Journal of Computational Science 4 (2013) 24–35

Contents lists available at SciVerse ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries

Ruibin Qin, Lilia Krivodonova*

University of Waterloo, Waterloo, Ontario, Canada

ARTICLE INFO

Article history:
Received 24 November 2011
Received in revised form 9 February 2012
Accepted 16 March 2012
Available online 3 April 2012

Keywords:
Discontinuous Galerkin method
Cartesian grids
Hyperbolic conservation laws
Euler equations
High-order methods

ABSTRACT

We present a discontinuous Galerkin method (DGM) for solutions of the Euler equations on Cartesian grids with embedded geometries. Cartesian grid methods can provide considerable computational savings for computationally intensive schemes like the DGM. Cartesian mesh generation is also simplified compared to the body fitted meshes. However, cutting an embedded geometry out of the grid creates cut cells. These are difficult to deal with for two reasons: the restrictive CFL number and irregular shapes. Both of these issues are more involved for the DG than for finite volume methods, which most Cartesian grid techniques have been developed for. We use explicit time integration employing cell merging to avoid restrictively small time steps. We provide an algorithm for splitting complex cells into triangles and use standard quadrature rules on these for numerical integration. To avoid the loss of accuracy due to straight sided grids, we employ the curvature boundary conditions. We provide a number of computational examples for smooth flows to demonstrate the accuracy of our approach.

1. Introduction

Cartesian grid methods provide an appealing approach for solving problems in complex geometries due to the relative simplicity of mesh generation which is often called the bottleneck of computational fluid dynamics simulations. Creating a high quality body-fitted mesh is a time-consuming process often requiring significant user input. Cartesian mesh generation consists of cutting a geometry from the Cartesian grid which is a more straightforward process and one that can be made more automated [2]. However, it creates the so-called cut cells where the Cartesian grid intersects the boundary of the immersed body. Cut cells may have irregular shapes, which makes integration on them difficult. In addition, the size of such cells can be magnitudes smaller than the size of regular grid cells. This results in a restrictive Courant–Friedrichs–Lewy (CFL) condition when an explicit time integration scheme is used. These two problems are major challenges in developing viable numerical methods for solving convection dominated problems on Cartesian grids.

A number of approaches have been proposed in the literature for dealing with the time step restriction. A particularly simple idea is to merge a small cut cell with its neighbors until a cell of a sufficient size is created [7,3,18]. Though conceptually simple, this method is difficult to implement robustly, especially for three-dimensional problems. This prompted development of other approaches such as *h*-box [12] and flux redistribution [8] methods for finite volume schemes. The *h*-box method works by reconstruction of the solution on a cut cell using the information contained in a box of the size of a regular cell. Flux redistribution allows only part of the flux to be used to compute the solution on a small cell. A number of finite difference methods seek to avoid time step restriction and to take into account the curvature of the boundary in treating boundary points, e.g. [9,20]. Finally, implicit time stepping is used in practical applications and for solving Navier–Stokes equations [17,10,15].

Computational savings provided by the structure of Cartesian grids can be especially beneficial for the computationally intensive discontinuous Galerkin (DG) methods. Nearly every aspect of evaluation of integrals over a regular cell's boundary and volume can be simplified by exploiting the regularity of the grid. For example, the mapping from physical elements to the canonical element is a simple scaling on Cartesian grids which can simplify and speed-up computing of gradients of the test functions. The issue that makes the Cartesian grid approach for the DGM particularly difficult is evaluation of the integral of the flux and a test function product in (3) on cut cells. The integration is usually performed using a numerical quadrature rule [13]. However, classical quadrature rules are known only for standard elements such as triangles and quadrilaterals. While generalized quadrature rules for arbitrary polygons can be computed, e.g. [16], creation of such rules is expensive (a few minutes for each polygon) and very sensitive to round-off errors. This makes them unsuitable for our applications. An interesting approach was proposed in [10]. They generate a quadrature rule

* Corresponding author.
E-mail address: lgk@math.uwaterloo.ca (L. Krivodonova).

1877-7503/$ – see front matter.
doi:10.1016/j.jocs.2012.03.008

for each cut cell by randomly choosing sampling points and computing weights that guarantee that basis functions are integrated exactly. The algorithm requires the number of integration points to exceed the optimal number (upwards of 400 for $p = 5$) and might suffer from a poor choice of points [15].

We propose to compute the cell volume integration by splitting each irregular cell into triangles and using a standard quadrature rule on each one of them. This has several advantages. Firstly, the quadrature rule is fast to generate and does not suffer from bad conditioning arising either because of the location of the integration points or the cell's shape. Secondly, it is more efficient as it does not involve storage of the values of basis functions at these points or reevaluation of them at each time step. The algorithm for splitting an arbitrary polygon into triangles is presented in Section 6. For an arbitrarily shaped cut or merged cell, it might, and often does, produce very thin triangles. This issue cannot be avoided without remeshing, i.e. moving vertices near the boundary, which is difficult to make robust. It is known that numerical approximation on such triangles can be poor [5]. For this reason, creating standard shape cells from cut cells is likely to be detrimental even when implicit time integration is used. However, it is easy to see and we show this in Section 6, that numerical quadrature does not suffer from thin element arising in splitting. For the same reason, i.e. to avoid ill-conditioning associated with mapping a badly shaped cell onto a canonical element, we define basis function on a rectangular bounding box containing a cut or merged cell.

We use explicit time integration and cell merging to avoid time step restriction imposed by small cut cells. The reasons for this are as follows. Implicit time integrators are prohibitively expensive for the DGM where the number of degrees of freedom increases quickly with the order of approximation, especially in three dimensions. Also, one of the main advantages of the DGM, i.e. the locality of approximation, is lost when a global system is constructed in implicit schemes. More involved techniques like the h-box and flux distribution methods are not straightforward to adapt to the discontinuous Galerkin framework. Both methods stabilize the solution on a small cell by modifying the flux on its edges. However, the DGM has a contribution from the integral over cell volume which needs to be stabilized as well.

The outline of this paper is as follows. We give a description of the general formulation of the RKDG method in Section 2. Sections 3 and 4 describe admissible cell types and the merging process. Imposing solid wall boundary conditions are described in Section 5. Integration over cut cells is described in Section 6. In section 7, we present numerical experiments which are followed by a discussion.

2. Discontinuous Galerkin formulation

In this paper, we consider the two-dimensional Euler equations describing compressible, inviscid flows

$$\partial_t \begin{pmatrix} \rho \\ \rho u_x \\ \rho u_y \\ E \end{pmatrix} + \boldsymbol{div} \begin{pmatrix} \rho u_x & \rho u_y \\ \rho u_x^2 + p & \rho u_x u_y \\ \rho u_x u_y & \rho u_y^2 + p \\ u_x(E+p) & u_y(E+p) \end{pmatrix} = \mathbf{0}. \tag{1}$$

Here, ρ is the density, u_x, u_y are components of the velocity, E is the internal energy and p is the pressure of the flow. For the ideal gas,

$$p = (\gamma - 1)\left[E - \rho\frac{u_x^2 + u_y^2}{2}\right].$$

We take $\gamma = 1.4$ for air. To describe the method, we rewrite (1) as a general conservation law

$$\partial_t \boldsymbol{u}(\boldsymbol{x}, t) + \boldsymbol{divF}(\boldsymbol{u}(\boldsymbol{x}, t)) = \mathbf{0}. \tag{2}$$

The computational domain Ω is then divided into a collection of elements

$$\Omega = \bigcup_{j=1}^{N} \Omega_j.$$

A Galerkin problem on element Ω_j is constructed by multiplying Eq. (2) by a test function $v \in \mathcal{H}^1(\Omega_j)$, integrating the result on Ω_j, and using the Divergence theorem to obtain

$$\int_{\Omega_j} v\partial_t \boldsymbol{u}\,\mathrm{d}\tau - \int_{\Omega_j} \boldsymbol{grad}\ v \cdot \boldsymbol{F}(\boldsymbol{u})\,\mathrm{d}\tau + \int_{\partial\Omega_j} v\boldsymbol{F}_n\,\mathrm{d}\sigma = \mathbf{0}, \tag{3}$$

$$\forall v \in \mathcal{H}^1(\Omega_j),$$

where $\partial\Omega_j$ is the boundary of Ω_j, $\boldsymbol{F}_n = \boldsymbol{F}(\boldsymbol{u}) \cdot \boldsymbol{n}$, and $\boldsymbol{n}$ is the outward facing normal vector. Several issues must be resolved before the above weak form can be used as a numerical method.

The solution $\boldsymbol{u}$ is approximated by a piecewise-polynomial function $\boldsymbol{U}$ of degree up to p,

$$\boldsymbol{U}_j = \sum_{i=1}^{N_p} \boldsymbol{c}_{ji}(t)\varphi_i(\boldsymbol{x}), \tag{4}$$

where N_p is the number of basis functions. The choice of basis functions is described in Section 6.2.

The integrals in (3) need to be evaluated by quadrature rules. We use the tensor product of one-dimensional Gauss–Legendre quadrature on regular cells. Integration on arbitrarily shaped cut cells is described in Section 6. Numerical flux is used to evaluate the integrand in line integrals.

After discretization in space, Eq. (3) can be rewritten in an ODE form as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{MC} = L_h(\boldsymbol{C}, t). \tag{5}$$

$\boldsymbol{C}$ represents the vector of the coefficients $\boldsymbol{c}_{ji}(t)$ ($i = 0, 1, \ldots, N_p$, j is the elemental index) of the approximate solutions, and $L_h(C, t)$ is the righthand side of the ODE consisting of $\int_{\Omega_j} \boldsymbol{grad} v \cdot \boldsymbol{F}(\boldsymbol{u})\,\mathrm{d}\tau$ and $\int_{\partial\Omega_j} v\boldsymbol{F}_n\,\mathrm{d}\sigma$ in (3). Due to the choice of basis functions, the mass matrix $\boldsymbol{M}$ is a multiple of the identity matrix on regular cells.

Eq. (5) is solved using an ODE solver. To preserve stability of the scheme, we use the strong stability preserving (SSP) explicit Runge-Kutta time discretization [19]. The size of a stable time step depends on the order of a discontinuous Galerkin approximation and the stability region of the chosen time integration scheme. For $p = 1, 2, 3$ and an s-stage, s-order Runge-Kutta method ($s = 2, 3$), the time step is given by

$$\Delta t \leqslant \min_j \frac{h_j}{(2p+1)|\lambda_j|}, \tag{6}$$

where h_j is a measure of the cell size, p is the order of the polynomial basis, and $|\lambda_j|$ is the maximum wave speed. More details on the particular implementation used here can be found in [11,6].

3. Mesh description

Generation of Cartesian grids is a more straightforward process than creation of body fitted meshes. A fast and robust algorithm that we largely follow here is described in, e.g. [2]. We assume that the background Cartesian grid resolves the geometry to a reasonable degree so that simulations can be carried out. However, we