Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 211 (2006) 492–512
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

High-order accurate implementation of solid wall boundary conditions in curved geometries

Lilia Krivodonova *, Marsha Berger

Department of Mathematics, Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, NY 10012, United States

Received 17 November 2004; received in revised form 19 May 2005; accepted 23 May 2005
Available online 8 August 2005

Abstract

We propose a new technique to implement solid wall boundary conditions for steady two-dimensional Euler equations for problems in curved geometries. The technique is to be used with high-order methods on unstructured, straight-sided element meshes. By modeling flow around a physical rather than computational geometry, significant improvement in quality of the solution is achieved. The technique does not require a complex reconstruction and is easy to implement. Examples are presented to demonstrate validity of the new approach.

1. Introduction

Correct treatment of boundary conditions is crucial for developing accurate numerical schemes. Difficulties can arise when the numerical boundary does not coincide with the physical boundary. Large errors may arise in the boundary layer and pollute the solution inside the domain. This is especially important for higher-order methods, where errors due to geometrical approximation may dominate the discretization error, rendering the use of a higher-order scheme useless [5,18]. In this paper, we focus on implementation of solid wall boundary conditions for two-dimensional compressible inviscid Euler equations in curved geometries. Though our results are applicable to any higher-order scheme, we are primary interested in the discontinuous Galerkin method (DGM). It has been shown [4,5] that the DGM for flow problems is highly sensitive to the accuracy of the boundary representation. Our goal is to improve the accuracy of the boundary conditions without resorting to a full fledged higher-order representation of the geometry.

* Corresponding author.
E-mail address: krivol@cims.nyu.edu (L. Krivodonova).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2005.05.029

The most popular way to impose boundary conditions at solid walls for flow problems is the reflection technique, where an extra row (rows) of ghost cells is added behind the wall. All interior solution components are reflected symmetrically to ghost states except for the normal velocity which is negated; then a Riemann problem is solved on the boundary. Due to the symmetry of the reflection, only pressure contributes to the boundary flux. As a modification of the method, pressure obtained from interior values might be used directly in computation of the boundary flux eliminating the extra work required to construct ghost cells. With the DGM, this results in a slightly less accurate solution. The method works well when a wall is straight and leads to large errors when it is not. The inaccuracy of the approach when applied to curved geometries was demonstrated by Moretti [12] in the late 1960s. The normal derivative of pressure at the boundary is zero for a numerical solution and is far from it for the exact one. Thus, the reflecting boundary conditions are physically wrong for a curved geometry.

In general, more accurate techniques take into account the curvature of the solid wall instead of treating it locally as a straight line. Rizzi [14] incorporates curvature into boundary conditions by streamline differentiating the normal velocity equation, where the normal is taken relative to the physical geometry. Pressure at the boundary points is extrapolated from interior values of density, velocity, pressure, and the rate of change of the normal vector. More recent work of Dadone [8] extends this result. Pressure at the ghost cells is computed by Rizzi's formula and then used to obtain density and tangential velocity from the constant entropy and total-enthalpy vortex model. The normal component of velocity is computed according to the reflecting technique. Two rows of ghost cells are required for the second-order finite volume method. The solutions obtained with this approximation are more accurate at the expense of loss of conservation. These boundary conditions are applicable only to non-transient problems.

Bassi and Rebay [4,5] showed that DGM solutions are more sensitive to the error arising at curved boundaries than those obtained with finite volume methods of the same (theoretical) order of accuracy. Moreover, the solution may become less accurate as the order of approximation increases (Section 3). This is especially unfortunate since the DGM is considered an extension of the finite volume method to higher orders. As such, it can be computationally very efficient in regions where the solution is smooth. For example, to capture the structure of the solution near the leading edge of an airfoil. Bassi and Rebay [4,5] concluded that a high-order approximation of the physical geometry is a must for obtaining a meaningful solution. This is the point of view currently accepted by the DGM community [6]. Higher-order geometrical approximation is achieved by using mesh elements with one or more curved sides, usually described by polynomials. It has been shown numerically [5] that with the reflecting boundary conditions the order of the polynomials approximating the geometry must be at least equal to the order of the polynomial basis but not less than two in order to achieve the optimal rate of convergence. This is similar to the finite element method for elliptic equations where errors due to discretization of the domain arise in the boundary layer and may dominate the discretization error of the numerical scheme there. Choosing the degree of the polynomial approximating the boundary equal to the order of the basis functions is proven to be sufficient for obtaining the optimal rate of convergence in the energy norm [18].

However, curved element meshes are associated with extra computational expenses. First, curved elements need to be mapped onto the computational straight-sided element by a nonlinear mapping. To account for the non-constant Jacobian, a higher-order integration scheme must be used to compute volume and boundary integrals. Additionally, basis functions must be created and stored for each curved element when an orthogonal basis is used. The quadrature-free form of the DGM avoids a nonlinear mapping by employing canonical elements with curved sides [3]. When an adaptive scheme with p-refinement is used, either the geometrical order must be high enough from the beginning or a new, higher-order mesh element must be created every time a boundary element is marked for p-refinement. However, the biggest difficulty is to discretize the domain with curved elements. It is often unrealistic to obtain curved-element meshes for complicated geometries, especially in three dimensions. Moreover, most of the non-commercial

mesh-generating software currently produces only linear elements. In the absence of readily available products, coding higher-order curved elements on one's own may prove to be a tedious task.

This situation motivated us to seek a simpler technique. We start with an unstructured straight-sided element mesh. Since we are modeling compressible inviscid flow around an object, we assume that the streamlines follow the contours of the body. We require the numerical velocity at every boundary integration point to coincide with the streamline direction near the body surface, i.e., to be orthogonal to the "true" geometrical boundary rather than to the computational boundary. Instead of being tangential to the edge of the element, the flow "leaves" the domain at an integration point and "returns" at the symmetric integration point (we use Gauss–Legendre quadrature where integration points are symmetric about the center of an edge). It is reminiscent of the transpiration boundary conditions [13], which are used to simulate a flow around a perturbed geometry when remeshing is considered impractical. Curvature of a solid boundary is either obtained from the geometrical description of the object or approximated locally using data from neighboring elements. Setting up a ghost state, we reflect the velocity with respect to the "true" normal and map the other components of the solution symmetrically. However, the normal used in computing the boundary flux is the usual normal to the computational element, constant along the edge.

While our curvature boundary conditions (CBC) are more accurate than the reflecting boundary conditions (RBC) as measured by errors in solution components and quantities of interest, they are not conservative. Non-conservative schemes are usually treated with caution because they may result in an incorrect shock position and strength. However, such schemes have been successfully used in some situations. In recent years, work has appeared where authors advocate loss of conservation as a lesser evil. For example, by applying a non-conservative scheme near the interface of two fluids, Karni [11] eliminates large spurious oscillations. The dramatic improvement in the quality of the solution outweighs the (very small) loss of conservation. As we have already mentioned, the boundary conditions of Dadone [8] for the finite volume method are also non-conservative. As it is the case with [11], the resulting scheme is conservative except for boundary elements, i.e., the loss of conservation occurs in a lower dimension. The loss of mass in our experiments is almost negligible (the worst is 0.0004% for a round body and a very coarse mesh). The non-penetration boundary conditions are satisfied in the limit as the total mass loss converges to zero under h- and p-refinement. On the other side, the RBC do not result in a numerical scheme properly modeling the exact solution on a straight-sided element mesh. As we show on an example of a flow around a circular cylinder (Section 3), the reflecting technique results in unsteady flow with vortices attached to the back of a cylinder when the exact solution is a steady irrotational flow. No reasonable mesh refinement eliminates the wake, and p-refinement actually decreases the quality of the solution. This makes the curvature boundary conditions preferential in our view.

The rest of the paper is organized as follows. Section 2 briefly describes the essential features of the DGM. For more detail see [7] and references therein. The reflecting boundary conditions are discussed in Section 3. The new curvature boundary conditions and their numerical implementation are introduced in Section 4. Section 5 is devoted to numerical experiments. We show that the errors in solutions obtained with the CBC are close to those obtained with exact boundary conditions (Example 5.1) or comparable to those obtained on curved high-order meshes (Example 5.2) and exhibit the theoretical rate of convergence. We find that the global aerodynamic constants for external flows are similar to described in the literature (Example 5.3). Conclusions are in Section 6.

2. Discontinuous Galerkin formulation

We consider the two-dimensional Euler equations

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2+P\\ \rho uv\\ u(E+P)\end{pmatrix}+\frac{\partial}{\partial y}\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2+P\\ v(E+P)\end{pmatrix}=\mathbf{0} \tag{1a}$$

on a bounded domain Ω with appropriate well-posed boundary data prescribed on $\partial\Omega$. As is customary, ρ is the density of the fluid, u and v are components of the velocity vector $\vec{v}$, P is the pressure, and E is the total energy. We assume the fluid to be an ideal polytropic gas satisfying the equation of state

$$P=(\gamma-1)\left(E-\frac{\rho\|\vec{v}\|^2}{2}\right), \tag{1b}$$

where γ is the adiabatic exponent, which was set to 1.4 for the numerical experiments.

We use the discontinuous Galerkin method in the formulation originally proposed by Cockburn and Shu. Here, we provide a brief synopsis of the numerical scheme; see [7] and references therein for a detailed analysis.

In order to describe the method, we write (1a) as a general conservation law

$$\partial_t\mathbf{u}+\operatorname{div}\mathbf{F}(\mathbf{u})=\mathbf{0},\quad \mathbf{x}\in\Omega,\quad t>0, \tag{2a}$$

$$\mathbf{u}=\mathbf{u}^0,\quad t=0. \tag{2b}$$

We divide the problem domain Ω into a collection of non-overlapping elements

$$\Omega=\bigcup_{j=1}^{N_h}\Omega_j. \tag{3}$$

Then, we construct a Galerkin problem on element Ω_j by multiplying (2a) by a test function $\mathbf{v}\in L^2(\Omega_j)$, integrating the result on Ω_j, and using the Divergence Theorem to obtain

$$\int_{\Omega_j}\mathbf{v}\partial_t\mathbf{u}\,\mathrm{d}s+\int_{\partial\Omega_j}\mathbf{v}\mathbf{F}(\mathbf{u})\cdot\vec{n}\,\mathrm{d}\tau-\int_{\Omega_j}\operatorname{grad}\mathbf{v}\cdot\mathbf{F}(\mathbf{u})\,\mathrm{d}s=\mathbf{0}\quad\forall\mathbf{v}\in(L^2(\Omega_j))^4, \tag{4}$$

where $\vec{n}$ is the normal vector to $\partial\Omega_j$. The solution $\mathbf{u}$ is approximated by a vector function $\mathbf{U}_j=(U_{j,1},\,U_{j,2},\,U_{j,3},\,U_{j,4})^{\mathrm{T}}$, where

$$U_{j,k}=\sum_{i=1}^{N_p}c_{i,k,j}\varphi_i,\quad k=1,2,3,4, \tag{5}$$

in a finite-dimensional subspace of the solution space. The basis $\{\varphi_i\}_{i=1}^{N_p}$ is chosen to be orthonormal in $L^2(\Omega_j)$ [10], which will produce a multiple of the identity for the mass matrix on Ω_j.

Due to the discontinuous nature of the numerical solution, the normal flux $\mathbf{F}_n=\mathbf{F}(\mathbf{u})\cdot\vec{n}$, is not defined on $\partial\Omega_j$. The usual strategy is to define it in terms of a numerical flux $\mathbf{F}_n(\mathbf{U}_j,\mathbf{U}_k)$ that depends on the solution $\mathbf{U}_j$ on Ω_j and $\mathbf{U}_k$ on the neighboring element Ω_k sharing the portion of the boundary $\partial\Omega_{jk}$ common to both elements. In our experiments, we used the Roe numerical flux [15]. Finally, the L^2 volume and surface inner products in (4) are computed using $2p$ and $2p+1$ order accurate Gauss quadratures [10], respectively, where p is the order of the orthonormal basis. The resulting system of ODEs can be solved element-wise when an explicit numerical scheme is used. We use an explicit total variation bounded Runge–Kutta scheme of an appropriate order [7].

3. Reflecting boundary conditions

The reflecting boundary conditions state that no flow penetrates a solid wall, i.e., the normal velocity at the wall is zero. Depending on the numerical scheme, a ghost state or cell is created on the part of the numerical boundary $\partial\Omega^{w}$ corresponding to the solid wall. With the DGM, a ghost state is created at every integration point on $\partial\Omega^{w}$, where all components of the ghost solution are set equal to the corresponding interior values at the same point except for the normal velocity, which is negated. Then, the interior and ghost states are passed to a Riemann solver. Due to the symmetry of the reflection, the solution to the Riemann problem at integration point $\mathbf{x}_j \in \partial\Omega^{w}$ satisfies [17]

$$\vec{v}(\mathbf{x}_j) \cdot \vec{n} = 0. \tag{6}$$

This approach works well for straight-sided bodies. However, results are inferior when a physical geometry is more complex. As an example, we consider a Mach 0.38 flow around a circular cylinder on a 128×32 O-grid mesh. A detailed description of the problem is given in Example 5.2. The computations were initiated with the free stream values. After decreasing three orders of magnitude to the 10^{-2}–10^{-3} range, residuals given by (20) oscillated in this range and failed to converge further. Computations were stopped after no improvement was observed for a number of time steps. Mach isolines are presented in Fig. 1, left, for $p = 1, 2, 3$ with an increment $\Delta M = 0.038$. Plotting was performed using data from inside of mesh elements. Since the numerical solutions are discontinuous across interelement boundaries, the isolines

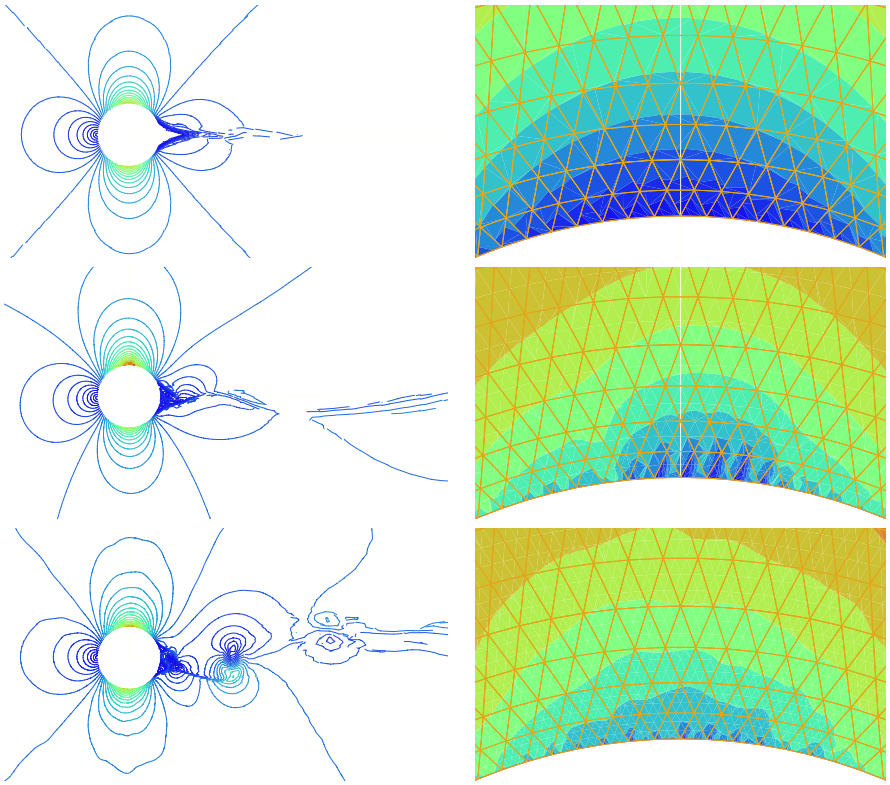

Fig. 1. Mach isolines (left) and density at the top of the cylinder (right) with reflecting boundary conditions. $p = 1, 2, 3$, from top to bottom. A wake is formed at the rear; the solution does not achieve a steady symmetric irrotational form.

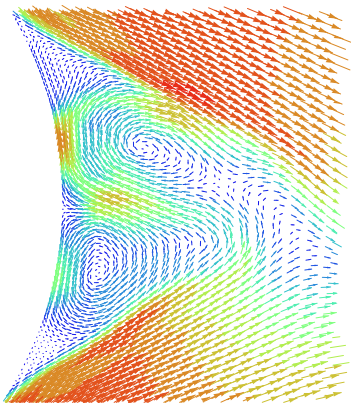

Fig. 2. Velocity field near the outflow part of a cylinder. $p = 3$, reflecting boundary conditions.

appear as broken lines where the solution changes rapidly from an element to its neighbor. The quality of the numerical solutions is visibly poor. They do not appear to be good approximations to the exact solution, which is steady, smooth, subcritical and symmetric, with streamlines following contours of the body [2]. The numerical solutions, however, are unsteady and may become transonic with higher ($p = 2, 3$) orders of approximation. Instead of "wetting" the surface, the flow separates from the back of the cylinder forming a wake. A close look at the velocity plot in Fig. 2 reveals two vortices at the back side of the cylinder. Remarkably, the quality of solutions deteriorate as the order of approximation increases: the solutions become less symmetric and the wake increases.

The likely explanation is that by increasing the order of approximation, we obtain a more accurate solution to a wrong problem: flow around a polygon. Rarefaction waves are formed at the vertices of the polygon [2]. These are better resolved with higher p. Density plots near the top of the cylinder (with the background mesh) in Fig. 1, right, demonstrate concentration of the error near vertices. Isolines take a wave-like shape instead of a smooth curve. This becomes increasingly so and affects solution in further parts of the domain as p increases.

4. Curvature boundary conditions

Meshing a non-polygonal shaped domain necessarily introduces an error. As we have seen, the DGM is highly sensitive to the error due to approximation of a curved geometry by a straight-sided element mesh. This error may dominate the discretization error of the scheme and lead to a wrong solution. We seek to impose boundary conditions which would take this into consideration and model internal or external flow more accurately.

We start by observing that the "no flow through a wall" rule refers to the physical boundary. Thus, (6) does not model the non-penetration boundary conditions properly when a computational geometry does not coincide exactly with the physical one. Imposing no flow through the physical boundary in this case means allowing some flow through the computational boundary. Modeling compressible inviscid flow, we assume that streamlines follow the contours of an object. This is equivalent to requiring the velocity vector on the surface to be orthogonal to the normal to the surface. We assume this to be true in a small vicinity to the surface and impose the following condition at every integration point:

$$\vec{v}(\mathbf{x}) \cdot \vec{N}(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega_j^{\mathrm{w}}, \tag{7}$$

where $\vec{N}(\mathbf{x})$ is the unit normal to the physical geometry at point $\mathbf{x}$. Fig. 3 illustrates (7) for element Ω_j adjacent to the solid boundary.

In practice, an analytical description of the surface is usually not available. Thus, in order to evaluate the boundary integral in (4) numerically, $\vec{N}(\mathbf{x})$ needs to be approximated at the integration points from the available data. In our computations, we assumed that the only available information is the mesh itself. We approximated the physical geometry by an arc of the circle passing through the vertices of $\partial\Omega_j^{\mathrm{w}}$. The radius of the circle is taken to be the average of the radii of two circles passing though three points: two vertices of $\partial\Omega_j^{\mathrm{w}}$ and the vertex lying on the solid boundary immediately to the left or right of $\partial\Omega_j^{\mathrm{w}}$. Using the assumption that (7) is satisfied in close vicinity of the wall, $\vec{N}$ at $\mathbf{x}_i$, $i = 1,2,\ldots,n_{\mathrm{G}}$, is computed as the unit normal to the circle passing through $\mathbf{x}_i$ and the symmetric Gauss–Legendre point $\mathbf{x}_{\mathrm{G}-i}$ with the center at the same point as the circle approximating the boundary. Only when the number of integration points is odd does $\vec{N}$ at the center of $\partial\Omega_j^{\mathrm{w}}$ coincide with the normal to the edge of the straight-sided element. The process is illustrated in Fig. 4 for $p = 4$ approximation that requires 9th-order accurate numerical integration which we perform by the 5-point Gauss–Legendre rule. Possible sharp corners of the physical object are dealt with by computing the dot product of the normal to the straight-sided edge $\vec{n}_j$ and the normals to its neighbors. We assume a sharp corner, such as the cusp of the airfoil in Example 5.3, to be present if $\vec{n}_j \cdot \vec{n}_{\mathrm{neigh}} < 0$. One sided approximation of the local curvature is used in this case.

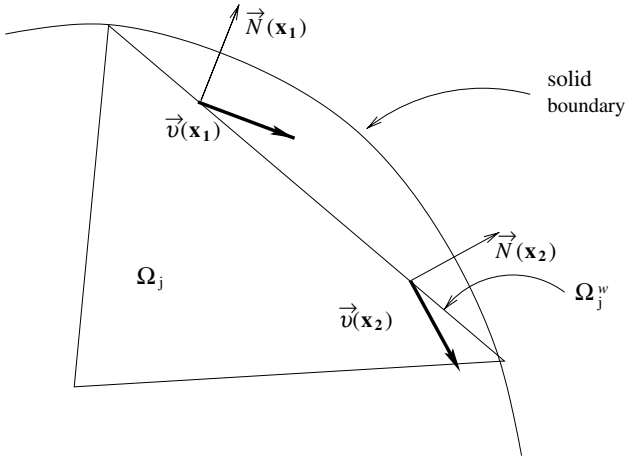

Fig. 3. Curvature boundary conditions for the two point Gauss–Legendre integration rule. Velocity at points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is orthogonal to the physical normals $\vec{N}(\mathbf{x}_1)$ and $\vec{N}(\mathbf{x}_2)$.

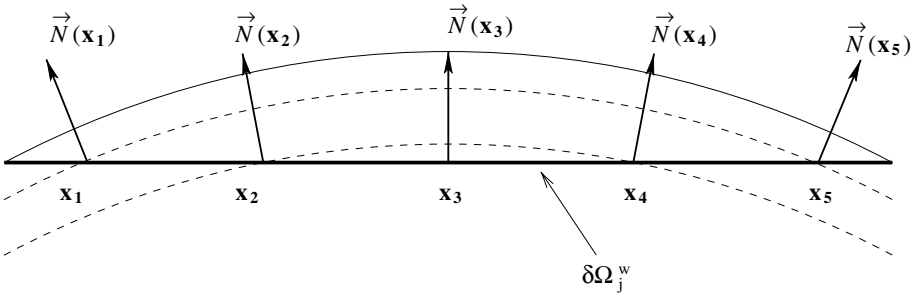

Fig. 4. Construction of vectors $\vec{N}$ at boundary integration points $\boldsymbol{x}_i$, $i = 1,2,\ldots,5$ for $p = 4$.

We present three algorithms for imposing (7) numerically. We show in Section 5 that Algorithm I is less accurate than the other two. Algorithms II and III result in almost identical solutions. We prefer Algorithm II for its simplicity and use it in our numerical experiments in Section 5.

Before proceeding, we need to describe the notation used. Here, $\vec{n} = (n_1, n_2)$ and $\vec{t}$ are unit normal and tangential vectors to the computational boundary, $\vec{N} = (N_1, N_2)$ and $\vec{T}$ to the physical boundary, respectively. Subscripts refer to projections on the respective vector. Superscripts are reserved to differentiate between different states at the same point: b denotes a boundary value, i.e., the value used to compute the numerical flux $\mathbf{F}_n$ in (4), the superscript g refers to a ghost cell value, and variables without a superscript denote interior values. According to this notation, $v_{\mathrm{n}}^{\mathrm{g}}$ is the projection of ghost state velocity vector $\vec{v}^{\mathrm{g}}$ onto $\vec{n}$.

Algorithm I. This algorithm uses interior solution values to compute the numerical flux on the solid boundary in (4). This eliminates the need to solve the Riemann problem on $\partial\Omega_j^{\mathrm{w}}$. Condition (7) is enforced by setting the projection of the velocity vector onto $\vec{N}$ to zero at each integration point. This is done locally in the boundary solver without modifying the solution inside Ω_j.

Let $\vec{v} = (u, v)$ be the velocity at integration point $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n_{\mathrm{G}}$, on $\partial\Omega_j^{\mathrm{w}}$. Computing the tangential boundary velocity $v_{\mathrm{T}}^{\mathrm{b}} = \vec{v}^{\mathrm{b}} \cdot \vec{T}$ relative to the physical geometry as

$$v_{\mathrm{T}}^{\mathrm{b}} = v_{\mathrm{T}} = uN_2 - vN_1, \tag{8}$$

setting the normal boundary velocity $v_{\mathrm{N}}^{\mathrm{b}} = \vec{v} \cdot \vec{N}$ to zero,

$$v_{\mathrm{N}}^{\mathrm{b}} = 0 \tag{9}$$

and rotating (8, 9) back into (u,v) space, we force the boundary state velocity $\vec{v}^{\mathrm{b}}$ at $\boldsymbol{x}_i$ to satisfy (7). Using the interior values of density and pressure for the other components of $\mathbf{U}_j^{\mathrm{b}}(\mathbf{x}_i)$ results in the solution vector at $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n_{\mathrm{G}}$, on $\partial\Omega_j^{\mathrm{w}}$

$$\begin{aligned} \rho^{\mathrm{b}} &= \rho, \\ u^{\mathrm{b}} &= (uN_2 - vN_1)N_2, \\ v^{\mathrm{b}} &= -(uN_2 - vN_1)N_1, \\ P^{\mathrm{b}} &= P. \end{aligned} \tag{10}$$

Then, (10) is used to compute the numerical flux in (4) as

$$F_n(\mathbf{U}_j^{\mathrm{b}}) = (\rho^{\mathrm{b}}\vec{v}^{\mathrm{b}} \cdot \vec{n}, \rho^{\mathrm{b}}u^{\mathrm{b}}\vec{v}^{\mathrm{b}} \cdot \vec{n}, \rho^{\mathrm{b}}v^{\mathrm{b}}\vec{v}^{\mathrm{b}} \cdot \vec{n}, \vec{v}^{\mathrm{b}} \cdot \vec{n}(E^{\mathrm{b}} + p^{\mathrm{b}})). \tag{11}$$

Algorithm II. This algorithm seeks to improve on Algorithm I by constructing ghost states at integration points on $\partial\Omega_j^{\mathrm{w}}$ and solving the Riemann problem there. The ghost state values are extrapolated from the interior solution with the aim for $\mathbf{U}_j^{\mathrm{b}}$ to satisfy (7). Although the velocity vectors obtained by Algorithms I and II are similar, the pressure P^{b} and density ρ^{b} resulting from the Riemann solver correspond to $v_{\mathrm{N}}^{\mathrm{b}} = 0$, not to the original velocity vector, and, thus, are more accurate.

We proceed as follows. A ghost state $\mathbf{U}_j^{\mathrm{g}}(\mathbf{x}_i)$ is created at each boundary integration point $\mathbf{x}_i$, $i = 1, 2, \ldots, n_{\mathrm{G}}$. The velocity vector $\vec{v} = (u, v)$ is reflected to the ghost state with respect to $\vec{T}$ (Fig. 5). The normal and tangential components relative to the physical geometry at the ghost state are given by

$$v_{\mathrm{N}}^{\mathrm{g}}(\mathbf{x}_i) = -v_{\mathrm{N}}(\mathbf{x}_i), \qquad v_{\mathrm{T}}^{\mathrm{g}}(\mathbf{x}_i) = v_{\mathrm{T}}(\mathbf{x}_i), \tag{12}$$

while the density and pressure are copied exactly from the interior values at the same point. Rotating (12) back into (u, v) space, we obtain the ghost state vector as

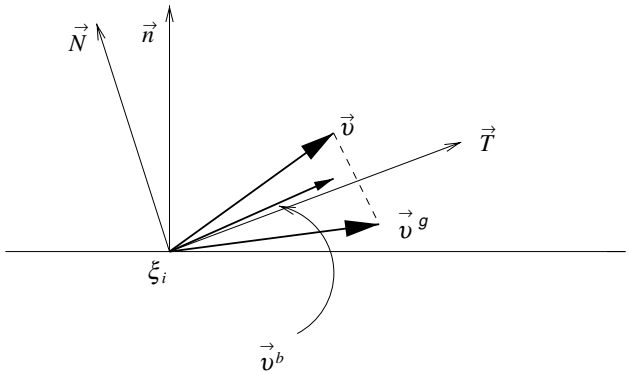

Fig. 5. Velocity vector reflection. Algorithm II.

$$\begin{aligned}
\rho^{\mathrm{g}} &= \rho,\\
u^{\mathrm{g}} &= u[(N_2)^2 - (N_1)^2] - 2N_1N_2v,\\
v^{\mathrm{g}} &= v[(N_1)^2 - (N_2)^2] - 2N_1N_2u,\\
P^{\mathrm{g}} &= P.
\end{aligned} \tag{13}$$

Finally, the Riemann problem $riemann(\mathbf{U}, \mathbf{U}^{\mathrm{g}}; \vec{n})$ is solved. Due to the identical pressure and density in the ghost and interior states, the exact solution of the Riemann problem consists of either two shocks or two rarefaction waves [17] with the normal velocity at the interface given by

$$v_{\mathrm{N}}^{\mathrm{b}} = \tfrac{1}{2}(v_n + v_n^g). \tag{14}$$

The tangential velocity $v_{\mathrm{t}}^{\mathrm{b}}$ equals to v_{t} or $v_{\mathrm{t}}^{\mathrm{g}}$ depending on the sign of $\vec{v}\cdot\vec{n}$. As a result, $\vec{v}^{\mathrm{b}}$ is not exactly orthogonal to $\vec{N}$. The error depends on the velocity vector and the curvature of the boundary. In all our experiments, even on very coarse meshes, it was small and did not noticeably affect quality of the solution. However, by slightly modifying the mapping of the velocity to the ghost state we can force $\mathbf{U}_j^{\mathrm{b}}$ to satisfy (7) exactly.

Algorithm III. This algorithm is a modification of Algorithm II, where the no flow through the physical boundary is satisfied exactly. The velocity vector $\vec{v}^{\mathrm{b}}$ resulting from the solution of the Riemann problem on $\partial\Omega_j^{\mathrm{w}}$ can be forced to satisfy (7) by requiring $v_{\mathrm{t}}^{\mathrm{g}} = v_{\mathrm{t}}$ and reflecting v_n with respect to $\vec{T}$ (Fig. 6)

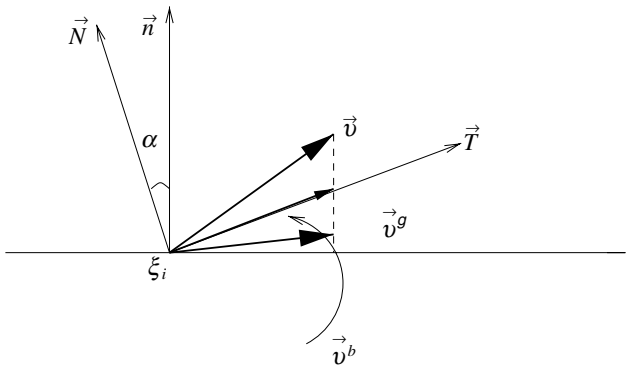

Fig. 6. Velocity vector reflection. Algorithm III.

$$v_{\mathrm{n}}^{\mathrm{g}} = v_{\mathrm{n}} - 2(v_{\mathrm{n}} - \mathrm{sgn}(v_{\mathrm{n}})|v_{\mathrm{t}}| \tan\alpha), \tag{15}$$

where α is the angle between $\vec{N}$ and $\vec{n}$ and the two normals point to the same side of $\partial\Omega_j^{\mathrm{w}}$, i.e., $(\vec{N}\cdot\vec{n}) > 0$. After rotating (15) back into (u, v) space, the ghost state is computed as

$$\begin{aligned} \rho^{\mathrm{g}} &= \rho, \\ u^{\mathrm{g}} &= v_{\mathrm{n}}^{\mathrm{g}} n_1 + v_{\mathrm{t}}^{\mathrm{g}} n_2, \\ v^{\mathrm{g}} &= v_{\mathrm{n}}^{\mathrm{g}} n_2 - v_{\mathrm{t}}^{\mathrm{g}} n_1, \end{aligned} \tag{16}$$

$$P^{\mathrm{g}} = P. \tag{17}$$

Again, the Riemann problem $\mathit{riemann}(\mathbf{U}, \mathbf{U}^{\mathrm{g}}; \vec{n})$ is solved at x_i, $i = 1, 2, \ldots, n_{\mathrm{G}}$.

Although the velocity vector resulting from Algorithm III should be more accurate than one obtained with Algorithm II, the algorithms performed nearly identically in our experiments. We should note, however, that condition (7) is not exact by itself due to the approximate nature of $\vec{N}$ for a general non-circular geometry. We preferred Algorithm II for its simplicity. It was used in numerical experiments in Section 5 unless otherwise indicated. The comparative performance of the three algorithms is discussed in Example 5.1.

5. Numerical examples

In order to demonstrate the new method, we present several examples. The results were obtained by using a time-dependent DG code and integrating until a steady state was reached. Plotting was done using data from inside of elements; no smoothing was applied. The discontinuous nature of the numerical solution will in this case result in a broken pattern of isolines when jumps in the solution values across inter-element boundaries are large.

5.1. Supersonic vortex

We consider an isentropic supersonic flow between two concentric circular arcs of radii $r_{\mathrm{i}} = 1$ and $r_{\mathrm{o}} = 1.384$ in the first quadrant, a test problem found in [1]. The exact density in terms of radius r is given by

$$\rho = \rho_{\mathrm{i}} \left(1 + \frac{\gamma - 1}{2} M_{\mathrm{i}}^2 \left(1 - \left(\frac{r_{\mathrm{i}}}{r}\right)^2\right)\right)^{(1/(\gamma-1))}. \tag{18}$$

The velocity and pressure are given by

$$\|\vec{v}\| = \frac{c_{\mathrm{i}} M_{\mathrm{i}}}{r}, \qquad P = \frac{\rho^{\gamma}}{\gamma}, \tag{19}$$

where c_{i} is the speed of sound on the inner circle. The Mach number on the inner circle M_{i} is set to 2.25 and the density ρ_{i} to one.

We solve the problem on a sequence of unstructured triangular meshes containing 140, 620 and 2406 elements (Fig. 7). All computations were performed until solution coefficients reached a steady state, defined as the residual

$$R = \sqrt{\sum_{j=1}^{N_h} \sum_{k=1}^{4} \sum_{i=1}^{N_p} (c_{i,k,j}^{n+1} - c_{i,k,j}^{n})^2} \tag{20}$$

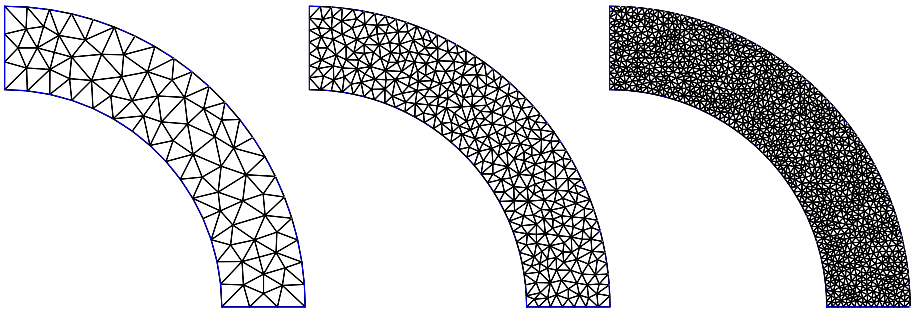

Fig. 7. The sequence of meshes used for supersonic vortex.

being less than 10^{-15}, where $c^n_{i,k,j}$ is a solution coefficient on Ω_j given by (5) at time step n. We measure the exact L^2 error in density and pressure for each of the proposed algorithms as well as for the reflecting boundary conditions. For comparison, we also measure the errors when the exact boundary conditions are applied, i.e., when the exact solution is used in computing the flux on the boundary. The results are reported in Tables 1 and 2. The errors associated with Algorithms I, II, and III are close to those of the solutions obtained with the exact boundary conditions, indicating that the part of the error due to the boundary conditions is small. Algorithms II and III perform the best and are almost identical, Algorithm

Table 1
L^2 errors in pressure and convergence rates r for supersonic vortex

N	p							
	$p=1$		$p=2$		$p=3$		$p=4$	
	Error	r	Error	r	Error	r	Error	r
Exact boundary conditions								
140	3.34E − 03	–	2.06E − 04	–	5.38E − 06	–	3.21E − 07	–
620	9.35E − 04	1.84	3.09E − 05	2.74	4.13E − 07	3.70	1.44E − 08	4.48
2406	2.31E − 04	2.02	3.73E − 06	3.05	2.56E − 08	4.01	5.54E − 10	4.70
Algorithm I								
140	4.43E − 03	–	2.61E − 04	–	6.99E − 06	–	4.09E − 07	–
620	1.09E − 03	2.02	4.76E − 05	2.46	5.40E − 07	3.69	1.81E − 08	4.50
2406	2.45E − 04	2.15	5.48E − 06	3.12	3.34E − 08	4.02	7.26E − 10	4.64
Algorithm II								
140	4.01E − 03	–	2.63E − 04	–	6.61E − 06	–	4.04E − 07	–
620	1.03E − 03	1.96	4.43E − 05	2.57	4.81E − 07	3.78	1.84E − 08	4.46
2406	2.38E − 04	2.11	4.91E − 06	3.17	3.06E − 08	3.97	6.98E − 10	4.72
Algorithm III								
140	3.98E − 03	–	2.63E − 04	–	6.64E − 06	–	4.05E − 07	–
620	1.03E − 03	1.95	4.44E − 05	2.57	4.81E − 07	3.79	1.86E − 08	4.44
2406	2.37E − 04	2.12	4.92E − 06	3.17	3.06E − 08	3.97	6.99E − 10	4.73
Reflecting boundary conditions								
140	4.74E − 02	–	6.05E − 02	–	6.63E − 02	–	7.60E − 02	–
620	1.50E − 02	1.66	1.89E − 02	1.68	2.41E − 02	1.46	3.02E − 02	1.33
2406	6.32E − 03	1.25	7.71E − 03	1.29	9.10E − 03	1.41	1.17E − 02	1.37

Table 2
L^2 errors in density and rates of convergence r for supersonic vortex

N	p							
	$p=1$		$p=2$		$p=3$		$p=4$	
	Error	r	Error	r	Error	r	Error	r
Exact boundary conditions								
140	4.04E − 03	–	1.89E − 04	–	6.75E − 06	–	5.34E − 07	–
620	1.19E − 03	1.76	2.72E − 05	2.80	6.21E − 07	3.44	3.65E − 08	3.87
2406	2.95E − 04	2.01	3.30E − 06	3.04	3.97E − 08	3.97	9.12E − 10	5.32
Algorithm I								
140	5.30E − 03	–	2.42E − 04	–	7.36E − 06	–	5.69E − 07	–
620	1.50E − 03	1.82	4.00E − 05	2.60	6.48E − 07	3.51	3.71E − 08	3.94
2406	3.53E − 04	2.09	4.61E − 06	3.12	4.17E − 08	3.96	9.72E − 10	5.25
Algorithm II								
140	4.65E − 03	–	2.26E − 04	–	7.32E − 06	–	5.70E − 07	–
620	1.37E − 03	1.76	3.66E − 05	2.63	6.30E − 07	3.54	3.69E − 08	3.95
2406	3.29E − 04	2.06	4.11E − 06	3.15	4.05E − 08	3.96	9.52E − 10	5.28
Algorithm III								
140	4.64E − 03	–	2.26E − 04	–	7.34E − 06	–	5.70E − 07	–
620	1.37E − 03	1.76	3.66E − 05	2.63	6.30E − 07	3.54	3.70E − 08	3.95
2406	3.29E − 04	2.06	4.11E − 06	3.15	4.06E − 08	3.96	9.52E − 10	5.28
Reflecting boundary conditions								
140	3.32E − 02	–	4.32E − 02	–	4.75E − 02	–	5.05E − 02	–
620	1.05E − 02	1.66	1.35E − 02	1.68	1.72E − 02	1.47	2.14E − 02	1.24
2406	4.39E − 03	1.26	5.43E − 03	1.31	6.43E − 03	1.42	8.23E − 03	1.38

I is slightly less accurate. The convergence rates are the same as with the exact boundary conditions. The rates of convergence do not perfectly correspond to the number of elements due to the fact that the meshes used are not structured and not nested. The errors with the RBC are significantly larger. Moreover, a reduced $h^{3/2}$ convergence rate is observed under h-refinement for all p, while p-refinement results in decreased accuracy.

5.2. Flow around a circular cylinder

We consider a subsonic flow at Mach number $M_\infty = 0.38$ on four O-grid meshes having 16×4, 32×8, 64×16 and 128×32 points (Fig. 8). The first number refers to the number of points in the circular direction, the second describes the number of concentric circles in the mesh. The radius of the cylinder is $r_0 = 0.5$, the domain is bounded by $r_{32} = 20$, the radii of concentric circles for 128×32 mesh are set up as in [9]

$$r_j = r_0\left(1 + \frac{2\pi}{128}\sum_{k=0}^{j-1}\alpha^k\right), \qquad j = 1,2,\ldots,32, \tag{21}$$

with $\alpha = 1.1648336$. The coarser meshes are obtained by successively unrefining the finest mesh. All plots for this example are shown in the $[-2,2]\times[-2,2]$ square. First, we solve the problem on the sequence of meshes with $p = 1$ and plot Mach isolines in Fig. 9 with $\Delta M = 0.038$. The plots visually compare well with those obtained by Bassi and Rebay [5] and are significantly more accurate than ones obtained with the RBC

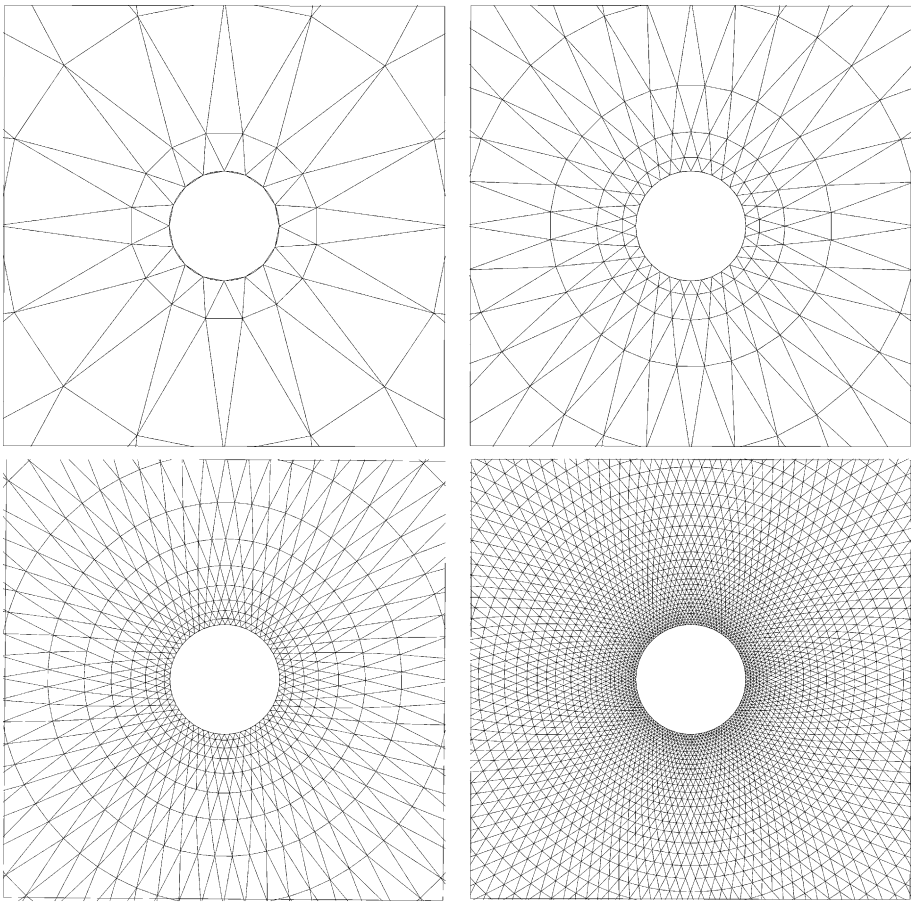

Fig. 8. 16×4, 32×8, 64×16, and 128×32 meshes around a circular cylinder.

(Section 3). The solution obtained on the finest mesh is symmetric and does not have a visible wake. Next, we perform p-refinement on the coarsest mesh and plot Mach isolines with the same ΔM for $p=1,2,3,4$ in Fig. 10. The quality of the solution clearly improves as p increases. The solution corresponding to $p=4$ is similar to one obtained on the finest mesh with $p=1$. In all our experiments, the velocity vectors near the surface followed the contour of the cylinder. As an illustration, we plot in Fig. 11 the velocity profile and a zoom near the rear stagnation point for the $p=2$ solution on the 32×8 mesh. The plots reveal a smooth flow that "wets" the surface.

To quantify our findings, we measure L^2 errors in entropy ϵ_{ent} defined as

$$\epsilon_{\text{ent}}=\frac{P}{P_\infty}\Big/\left(\frac{\rho}{\rho_\infty}\right)^{\gamma}-1, \tag{22}$$

where P_∞ and ρ_∞ are pressure and density of the free stream, respectively. The results with h- and p-refinement are reported in Table 3. Numbers compare well with [5]. Further, we present two aerodynamic quantities: the pressure coefficient C_{p}

$$C_{\text{p}}=\frac{P-P_\infty}{0.5\rho_\infty\|\vec{v}_\infty\|^2} \tag{23}$$

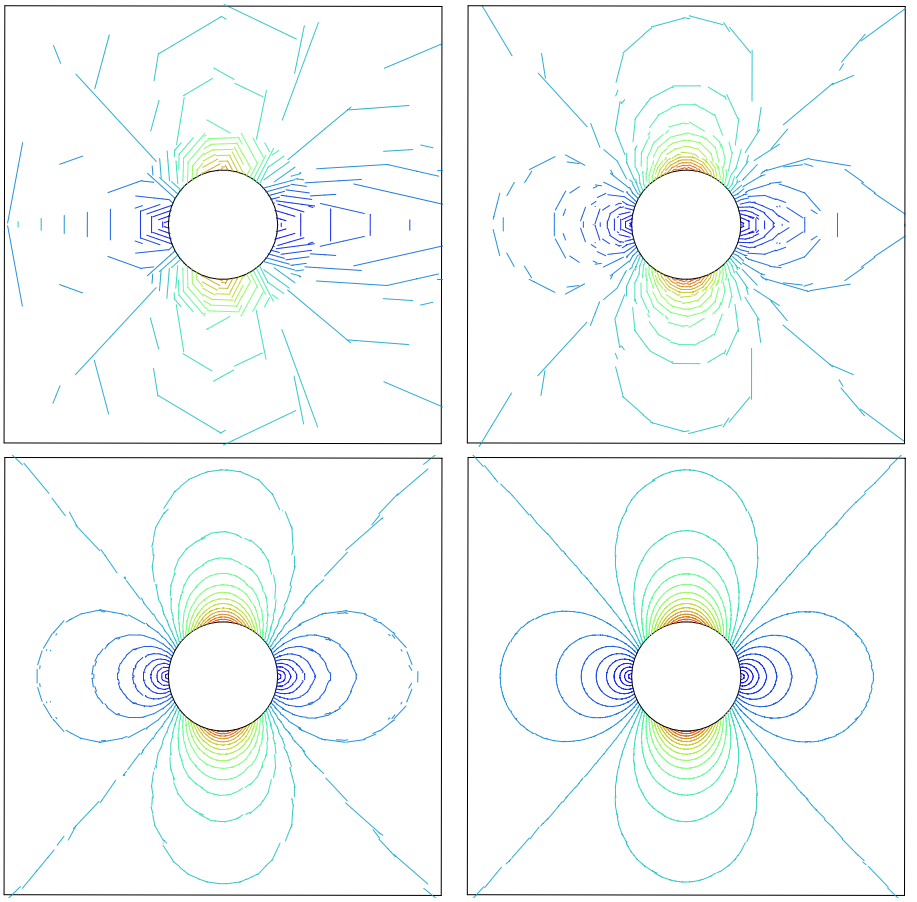

Fig. 9. Mach isolines on 16×4, 32×8, 64×16, and 128×32 meshes from left to right and from top to bottom. Circular cylinder, $p=1$, $\Delta M=0.038$.

and the total pressure loss coefficient defined as a ratio of the total pressure P_t

$$P_t = P\left(1+\frac{\gamma-1}{2}M^2\right)^{\gamma/(\gamma-1)} \tag{24}$$

at a point to the total pressure of the free stream. These are reported under h- and p-refinement in Figs. 12 and 13. The total pressure loss coefficient on the surface converges to unity pointwise under h- and p-refinement (Fig. 13); additionally, the convergence history of the error in the total pressure on the surface in the L^2 norm is presented in Fig. 14. Lift coefficients in the experiments were ranging from 10^{-10} to 10^{-15}, confirming that the solutions possess vertical symmetry. The mass loss, defined as the total flux through the solid boundary (or farfield boundary), was insignificant in all experiments. It ranged from 3.3×10^{-8} to 4.6×10^{-3} in absolute values. This corresponds to the $2.7\times10^{-9}\%$ to $3.8\times10^{-4}\%$ range. The convergence history under h- and p-refinement is reported in Fig. 15.

In our experiments, the CBC did not slow the speed of reaching a steady state. The number of time steps needed for convergence increased, as expected, in inverse proportion to either the reduced element size (h-refinement) or the CFL constraint (p-refinement). The number of time steps in our computations is comparable to those in [5] using higher-order elements. Comparing solutions obtained under h- and

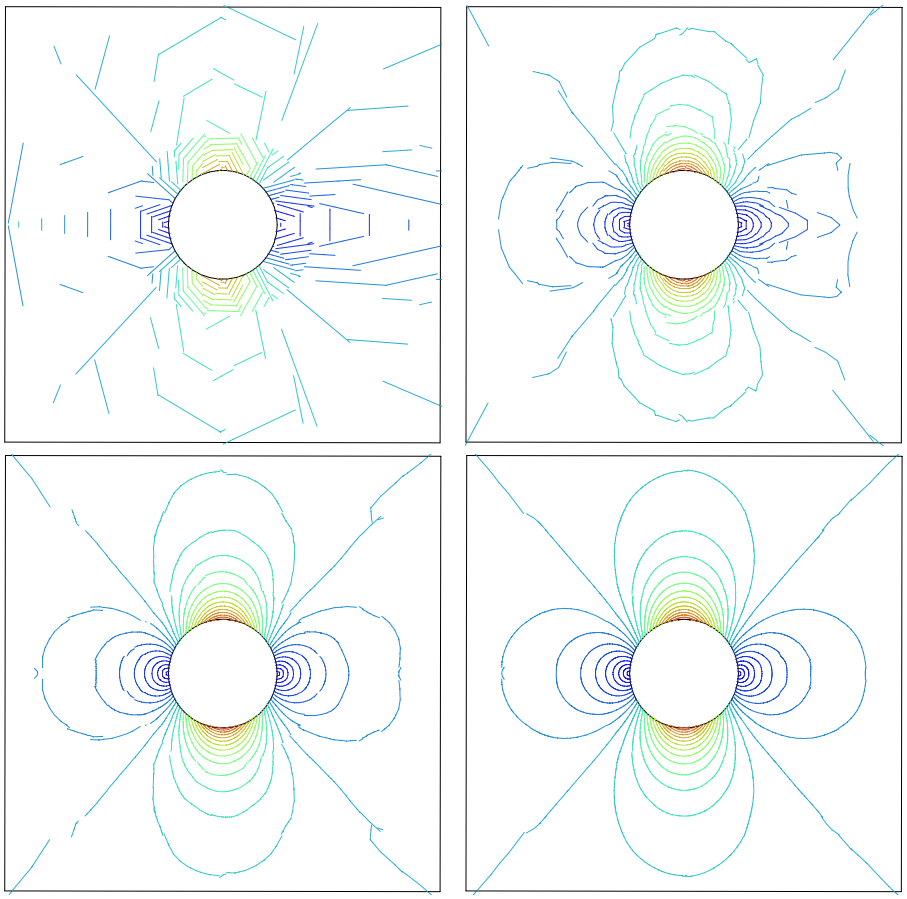

Fig. 10. Mach isolines under p-refinement on 16×4 grid. Circular cylinder, $p = 1, 2, 3, 4$ from left to right and from top to bottom, $\Delta M = 0.038$.

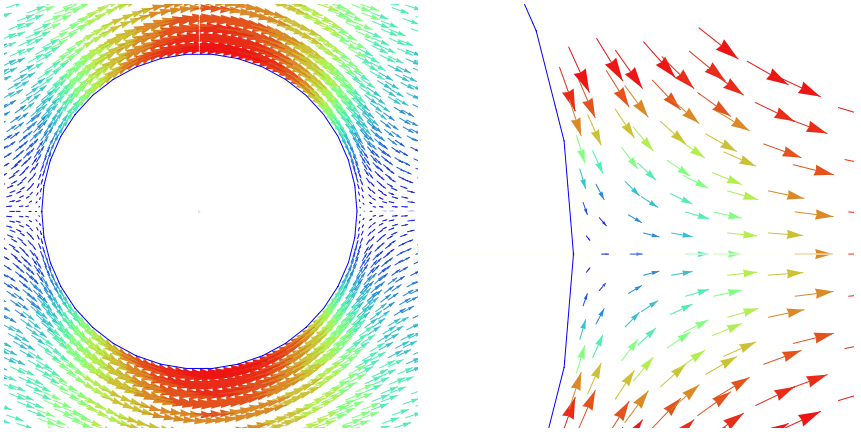

Fig. 11. Velocity near surface (left) and zoom at a stagnation point (right). Circular cylinder, $p = 2$, 32×8 mesh.

Table 3
L^2 errors in entropy and convergence rates for the circular cylinder

N	p					
	$p=1$		$p=2$		$p=3$	
	ϵ_{ent}	r	ϵ_{ent}	r	ϵ_{ent}	r
16×4	5.12E − 02	–	6.87E − 03	–	1.00E − 03	–
32×8	9.28E − 03	2.46	4.37E − 04	3.97	5.41E − 05	4.21
64×16	1.42E − 03	2.71	3.75E − 05	3.54	3.55E − 06	3.93
128×32	2.09E − 04	2.76	4.05E − 06	3.21	2.43E − 07	3.87

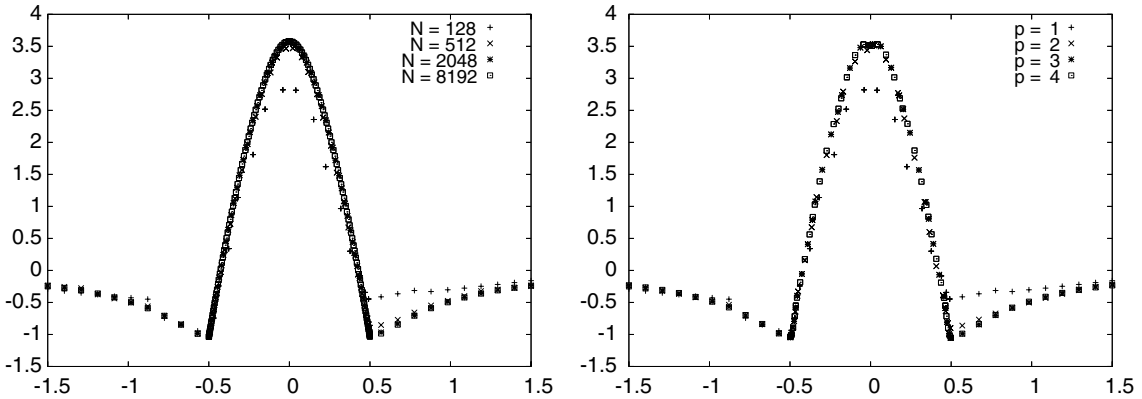

Fig. 12. Pressure coefficient on the surface under h-refinement with $p=1$ (left) and p-refinement on the coarsest mesh (right). Circular cylinder, $(p+1)$ points per surface edge plotted, N is the number of elements in a mesh.

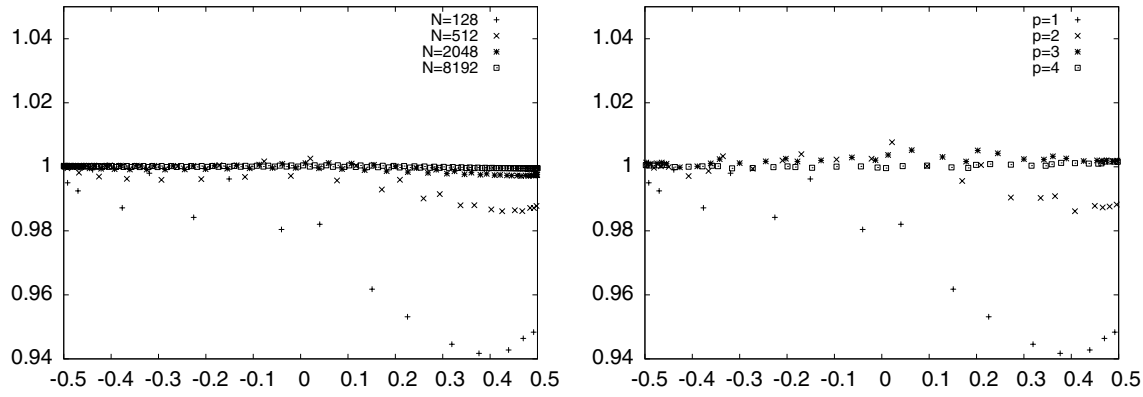

Fig. 13. Total pressure loss coefficient on the surface: h-refinement with $p=1$ (left) and p-refinement on the coarsest mesh (right). Circular cylinder, $(p+1)$ points per surface edge plotted, N is the number of elements in a mesh.

p-refinement, we notice that the solution on the finest mesh with $p=1$ and the solution on the coarsest mesh with $p=4$ are very close quantitatively and qualitatively. However, the number of degrees of freedom increases significantly faster under h-refinement (Fig. 14). As a result, the solution with $p=4$ on 16×4

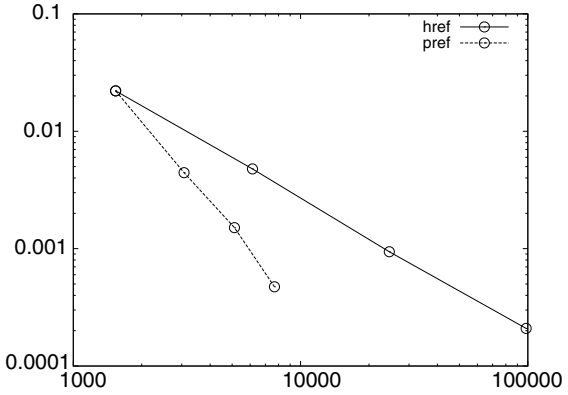

Fig. 14. L^2 error in total pressure on the surface as a function of degrees of freedom, h- and p-refinement, circular cylinder.

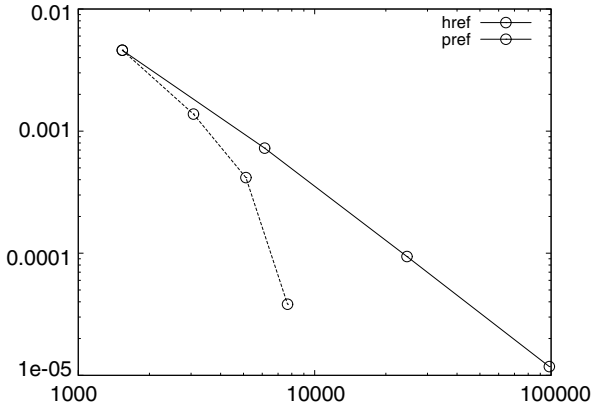

Fig. 15. Total mass loss as a function of degrees of freedom, h- and p-refinement, circular cylinder.

mesh required 40 times less CPU time than the linear approximation on the finest mesh. The advantage of the higher-order method is self-evident here.

We performed numerical experiments with flow around elliptic cylinders that produced qualitatively similar results. They are not reported here to save space.

5.3. Flow around NACA0012 airfoil

In contrast with the previous examples, where the exact boundary was described by circles, an error is introduced by the numerical approximation of the normals to the physical boundary for a NACA0012. It does not appear to affect the accuracy much. The only change that was made for this example was to make sure that one mesh point was located exactly at the end point of the airfoil. As already described in Section 4, the curvature of the two boundary elements containing the end point was computed using one-sided approximation, i.e., involving only one adjacent boundary edge lying on the same (upper or lower) half of the airfoil. A more sophisticated reconstruction of the geometry might be necessary for more complex cases; see for example [16].

The surface of the NACA0012 airfoil is given by

$$y = \pm 5t(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4), \tag{25}$$

with $t = 0.12$. We solve a subcritical problem with $p = 2$ on the 2960 element unstructured mesh with 102 elements on the surface shown in Fig. 16. The mesh is not symmetric with respect to the chord of the airfoil. The free stream Mach number is $M_\infty = 0.63$ and the angle of attack $\alpha = 2°$. The calculations were started with the uniform free stream and stopped when residuals reached the machine precision. The Mach and pressure coefficient isolines are shown in Fig. 17 with $\Delta M = 0.05$ and $\Delta C_p = 0.1$, the wall distributions of these quantities are presented in Fig. 18. The isolines around the trailing edge are smooth, without a cusp, implying insignificant spurious entropy production. The maximum entropy error is 1.3×10^{-3}, the entropy isolines are shown in Fig. 19. The total mass loss is 1.8×10^{-3} in absolute value. Next, we compute the lift and drag coefficients as

$$\begin{pmatrix} C_D \\ C_L \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} C_x \\ C_y \end{pmatrix}, \tag{26}$$

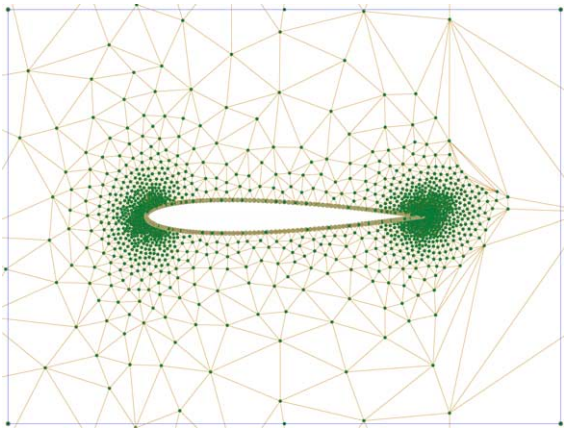

Fig. 16. Mesh around NACA0012.

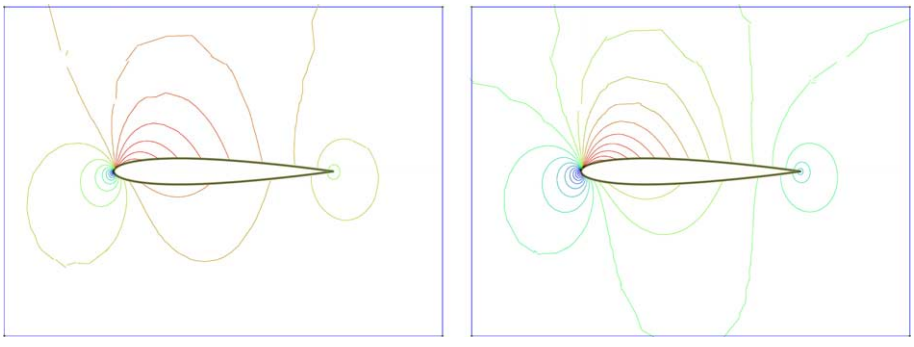

Fig. 17. Mach number (left) and $-C_p$ (right) isolines, NACA0012, $M_\infty = 0.63$, $\alpha = 2°$, $\Delta M = 0.05$, $\Delta C_p = 0.1$.

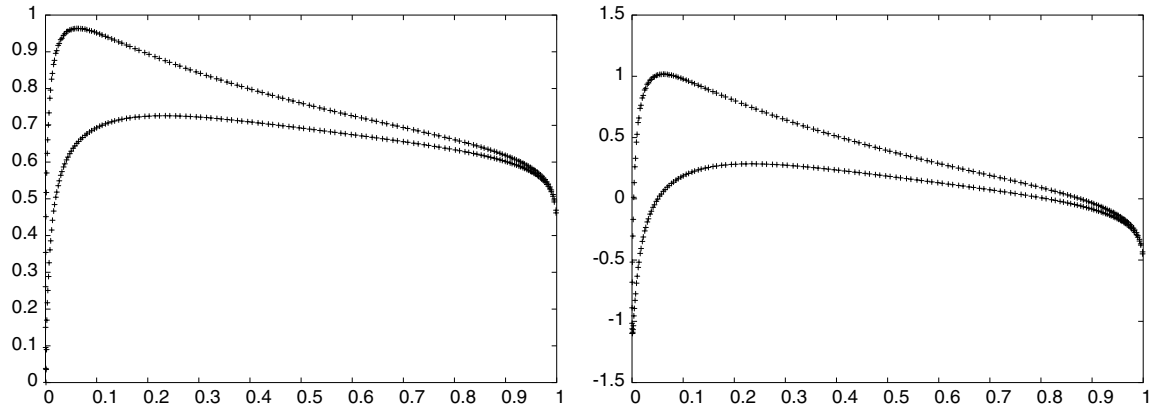

Fig. 18. Mach number (left) and $-C_p$ (right) on the surface of NACA0012, $M_\infty = 0.63$, $\alpha = 2°$.

Fig. 19. Entropy isolines, $M_\infty = 0.63$, $\alpha = 2°$.

where

$$C_x = \frac{\int_{\text{body}} Pn_1 \, ds}{0.5\rho_\infty \|\vec{v}_\infty\|^2 L}, \quad C_y = \frac{\int_{\text{body}} Pn_2 \, ds}{0.5\rho_\infty \|\vec{v}_\infty\|^2 L}, \tag{27}$$

and L is the chord length. The computed aerodynamic coefficients $C_L = 0.333$ and $C_D = 0.00015$ compare well with results in the literature [9].

Finally, we solve a transonic problem with the free stream Mach number 0.85 and the angle of attack 1°. The lift coefficient for this problem is known to be very sensitive and difficult to compute accurately. We computed the problem on the coarse grid containing 102 elements on the surface of the airfoil (Fig. 16).

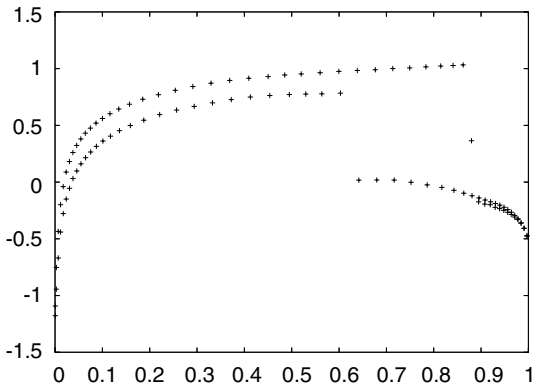

Fig. 20. Pressure coefficient $-C_p$, NACA0012 airfoil, $M_\infty = 0.85$, $\alpha = 1°$, 102 points on the surface of the airfoil.

The pressure coefficient on the surface is shown in Fig. 20. The lift and drag coefficients computed with a scalar limiter are $C_L = 0.373$ and $C_D = 0.0565$. They are similar to the results reported in [9].

6. Discussion

We proposed a new method for imposing solid wall boundary conditions for curved geometries. The method aims to approximate the flow around the physical rather than computational domain, thus eliminating large errors in the boundary layer. Ghost states are created at integration points on the solid boundaries; the solution values at these points are set so that the velocity vector resulting from solving the Riemann problem is tangent to the physical boundary. The curvature of the physical geometry is obtained directly from the geometric description of the body, if it is available, or approximated locally on each element using information from neighboring elements. We show on several examples that solutions obtained with the curvature boundary conditions converge under h- and p-refinement. The rate of convergence for the orders of approximation tested is $O(h^{p+1})$. Although the method is not conservative, the no flow through the wall condition is achieved in the limit, with the total flux through the solid boundary being small and converging to zero under refinement. The method does not depend on the order of approximation and as such can be very useful with p-refinement. It can be easily incorporated into an existing code since no special treatment of boundary elements is required and no construction of ghost cells is necessary.

In future work, we will apply this method to more complicated geometries and more complicated flow fields, for example time-dependent problems involving shocks. With these improved boundary conditions, p-refinement is now a more practical option for practical problems. Combining p- and h-refinement would be especially beneficial, and we believe is a fruitful area to investigate. Finally, extension to three-dimensional problems is an important next step.

Acknowledgments

The authors were supported in part by the Department of Energy Grant Nos. DE-FG02-00ER25053 and DE-FC02-01ER25472 and AFOSR Grant No. F19620-00-0099.

References

[1] M. Aftosmis, D. Gaitonde, T.S. Tavares, On the accuracy, stability and monotonicity of various reconstruction algorithms for unstructured meshes. AIAA-94-0415, January 1994.
[2] J. Anderson, Fundamentals of Aerodynamics, second ed., McGraw-Hill, 1991.
[3] H.L. Atkins, Continued development of the discontinuous Galerkin method for computational aeroacoustic applications, AIAA paper-97-1581, 1997.
[4] F. Bassi, S. Rebay, Accurate 2D Euler computations by means of a high order discontinuous finite element method, in: XIV International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, vol. 453, Springer, 1994, pp. 234–240.
[5] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, Journal of Computational Physics 138 (1997) 251–285.
[6] B. Cockburn, G. Karniadakis, C.-W. Shu (Eds.), Discontinous Galerkin Methods Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Berlin, 2000.
[7] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems, Journal of Computational Physics 141 (1998) 199–224.
[8] A. Dadone, Symmetry techniques for the numerical solution of the 2d Euler equations at impermeable boundaries, International Journal for Numerical Methods in Fluids 28 (1998) 1093–1108.

[9] A. Dervieux, B. van Leer, J. Periaux, A. Rizzi (Eds.), Numerical simulation of compressible Euler flows, Notes on Numerical Fluid Mechanics, vol. 26, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1989.
[10] J.E. Flaherty, L. Krivodonova, J.-F. Remacle, M.S. Shephard, Some aspects of discontinuous Galerkin methods for hyperbolic conservation laws, Finite Elements in Analysis and Design 38 (2002) 889–908.
[11] S. Karni, Hybrid multifluid algorithms, SIAM Journal on Scientific Computing 17 (1996) 1019–1039.
[12] G. Moretti, Importance of boundary conditions in the numerical treatment of hyperbolic equations, Physics of Fluids (1969) 13–20.
[13] P. Raj, B. Harris, Using surface transpiration with an Euler method for cost-effective aerodynamic analysis, AIAA paper-93-3506, 1993.
[14] A. Rizzi, Numerical implementation of solid boundary conditions for the Euler equations, Z.A.M.M. 58 (1978) T301–T304.
[15] P.L. Roe, Approximate Riemann solvers parameter, vectors and difference schemes, Journal of Computational Physics 43 (1981) 357–372.
[16] K. Siddiqi, B. Kimia, C.-W. Shu, Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution, Graphical Models and Image Processing 59 (1997) 278–301.
[17] E. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer, 1999.
[18] R. Wait, A.R. Mitchell, Finite Element Analysis and Applications, Wiley, Chichester, 1985.