 3.

2.1 Approximation of the PDE

To solve (2.1) numerically, we first discretize the computational domain $[0,l_x]\times[0,l_y]$, on a Cartesian grid with grid points (x_i,y_j), $x_i = (i-1)\Delta x$, $i = 1,2,\cdots,M$, $y_j = (j-1)\Delta y$, $j = 1,2,\cdots,N$. The grid spacings $\Delta x = l_x/(M-1)$ and $\Delta y = l_y/(N-1)$ are constant and we define half-indices according to $x_{i+1/2} = x_i + \Delta x/2$. We discretize time on a grid $0 = t_0 < t_1 < \cdots < t_n$, with variable time-step $\Delta t_k > 0$ using the notation $t_k = t_{k-1} + \Delta t_k$.

We use the following notation for vector valued variables, their semi-discrete approximation at grid point (x_i,y_j), and their fully discrete approximation at that grid point at time t_n:

$$\mathbf{u}(x,y,t) = \begin{pmatrix} u^{(1)}(x,y,t) \\ u^{(2)}(x,y,t) \\ \vdots \\ u^{(m)}(x,y,t) \end{pmatrix}, \quad \mathbf{u}_{i,j}(t) = \begin{pmatrix} u^{(1)}_{i,j}(t) \\ u^{(2)}_{i,j}(t) \\ \vdots \\ u^{(m)}_{i,j}(t) \end{pmatrix}, \quad \mathbf{u}^n_{i,j} = \begin{pmatrix} u^{(1)n}_{i,j} \\ u^{(2)n}_{i,j} \\ \vdots \\ u^{(m)n}_{i,j} \end{pmatrix}.$$

The t dependency will be surpressed from the semi-discrete notation when the meaning is obvious.

The semi-discrete approximation of the hyperbolic system (2.1) is given by

$$\frac{d\mathbf{u}_{i,j}(t)}{dt} + \frac{\mathbf{a}_{i+1/2,j} - \mathbf{a}_{i-1/2,j}}{\Delta x} + \frac{\mathbf{b}_{i,j+1/2} - \mathbf{b}_{i,j-1/2}}{\Delta y} = \mathbf{0}, \tag{2.4}$$

for $i = 3,\cdots,M-2$, $j = 3,\cdots,N-2$. Here $\mathbf{a}_{i+1/2,j}$ and $\mathbf{b}_{i,j+1/2}$ are discrete flux functions corresponding to the continuous flux functions $\mathbf{F}(\mathbf{u})$ and $\mathbf{G}(\mathbf{u})$, respectively. We will use the five point second order accurate MUSCL scheme, given by

$$\mathbf{a}_{i+1/2,j} = \mathbf{h}(\mathbf{u}^R_{i+1/2,j}, \mathbf{u}^L_{i+1/2,j}), \tag{2.5}$$

where $\mathbf{h}(\mathbf{u},\mathbf{v})$ is the numerical flux function of the first order upwind method [16], with a standard entropy correction [6]. Furthermore, $\mathbf{u}^R_{i+1/2,j}$ and $\mathbf{u}^L_{i+1/2,j}$ are the right and

left side limits of the piecewise linear reconstruction of the solution, as $x \to x_{i+1/2}$. It is known that unphysical oscillations can occur when the conserved variables are used for the piecewise linear reconstruction. For example, the conserved variables ($\mathbf{u}$) for Euler's equations of compressible gas dynamics are density, momentum, and energy, but it has turned out in numerical experiments that better results are obtained when the reconstruction is done with primitive variables, i.e., density, velocity, and pressure ($\mathbf{v}$). For this reason, we make a transformation of variables, $\mathbf{v}=\mathbf{v}(\mathbf{u})$, at each grid point and define the piecewise linear reconstruction as

$$\mathbf{v}_r(x,y_j)=\mathbf{v}_{i,j}+\frac{x-x_i}{\Delta x}\mathbf{s}_{i,j}, \qquad x_{i-1/2}<x<x_{i+1/2}. \tag{2.6}$$

Here $\mathbf{s}_{i,j}$ is the slope of $\mathbf{v}$ in the x-direction at the point (i,j). This is a one-dimensional reconstruction, defined for $y=y_j$. The slopes are functions of the forward- and backward differences of $\mathbf{v}$, applied componentwise,

$$s_{i,j}^{(k)}=S(v_{i+1,j}^{(k)}-v_{i,j}^{(k)},v_{i,j}^{(k)}-v_{i-1,j}^{(k)}), \quad k=1,2,\cdots,m,$$

where $S(x,y)$ is a limiter function. In the present work, we use the minmod function

$$S_{mm}(x,y)=\begin{cases}0, & xy<0,\\ x, & xy>0,|x|<|y|,\\ y, & \text{otherwise},\end{cases} \tag{2.7}$$

or the van Albada limiter

$$S_{va}(x,y)=((x^2+\epsilon)y+(y^2+\epsilon)x)/(x^2+y^2+2\epsilon), \tag{2.8}$$

where ϵ is a very small positive number, introduced to prevent division by zero. The limiter averages the forward- and backward differences, but puts more weight on the difference that has the smallest absolute value, thereby reducing unphysical oscillations near discontinuities. It can be proven that both above limiters give the TVD property for scalar conservation laws. Note that the slope $\mathbf{s}_{i,j}$ depends on $\mathbf{v}$ at $(i-1,j)$, (i,j), and $(i+1,j)$, therefore the slopes are defined for $i=2,\cdots,M-1$ and $j=1,\cdots,N$.

The left- and right-limit values at $x_{i+1/2}$ are obtained as

$$\mathbf{v}_{i+1/2,j}^R=\mathbf{v}_{i+1,j}-\mathbf{s}_{i+1,j}/2, \qquad \mathbf{v}_{i+1/2,j}^L=\mathbf{v}_{i,j}+\mathbf{s}_{i,j}/2, \tag{2.9}$$

for $i=2,\cdots,M-2$ and $j=1,\cdots,N$. The corresponding conserved variables, $\mathbf{u}_{i+1/2,j}^R$ and $\mathbf{u}_{i+1/2,j}^L$ are obtained by applying the inverse of the variable transformation $\mathbf{v}=\mathbf{v}(\mathbf{u})$, thereby determining the numerical flux (2.5).

The discrete flux function in the y-direction, $\mathbf{b}_{i,j+1/2}$, is constructed in a corresponding way.

2.2 Approximation near the boundary

We next describe the discrete boundary conditions for the boundary $x=0$ (i.e., $i=1$); the other sides are handled in a corresponding way. Boundary conditions are needed for $\mathbf{u}_{1,j}$ and $\mathbf{u}_{2,j}$, $j=1,\cdots,N$, since (2.4) is a five point scheme. To compute $\mathbf{u}_{2,j}$, we define the slope at $i=1$ by extrapolation to first or second order,

$$\mathbf{s}_{1,j}=\mathbf{s}_{2,j} \quad \text{or} \quad \mathbf{s}_{1,j}=2\mathbf{s}_{2,j}-\mathbf{s}_{3,j}, \quad j=1,\cdots,N.$$

Note that this is an extrapolation of already limited slopes, so the introduction of new oscillations should be minimal. Once $\mathbf{s}_{1,j}$ is defined, the numerical flux function $\mathbf{a}_{3/2,j}$, can be evaluated for $j=1,\cdots,N$, and we can apply the numerical scheme (2.4) also at $i=2$. We implement this boundary slope extrapolation as a part of the interior scheme (2.4). The solution at $i=2$, $j=2,\cdots,N-1$ is then computed by (2.4) and the scheme effectively becomes a three point scheme at the boundary. At the points $(2,2)$ and $(2,N-1)$, the procedure described above is applied in both the x- and y-directions.

It remains to impose values at the outermost points $i=1$, $j=1,\cdots,N$. If we order the eigenvectors in R such that the first r eigenvalues of $A_\perp$ are positive, we can decompose the characteristic variables according to

$$\begin{pmatrix} \widetilde{\boldsymbol{u}}^I \\ \widetilde{\boldsymbol{u}}^{II} \end{pmatrix} = R^{-1}\boldsymbol{u}.$$

The ingoing characteristic variables correspond to $\widetilde{\boldsymbol{u}}^I$ and are determined by the physical boundary conditions (2.3), which we discretize as

$$D^{in}R^{-1}\boldsymbol{u}_{1,j}=D^{in}R^{-1}\boldsymbol{u}_\infty, \quad j=1,\cdots,N.$$

The Jacobian matrix of the flux normal to the boundary, its eigenvalues and eigenvectors are evaluated at $(\boldsymbol{u}_{2,j})$. A numerical boundary condition for the remaining components of $\boldsymbol{u}$ is provided by extrapolation,

$$\tilde{\boldsymbol{u}}^{II}_{1,j}=2\tilde{\boldsymbol{u}}^{II}_{2,j}-\tilde{\boldsymbol{u}}^{II}_{3,j}, \quad j=1,\cdots,N. \tag{2.10}$$

The extrapolation can introduce unphysical states when the solution is discontinuous near the boundary. For such problems we replace (2.10) by

$$\tilde{\boldsymbol{u}}^{II}_{1,j}=\tilde{\boldsymbol{u}}^{II}_{2,j}-S_{mm}(\tilde{\boldsymbol{u}}^{II}_{3,j}-\tilde{\boldsymbol{u}}^{II}_{2,j},\tilde{\boldsymbol{u}}^{II}_{4,j}-\tilde{\boldsymbol{u}}^{II}_{3,j}), \tag{2.11}$$

where $S_{mm}(x,y)$ is the minmod limiter given by (2.7).

2.3 Time discretization

We use a two stage, second order accurate, Runge-Kutta method to integrate (2.4) in time. We introduce a constant CFL-number, c_{fl}. At each time t_n we choose a time step,

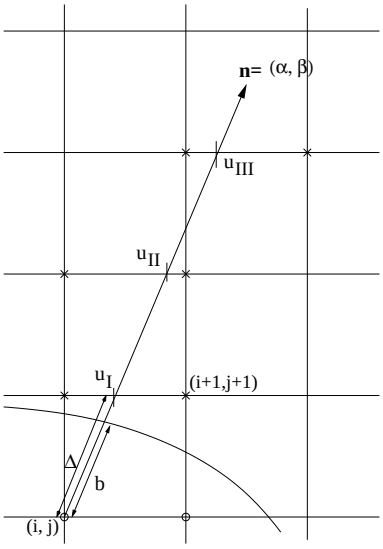

Figure 2: Boundary interpolation.

Δt_n, according to

$$\Delta t_n \max_{i,j} \left(\frac{Sp(A(\mathbf{u}_{i,j}^n))}{\Delta x} + \frac{Sp(B(\mathbf{u}_{i,j}^n))}{\Delta y} \right) = c_{fl},$$

where A and B are the Jacobian matrices of the flux functions $f(\boldsymbol{u})$ and $g(\boldsymbol{u})$, respectively, and $Sp(A)$ denotes the spectral radius of the matrix A.

3 Embedded boundary

Internal boundaries are embedded into the Cartesian grid and the boundary Γ_I is allowed to cut through the grid in an arbitrary fashion, as long as it is resolved on the grid. Fig. 2 shows a close up of a few grid points near an embedded boundary.

This section describes how to impose Dirichlet and extrapolated boundary conditions on an embedded boundary. We only consider scalar problems. For systems the scalar procedure is applied componentwise, as described in Section 2.2. For a scalar dependent variable $u(x,y,t)$, we denote the numerical solution at grid point (x_i,y_j) by $u_{i,j}(t)$, where the t dependence will be surpressed when possible. We assume a three point stencil, so that the internal scheme updates all points except the points just outside the boundary, the ghost points (one ghost point is indicated by a circle in Fig. 2). Note that the MUSCL scheme is a five point stencil, but the previously described slope extrapolation effectively reduces the scheme to a three-point stencil near the boundary.

Assume that the points in the interior of the domain, above the boundary curve in Fig. 2, have been updated using the interior difference scheme. To define the value of u at the ghost point, we proceed as follows. Let the index of the ghost point be (i,j). Let

$\mathbf{n}=(\alpha,\beta)$ be the normal to the boundary that goes through the point (i,j). The value u_I in Fig. 2 is obtained by linear interpolation along the grid line y_{j+1} between the values at the nearest left and right neighboring grid points. A corresponding linear interpolation procedure is used along the grid lines y_{j+2} and y_{j+3}, to define the values u_{II} and u_{III}. For the case shown in Fig. 2, we obtain

$$\begin{aligned} u_I &= w_1 u_{i,j+1} + (1-w_1) u_{i+1,j+1}, \\ u_{II} &= w_2 u_{i,j+2} + (1-w_2) u_{i+1,j+2}, \\ u_{III} &= w_3 u_{i+1,j+3} + (1-w_3) u_{i+2,j+3}, \end{aligned} \tag{3.1}$$

where the weights $0 \le w_k \le 1$, $k=1,2,3$, depend on where the normal intersects the horizontal grid lines. When the normal has a positive y-component and the angle between the normal and the x-axis is between $\pi/4$ and $\pi/2$, the normal will always intersect the grid line $y=y_{j+1}$ between x_i and x_{i+1}. There are two different cases for u_{II} (between $(i,j+2)$ and $(i+1,j+2)$ or between $(i+1,j+2)$ and $(i+2,j+2)$) and, similarly, three different cases for u_{III}. For other directions of the normal, the boundary interpolation scheme is obtained as mirror images and/or reflections of Fig. 2. Since the three interpolated values are obtained by linear interpolation, they can not cause any new maxima or minima in the presence of discontinuities.

3.1 Dirichlet boundary condition

Consider imposing the Dirichlet condition $u(x^\Gamma,y^\Gamma)=g(x^\Gamma,y^\Gamma)$ for (x^Γ,y^Γ) on Γ_i, where $g(x,y)$ is given data on the boundary. We obtain u_I, u_{II}, and u_{III} by interpolation as described above. Let (x_i^Γ,y_i^Γ) be the point where the normal intersects the boundary. Denote the distance between the ghost point and the boundary by b and let the distance between the ghost point and the grid line $y=y_{j+1}$ along the normal be Δ (see Fig. 2). By interpolating along the normal, we define values u_{b1} and u_{b2} at points $b+\Delta$ and $b+2\Delta$ from the ghost point, respectively. Let

$$u_{b1}=(b/\Delta)u_{II}+(1-b/\Delta)u_I, \quad u_{b2}=(b/\Delta)u_{III}+(1-b/\Delta)u_{II}.$$

Note that the points (x_i^Γ,y_j^Γ), (x_{b1},y_{b1}), and (x_{b2},y_{b2}) are equidistant by construction. Next we define a limited boundary slope,

$$s_\Gamma = S_{mm}(u_{b1}-g(x_i^\Gamma,y_j^\Gamma),u_{b2}-u_{b1}),$$

where $S_{mm}(x,y)$ is the minmod limiter given by (2.7). We can interpret s_Γ as the slope at the point $b+\Delta$ away from the ghost point, extrapolated to (x_i^Γ,y_i^Γ). The ghost point value is finally defined by extrapolating the boundary value using the limited boundary slope,

$$u_{i,j}=g(x_i^\Gamma,y_j^\Gamma)-\frac{b}{\Delta}s_\Gamma. \tag{3.2}$$

Remark 3.1. The above construction is always well-defined, since $h \leq \Delta \leq \sqrt{2}h$. In many finite volume methods, computation of the value at the boundary point includes dividing by a coefficient that can become arbitrarily small when the cut cell becomes small. This leads to a severe time step restriction when using an explicit method in time. Δ in (3.2) is bounded away from zero even when the cut cell size becomes very small, therefore we do not expect any time additional step restrictions from the embedded boundary in the present approach. This is verified by the numerical experiments in Section 4 were in all cases we successfully used the time step determined by the standard CFL condition in the interior of the domain. See [9] for an analysis that shows stability without additional time step restrictions from the boundary for a similar boundary procedure applied to the wave equation.

3.2 Extrapolation boundary condition

For variables that do not have a physical boundary condition, we use extrapolation as numerical boundary condition. Since there is no condition to be imposed, we can extrapolate without using any information about the boundary location. We use (2.11) along the normal to obtain

$$u_{i,j} = u_I - S(u_{III} - u_{II}, u_{II} - u_I). \tag{3.3}$$

In practice, it has turned out that the extrapolation (3.3) sometimes can give negative densities or pressures. This can for example occur for Euler's equations in wake regions behind objects in high speed fluid flow. In our solver, we first try (3.3), but if that formula takes the density or the pressure below zero at the ghost point, we instead use

$$u_{i,j} = u_I. \tag{3.4}$$

The boundary extrapolation (3.4) is only first order accurate, but has turned out to be very robust. This reduction of accuracy when extrapolating near vacuum only takes place at a small number of grid points. This accuracy reduction at a few boundary points should not lead to any reduction of accuracy in the L^1 or L^2 norms even when the solution is smooth, because there are already some points in the domain, the smooth extrema, where the TVD scheme reduces the formal order of accuracy from second to first.

4 Numerical experiments

We start by considering the Euler equations of compressible gas dynamics. The equations are

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{pmatrix}_y = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{pmatrix}_z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

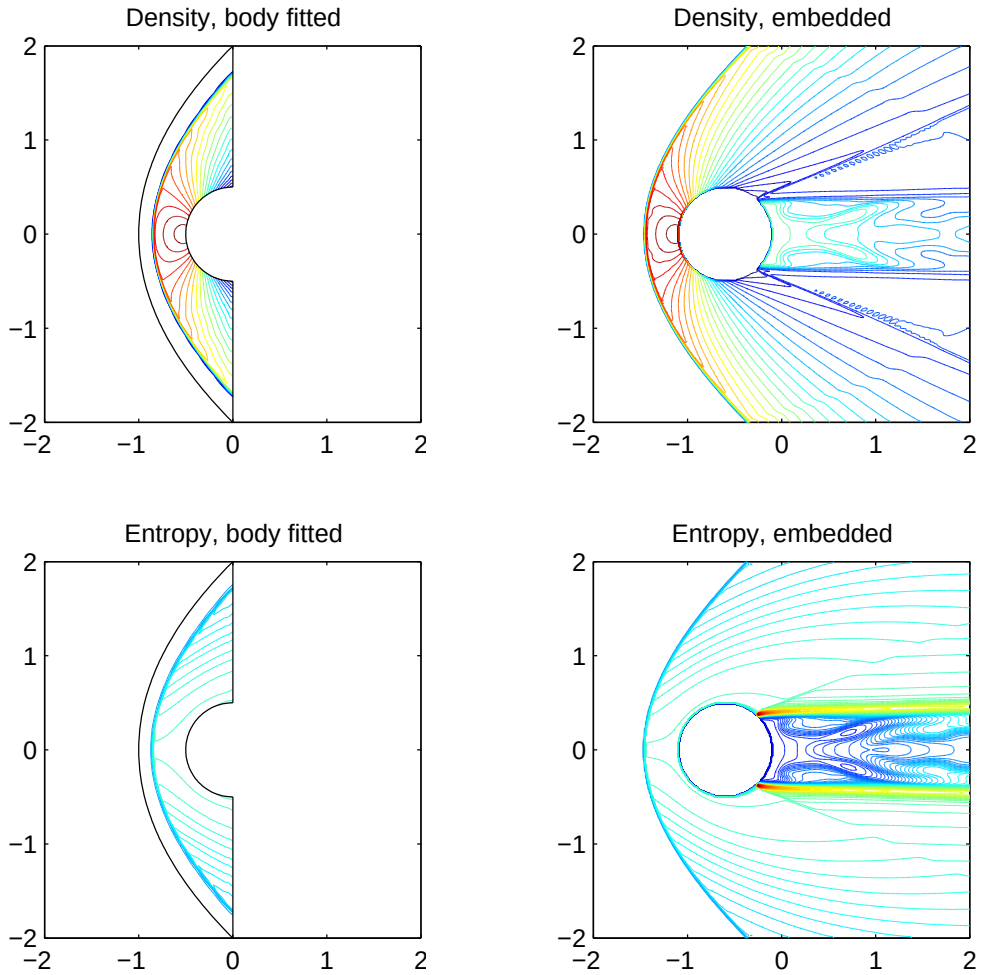

Figure 3: Mach 3 flow. Body fitted grid (left), embedded boundary (right). Density contours (upper) and entropy contours (lower).

where ρ is the density, u is the velocity in the x-direction, v is the velocity in the y-direction, w is the velocity in the z-direction, and e is the total energy. The pressure is defined by the ideal gas gamma law

$$p=(\gamma-1)\left(e-\frac{1}{2}\rho(u^2+v^2+w^2)\right).$$

On solid wall boundaries, we set the normal velocity to zero. In the notation of (2.2) this means taking $r=1$, $\mathbf{c}_i=0$, and defining the boundary operator to be a rotation which selects the component of the momentum normal to the wall. In this case L_i is independent of $\mathbf{u}$.

We compute supersonic two-dimensional compressible flow past a disk of radius 0.5.

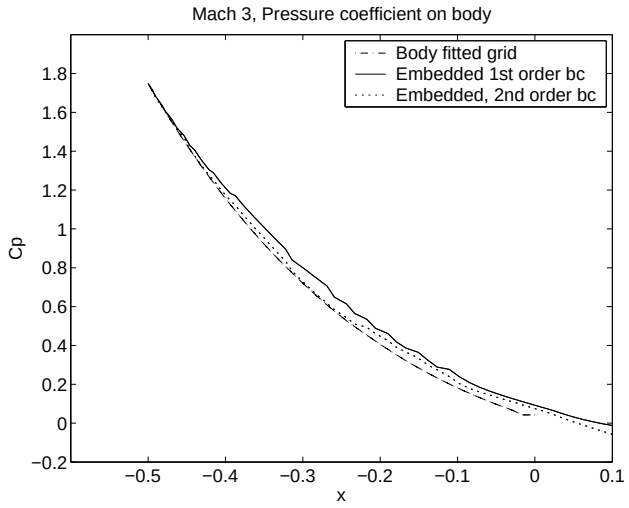

Figure 4: Mach 3 flow. Pressure coefficient on the body. Body fitted grid (dash-dot), embedded first order bc (solid), and embedded second order bc (dot).

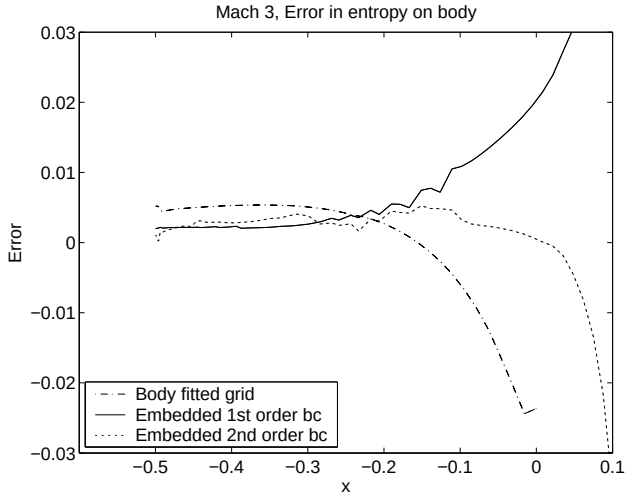

Figure 5: Mach 3 flow. Error in entropy on the body. Body fitted grid (dash-dot), embedded first order bc (solid), and embedded second order bc (dot).

The free stream Mach number is 3 and $\gamma = 1.4$. We solve the problem first with a body fitted grid of 101×51 points. This is a standard computation, see for example [17,18]. The body fitted grid only covers the front part of the disk. Secondly, we solve the same problem using embedded boundary method on the domain $[-2,2] \times [-2,2]$. For the embedded boundary method, 305×305 grid points are used. With these resolutions approximately the same number of grid points are placed on the body with both methods.

Fig. 3 shows the density contours for the body fitted grid (upper left), the density

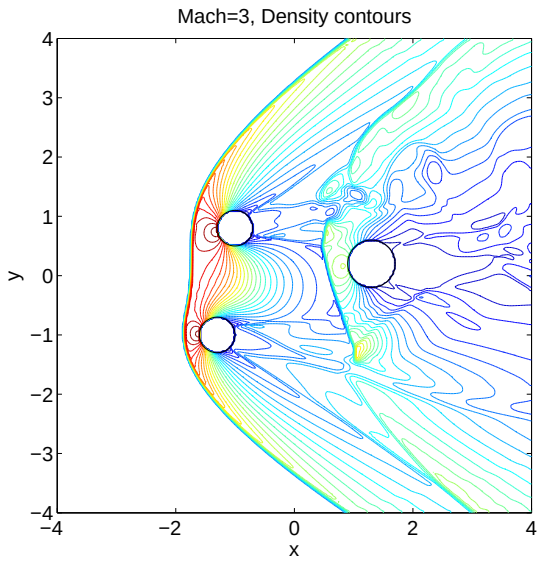

Figure 6: Mach 3 flow, three disks, density contours.

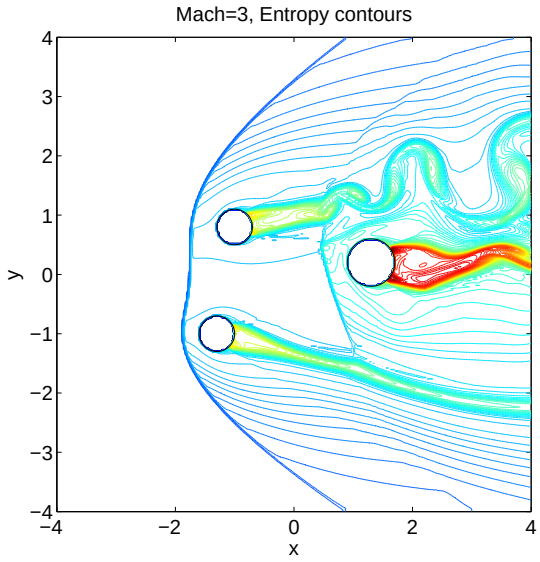

Figure 7: Mach 3 flow, three disks, entropy contours.

contours the embedded boundary method (upper right). Also shown are entropy contours for the body fitted grid (lower left) and for the embedded boundary method (lower right).

The flow in the wake behind the disk is determined by numerical viscosity, and has no physical relevance. Note that the flow is supersonic at the outflow boundaries for the body fitted grid computation, so the upstream flow field is not affected by having a smaller domain.

In Fig. 4 we show the pressure coefficient along the body as function of the x-coordinate, computed for the body fitted grid, the embedded boundary with the low order extrapolated boundary conditions (3.4) everywhere, and the embedded boundary with 2nd order extrapolated boundary conditions (3.3). The Dirichlet condition for the normal velocity was imposed according to (3.2) in both cases. The boundary values for the embedded boundary method are computed at the points where normals from ghost points intersect the boundary, as described by the interpolation scheme in Fig. 2. The curves show good agreement.

For compressible flows, the entropy is constant along streamlines. The entropy on the body can be computed analytically, by using the Rankine-Hugoniot conditions for the streamline normal to the bow shock at $y=0$. Fig. 5 displays the error in entropy along the body as function of the x-coordinate. The same three methods as were shown in Fig. 4 are also shown in Fig. 5. Near the stagnation point, both embedded boundary methods are more accurate than the body fitted grid method. The second order embedded method stays more accurate than the body fitted method along the entire body.

As a second example, in Fig. 6 we display two-dimensional flow past three disks

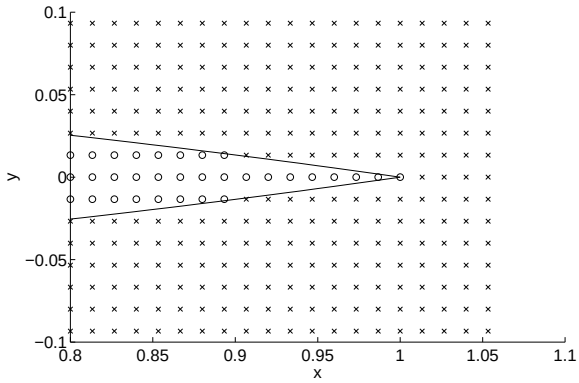

Figure 8: Trailing edge of the NACA0012 airfoil. Grid line in middle.

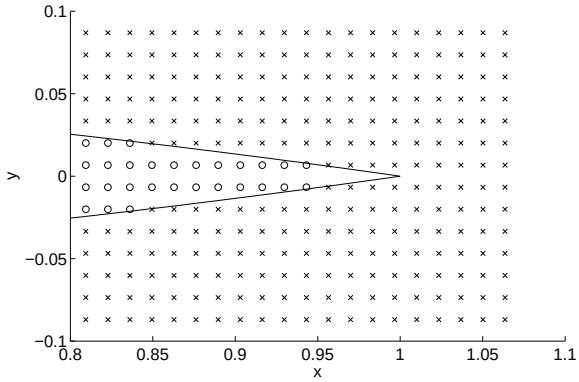

Figure 9: Trailing edge of the NACA0012 airfoil. No grid line in middle.

at Mach 3, computed by the embedded boundary method. The disks are centered at $(-1.3,-1.0)$, $(-1.0,0.8)$, and $(1.3,0.2)$, with radii 0.3,0.3, and 0.4 respectively. The computational domain is $[-4,4]\times[-4,4]$ and 305×305 grid points were used. In Fig. 7, the corresponding entropy contour lines are shown. This example demonstrates that the embedded grid method easily handles more than one object in the computational domain.

4.1 Flow past a NACA0012 airfoil

The boundary of the NACA0012 airfoil has an unsmooth normal at the trailing edge, which complicates the definition of the embedded boundary discretization. One difficulty is that the thickness of the airfoil becomes very small, so that near the trailing edge, there can be ghost points which serve as a ghost point for both the upper and the lower sides. This occurs if the trailing edge of the airfoil is centered around one horizontal grid line, see Fig. 8. The inside points are marked by 'o' and the points in the computational domain are marked by 'x'. In $0.9<x<1$ there is only one layer of points inside the airfoil. To handle this case, we let these special ghost points have two values and two normals,

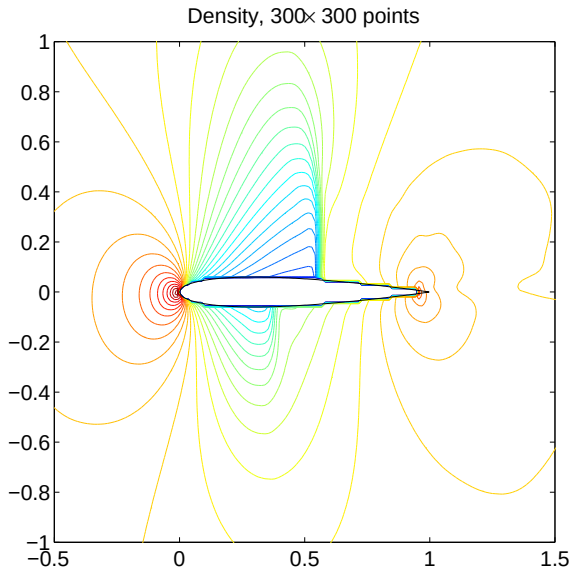

Figure 10: NACA0012 airfoil, Mach 0.8, angle 1.25°. Density contours.

one normal to the upper boundary curve, the other normal to the lower boundary curve. When computing above the airfoil, the upper value of the ghost point, and the upward normal is used. When computing below the airfoil, the lower value of the ghost point and downward normal are used. Imposing boundary data is straightforward, defining one value by interpolating from above and the other value by interpolating from below. The very last point serves as ghost point for three directions, and the above technique is extended to a three-valued ghost point.

Although this procedure is conceptually straightforward, implementation in a computer code requires some organization. We maintain a list of multiple valued ghost points with one list entry for each value. For the example in Fig. 8, the table would contain 17 entries. In order to define the table, special logic is used to make sure the two different normals belonging to the same ghost point are found. Each time the interior difference scheme accesses a point which could be a ghost point, the point is looked up in the multiple ghost point table. If it is found there, its value is supplied from the table, otherwise the value in the standard array is used.

To test the method, we computed transonic flow with Mach number 0.8 and angle of attack 1.25° around the NACA0012 airfoil. This is a test case which is often used, see for example [5]. The computational domain is $[-2,2]\times[-2,2]$. In Fig. 10, we show density contours of a solution computed with 300×300 grid points, and in Fig. 11, we display the pressure coefficient (C_p) on the boundary.

In this computation the number of grid points was even. A close up of the trailing edge is displayed in Fig. 9. There is no grid line exactly in the middle of the airfoil, therefore there are no special ghost points with multiple values. If instead the computation

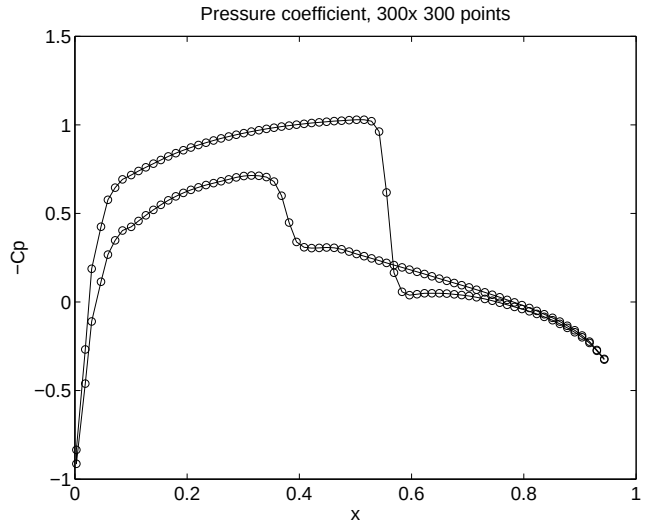

Figure 11: Pressure coefficient.

is done with 301×301 points, the situation is as displayed in Fig. 8. For this number of points, a few of the ghost points are handled as special multiply valued points. In Fig. 12, we show a comparison between the C_p curves for the computations with 300×300 points and 301×301 points. The difference is fairly significant, and because the geometries differs by $\mathcal{O}(h)$, it is not unreasonable to expect that the difference between the C_p curves is first order in the grid spacing.

To further investigate the convergence properties, we display in Fig. 13 the C_p curves for grids with 300×300, 600×600, and 1200×1200 grid points. The curves converge as the grid is refined, but convergence is slow at the shock on the upper side on the airfoil. Similarly, we show in Fig. 14 the C_p curves for grids with 301×301, 601×601, and 1201×1201 grid points. Comparing Figs. 13 and 14, we conclude that the convergence to the fine grid solution appears to be better when the number of points is odd. Fig. 15 shows a comparison between the C_p curves on the grids with 1200×1200 and 1201×1201 grid points. The even and odd cases seem to converge to the same solution, except for a minor discrepancy near the shock at the lower side of the airfoil.

4.2 Conservation properties

The proposed method is not guaranteed to be conservative at the embedded boundary. To investigate the size of the possible loss of mass from the embedded boundary, we here compute a steady subsonic flow in a channel with an elliptic obstacle. Fig. 16 shows density contours of the steady flow. The domain is of size $[-3,3]\times[-2,2]$ and the ellipse have the axis lengths 0.5 and 0.3 in the x- and y-directions respectively. Ideal gas flow enters at the left boundary with speed Mach 0.5. The upper and lower boundaries are solid walls where slip boundary conditions are imposed. Conservation is measured by

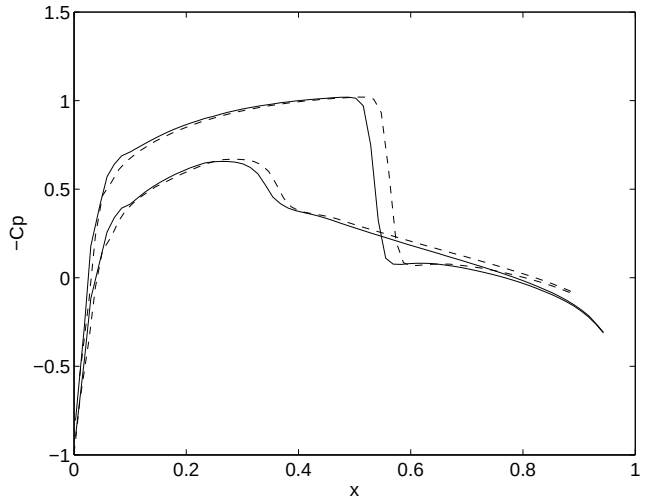

Figure 12: C_p with 300×300 points (solid) and with 301×301 points (dash).

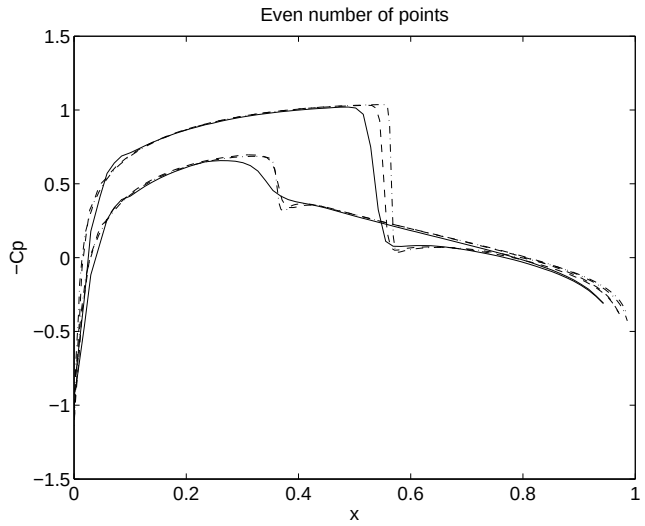

Figure 13: C_p with 300×300 points (solid), with 600×600 points (dash), and with 1200×1200 points (dash-dot).

comparing the total mass flux across the left inflow with the total mass flux across the right outflow boundary. The mass flux over the grid line x_i is approximated by the sum

$$F_i = \sum_{j=1}^{N-1} h\rho_{i,j+1/2}u_{i,j+1/2},$$

where N is the number of grid points in the j direction, and $\rho_{i,j+1/2}=(\rho_{i,j+1}+\rho_{i,j})/2$. Ideally, the influx F_1 should equal the outflux F_M, where M is the number of grid points in

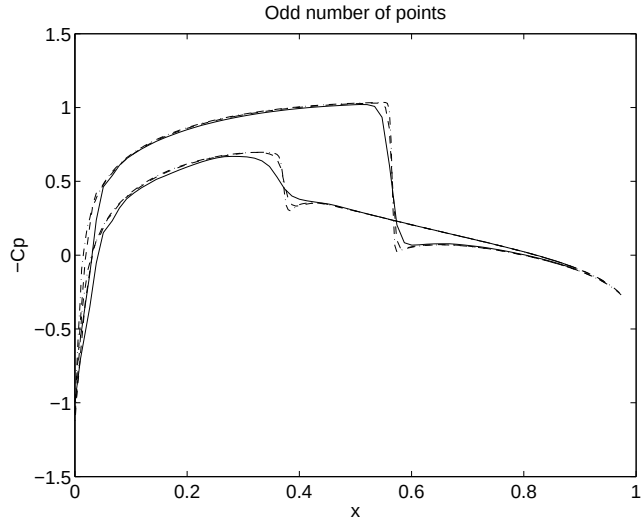

Figure 14: C_p with 301×301 points (solid), with 601×601 points (dash), and with 1201×1201 points (dash-dot).

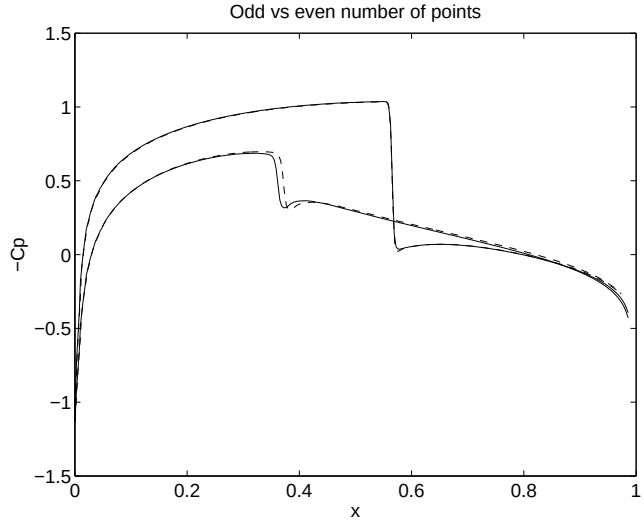

Figure 15: C_p with 1200×1200 points (solid) and with 1201×1201 points (dash).

the x-direction. In Table 1, we show the mass loss $\Delta F = F_M - F_1$ as function of the number of grid points. From Table 1, we conclude that the mass loss goes to zero as the grid is refined with a rate $\mathcal{O}(h^{1.6})$. The limiters in the boundary interpolation scheme locally reduces the order of the truncation error at some points on the embedded boundary. To investigate whether this is the reason for not getting full second order decrease, we display in the last column of Table 1 the mass losses without using the boundary limiters. The convergence rate is improved to $\mathcal{O}(h^{1.8})$ between the two finest grids. It is not prac-

Table 1: Mass loss due to non-conservative embedded boundary procedure.

M	ΔF	Relative loss	ΔF_{nolim}	Relative loss
301	0.015	0.54%	0.012	0.43%
601	0.0053	0.19%	0.0035	0.12%
1201	0.0018	0.064%	0.0010	0.036%

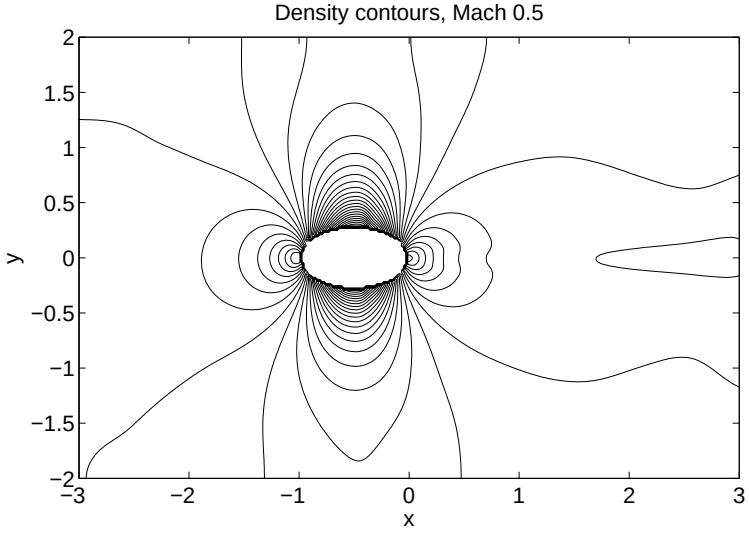

Figure 16: Mach 0.5 flow in a channel with obstacle.

tical, however, to solve without limiters. In the present computation we started from the converged steady solution obtained with limiters. For any other initial data we tried the computation became unstable due to discontinuities in the transient phase.

4.3 Three space dimensions

We here outline the extension of the embedded boundary technique to three space dimensions. First, we compute the normal to the boundary through each ghost point. This computation depends on the representation of the geometry. In the example below, we compute fluid flow past a sphere. In this case, analytical formulas can be used for calculating normals and boundary distances.

The sphere is simple enough that the computation of normals can be done directly by analytical formulas.

Given the boundary normals, we define values u_I,u_{II}, and u_{III} in the same way as in (3.1). In three dimensions these values are determined as weighted averages of the four neighbors of the intersection point between the normal and a coordinate plane. In Fig. 2, the normal can intersect the grid line $y = y_{j+2}$ in two different cells, and there are

three different possibilities for the intersection of $y=y_{j+3}$. In three dimensions the corresponding number of possible intersections are four in the $y=y_{j+2}$ grid plane and nine in the $y=y_{j+3}$ grid plane. Nevertheless, it is not difficult to implement these boundary conditions. Given the normal $\mathbf{n}=(n_x \;\; n_y \;\; n_z)$, we determine the element with the largest absolute value. Let us assume that $|n_x|\geq|n_y|$ and $|n_x|\geq|n_z|$. The other two cases, $|n_y|$ largest and $|n_z|$ largest, are treated with obvious changes.

The three dimensional version of the interpolation (3.1) can be written

$$\begin{aligned} u_m = &(1-\xi_m)(1-\eta_m)u_{i_m,j_m,k_m}+\xi_m(1-\eta_m)u_{i_m,j_m+s,k_m} \\ &+(1-\xi_m)\eta_m u_{i_m,j_m,k_m+s}+\xi_m\eta_m u_{i_m,j_m+s,k_m+s}, \quad m=1,2,3, \end{aligned} \tag{4.1}$$

where we set

$$u^I=u_1, \quad u^{II}=u_2, \quad u^{III}=u_3.$$

We have introduced

$$s=sign(n_x), \quad \xi=|n_y/n_x|, \quad \eta=|n_z/n_x|,$$

and

$$\xi_m=m\xi-\lfloor m\xi\rfloor, \quad \eta_m=m\eta-\lfloor m\eta\rfloor, \tag{4.2}$$

$$i_m=i+sm, \quad j_m=j+s\lfloor m\xi\rfloor, \quad k_m=k+s\lfloor m\eta\rfloor, \quad m=1,2,3, \tag{4.3}$$

where $\lfloor x\rfloor$ denotes the integer part of x.

Once u^I, u^{II}, and u^{III} are known, the remaining part of the boundary formulas is one dimensional, and we use (3.2) and (3.3) as previously.

We show in Fig. 17 the density distribution for the steady Mach 3 flow past a sphere. The sphere has radius 0.25 and is embedded in a Cartesian domain of size $[-1,1]\times[-1,1]\times[-1,1]$, discretized on a grid with $120\times120\times120$ grid points. The computation was run time accurately until the bow shock moved into its correct position.

5 Conclusions

We have presented a finite difference embedded boundary method for computing inviscid compressible flows around complex geometries. The method is second order accurate when the flow is smooth, easy to implement for two- and three-dimensional problems, and the explicit time-stepping is stable with a time step determined by the grid size away from the boundary.

Some work has been performed to generalize our method to handle viscous compressible flows [13], and a new embedded boundary compressible Navier-Stokes solver is under development. A challenge for embedded boundary methods is to resolve boundary layers in viscous flows when the flow exhibits a much smaller length scale normal to the boundary than in the tangential directions. A Cartesian mesh which is locally refined

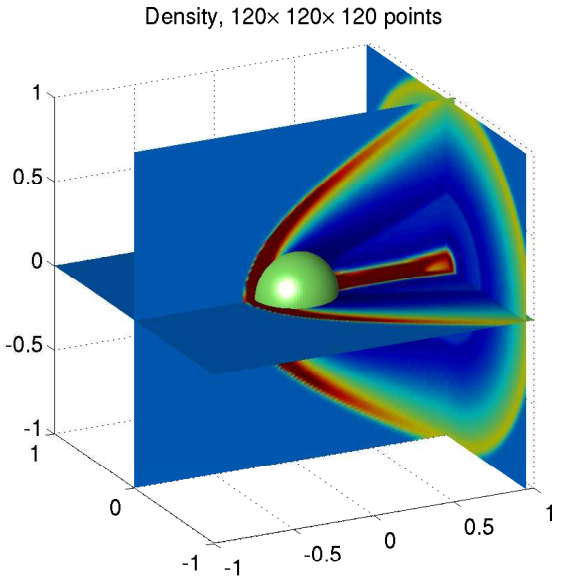

Figure 17: Mach 3 flow past a sphere. Density. $120\times120\times120$ grid points.

near the boundary can alleviate the resolution requirements, but more work is needed to handle mesh refinement boundaries intersecting the embedded boundary. To accurately and efficiently capture turbulent phenomena in Navier-Stokes flows, we would also like to increase the accuracy of the method beyond second order.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] M. J. Berger, C. Helzel and R. J. LeVeque, *H*-box methods for the approximation of hyperbolic conservation laws on irregular grids, SIAM J. Numer. Anal., 41 (2003), 893-918.
[2] D. Calhoun and R. J. LeVeque, A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries, J. Comput. Phys., 157 (2000), 143-180.
[3] W. J. Coirier and K. G. Powell, An accuracy assessment of Cartesian-mesh approaches for the Euler equations, J. Comput. Phys., 117 (1995), 121-131.
[4] P. Colella, D. T. Graves, B. J. Keen and D. Modian, A Cartesian grid embedded boundary method for hyperbolic conservation laws, J. Comput. Phys., 211 (2006), 347-366.
[5] H. Forrer and R. Jeltsch, A higher-order boundary treatment for Cartesian-grid methods, J. Comput. Phys., 140 (1998), 259-277.

[6] A. Harten, High resolution schemes for hyperbolic conservation laws, J. Comput. Phys., 49 (1983), 357-393.
[7] C. Helzel, Accurate methods for hyperbolic problems on embedded boundary grids, in: F. Asakura, H. Aiso, S. Kawashima, A. Matsumura, S. Nishibata and K. Nishihara (Eds.), Hyperbolic Problems, Theory, Numerics and Applications, Yokohama Publishers, 2006.
[8] C. Helzel, M. J. Berger and R. J. LeVeque, A high-resolution rotated grid method for conservation laws with embedded geometries, SIAM J. Sci. Comput., 26 (2005), 785-809.
[9] H.-O. Kreiss and N. A. Petersson, A second order accurate embedded boundary method for the wave equation with Dirichlet data, SIAM J. Sci. Comput., 27 (2006), 1141-1167.
[10] H.-O. Kreiss, N. A. Petersson and J. Yström, Difference approximations for the second order wave equation, SIAM J. Numer. Anal., 40 (2002), 1940-1967.
[11] H.-O. Kreiss, N. A. Petersson and J. Yström, Difference approximations of the Neumann problem for the second order wave equation, SIAM J. Numer. Anal., 42 (2004), 1292-1323.
[12] R. Mittal and G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid. Mech., 37 (2005), 239-261.
[13] O. K. Olsen, Embedded boundary method for Navier-Stokes equations, Report TRITA-NA-E05048, Department of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden, 2005.
[14] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield and M. L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, J. Comput. Phys., 120 (1995), 278-304.
[15] C. S. Peskin, The immersed boundary method, Acta Numerica, 11 (2002), 479-517.
[16] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys., 43 (1981), 357-372.
[17] B. Sjögreen, Iterative methods for stationary solutions to the steady state Navier-Stokes equations, Comput. Fluids, 21 (1992), 627-645.
[18] B. Sjögreen, High order centered difference methods for the compressible Navier-Stokes equations, J. Comput. Phys., 117 (1995), 67-78.