2015-09

Mass Conservation of the Unified Continuous and Discontinuous Element-Based Galerkin Methods on Dynamically Adaptive Grids with Application to Atmospheric Simulations

Kopera, Michal A.; Giraldo, Francis X.

Mass Conservation of Unified Continuous and Discontinuous Element-based Galerkin Methods on Dynamically Adaptive Grids with Application to Atmospheric Simulations, Journal of Computational Physics, September 2015
https://hdl.handle.net/10945/45485

Mass Conservation of the Unified Continuous and Discontinuous Element-Based Galerkin Methods on Dynamically Adaptive Grids with Application to Atmospheric Simulations.

Michal A. Kopera[a,*], Francis X. Giraldo[a]

[a] *Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA 93940*

Abstract

We perform a comparison of mass conservation properties of the continuous (CG) and discontinuous (DG) Galerkin methods on non-conforming, dynamically adaptive meshes for two atmospheric test cases. The two methods are implemented in a unified way which allows for a direct comparison of the non-conforming edge treatment. We outline the implementation details of the non-conforming direct stiffness summation algorithm for the CG method and show that the mass conservation error is similar to the DG method. Both methods conserve to machine precision, regardless of the presence of the non-conforming edges. For lower order polynomials the CG method requires additional stabilization to run for very long simulation times. We addressed this issue by using filters and/or additional artificial viscosity. The mathematical proof of mass conservation for CG with non-conforming meshes is presented in Appendix B.

Keywords: adaptive mesh refinement, continuous Galerkin method, discontinuous Galerkin method, non-conforming mesh, compressible Euler equations, atmospheric simulations

1. Introduction

Both element-based Galerkin methods and adaptive mesh refinement for atmospheric modeling are active fields of research. In [1] we provide an overview of the literature covering both topics, with particular attention to the discontinuous Galerkin method. In this paper we extend that work by comparing the continuous (CG) and discontinuous Galerkin (DG) methods with non-conforming adaptive mesh refinement. As a metric for the comparison we chose the mass conservation, as it is an important feature for many atmospheric applications. We believe this is a good metric because, for smooth solutions, both methods will yield similar accuracy; it is unclear how both methods will compare for non-smooth solutions but it is expected that they can achieve similarly good results, especially with the inclusion of stabilization techniques for both methods. However, this topic is beyond the scope of this paper but is something to address in the future. Both methods are implemented in a unified way, such that they use the same data structures and routines for computations, differing only in the inter-element communication, which is described in detail in Sec. 2.2. This approach allows us to be confident that the differences we see in the results are due to the different methods rather than their implementations.

There is a vast collection of work in the literature on CG with geometrically non-conforming mesh refinement, starting with the early work of [2, 3] and later [4, 5]. Those approaches were based on the *mortar element method*, which employs integral projection to ensure a weak continuity across the non-conforming edges. In this paper we follow the approach by Fischer et al. [6], that uses an interpolation based method for reconciling the continuity condition on non-conforming edges. The two methods were compared in [7].

*Corresponding author. Tel: +1 831-656-3247
Email addresses: makopera@nps.edu (Michal A. Kopera), fxgirald@nps.edu (Francis X. Giraldo)

[8] used the interpolation approach for geophysical simulations. [9] compared the interpolation based non-conforming spectral element method with a finite volume method for the shallow water equations on the sphere. All these papers discuss the mathematical principles of the interpolation method, mentioning that the presented mathematical constructs (i.e. scatter matrix) are not built directly in the code but are rather applied. We present in this paper one possibility for the implementation of the interpolation method (Sec. 2.4). It relies on the concept of local and global data storage and can be further optimized for parallel implementations to avoid two communication steps. We hope that this algorithm will simplify the process of the implementation of the non-conforming CG method for interested readers.

The main focus of the paper is on the mass conservation of the CG and DG methods. Excellent mass conservation properties are often mentioned as a big advantage of the DG method over its continuous counterpart. Hughes et al. [10] refuted that statement arguing that the CG method is both globally and locally (with an additional post-processing step) conservative. In [11] they compared the CG and DG methods and reached the conclusion that the conservation properties of both methods (when using exact integration) are the same (with an additional post-processing step to obtain the local conservation for CG). On the other hand, [9] briefly reported the lack of conservation in the non-conforming CG method, comparing it with the conservative finite volume method. Admittedly, in their work the authors considered the CG method on the sphere and with the non-conforming mesh refinement, which leaves room for arguing whether or not those additional features affect the conservation. None of the aforementioned papers report a quantitative measure of the conservation of mass, energy, or any quantity for that matter. [12] (with a later follow-up in [13]) addressed the conservation issue of the conforming continuous Galerkin method on the cubed-sphere grid showing the mimetic properties of the HOMME-SE model and extends [10] to inexact integration used in the model. We address the question how non-conforming adaptive mesh refinement affects the mass conservation of both the CG and DG methods. In our study we focus on the numerical results, as they provide us with information regarding not only the method itself, but also the quality of the implementation. We focus on two-dimensional Euler test cases; however, exactly the same methods can be applied to 3D geometries. In Appendix B we also present a mathematical proof of conservation of the *direct stiffness summation* (DSS) operation, which characterizes the CG method. The value of this proof is that such a proof had not been shown previously for the case of CG with non-conforming meshes.

The paper is organized as follows: in Sec. 2 we outline the governing equations and give an overview of the unified CG/DG method implemented in the Non-hydrostatic Unified Model of the Atmosphere (NUMA) [14, 15]. We discuss the data structures and routines shared by both methods and point out the differences. We provide pseudocode for the non-conforming DSS operation for CG. In Sec. 3 we provide the results of the mass conservation study on two atmospheric test cases: the density current and rising thermal bubble. We conclude the paper in Sec. 4.

2. Methods

2.1. Governing equations

We use the conservative form of the Euler equations described in detail in [16, 1]. Here we provide the equation set for completeness. We write the Euler equations as follows

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \\ &\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + P\mathbf{I}) = -\rho g \mathbf{k}, \\ &\frac{\partial \rho \theta}{\partial t} + \nabla \cdot (\rho \theta \mathbf{u}) = 0, \end{aligned} \tag{1}$$

where ρ is the density, $\mathbf{u} = (u, w)^T$ is the velocity field, θ is the potential temperature, $\mathbf{I}$ is the rank-2 identity matrix, $\mathbf{k} = (0, 1)^T$ is the directional vector that points along the z direction, g is the gravitational acceleration and P is the pressure obtained from the equation of state

$$P = P_0 \left(\frac{R\rho\theta}{P_0} \right)^{\frac{c_p}{c_v}}, \tag{2}$$

where $R = c_p - c_v$ is the gas constant, P_0 is the pressure at the lower boundary, and c_v and c_p are the specific heats at constant volume and pressure respectively.

To stabilize the computations we add a diffusion term $\nabla \cdot (\mu\rho\nabla\mathbf{u})$ and $\nabla \cdot (\mu\rho\nabla\theta)$ to the right hand side of the momentum and potential temperature equations, respectively. The artificial viscosity μ is varied among the test cases. Note that NUMA is equipped to use either the local adaptive viscosity described in [17], or the dynamic sub-grid scale model for LES (Dyn-SGS) reported in [18]. However, to allow others to replicate our results, in this work we use a constant μ.

For simplicity we can write equation set (1) in vector form

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F} = S(\mathbf{q}), \tag{3}$$

where $\mathbf{q}$ is the solution vector, $\mathbf{F}(\mathbf{q})$ is the flux and $S(\mathbf{q})$ is the source term.

2.2. Overview of the CG and DG methods

The cornerstone of our comparison is the fact that the NUMA code incorporates both CG and DG methods within a unified framework [19]. This means that both methods use the same data structures and routines for integration, differentiation etc., which allows for a direct comparison of the key differences. We described the DG approach in detail in a previous publication (see [1]). Here we want to highlight the CG method and the details of its implementation in NUMA.

The Galerkin machinery applied to Eq. (3) gives

$$\int_{\Omega_e} \psi_i \frac{\partial \mathbf{q}^e}{\partial t} \,\mathrm{d}\Omega_e + \int_{\Gamma_e} n \cdot (\psi_i \mathbf{F}^*(\mathbf{q})) \,\mathrm{d}\Gamma_e - \int_{\Omega_e} \nabla\psi_i \cdot \mathbf{F}(\mathbf{q}^e) \,\mathrm{d}\Omega_e = \int_{\Omega_e} \psi_i S(\mathbf{q}^e) \,\mathrm{d}\Omega_e, \tag{4}$$

where Ω_e is the standard element, Γ_e is the boundary of that element and ψ_i is the test function. We replaced $\mathbf{F}$ in the boundary term by a numerical flux $\mathbf{F}^*$. In NUMA we use the Rusanov flux [20]. Note that all the terms are defined either on an element Ω_e or on its edge Γ_e. We can further transform this to a discrete matrix form using the expansion $\boldsymbol{q}^e \approx \boldsymbol{q}^e_N = \sum_{k=1}^{N_p} \psi_k \boldsymbol{q}^e_k$ (where N_p is the number of points in the element - for two-dimensional quadrilateral elements $N_p = (N+1)^2$, where N is the expansion order) and reorganize to keep only the time derivative term on the left-hand-side:

$$\boldsymbol{M}^e \frac{d\boldsymbol{q}^e_N}{dt} = \boldsymbol{M}^e S(\boldsymbol{q}^e_N) + (\boldsymbol{D}^e)^T \boldsymbol{F}(\boldsymbol{q}^e_N) - (\boldsymbol{M}^{s,e})^T \boldsymbol{F}^*(\boldsymbol{q}_N) \tag{5}$$

where $\boldsymbol{M}^e$ is the mass matrix, $\boldsymbol{M}^{s,e}$ is the side mass matrix, $\boldsymbol{D}^e$ is the differentiation matrix. A detailed definition of those matrices can be found in [1]. At this point it is important to mention that the mass matrix can be constructed such that it is diagonal (by using inexact integration, see [16]), and therefore trivial to invert.

Up until now we have made no distinction between the CG and DG method. The difference between the two comes from the fact that for CG we require that the solution to (3) be C^0. Note that (5) is defined on a single (standard) element. Only the numerical flux term provides a communication between the neighboring elements, hence the lack of ^{e} superscript. If, however, we assume that the solution is C^0 across the element boundaries, the numerical flux term will be canceled by opposite neighboring values at internal edges (on the physical boundaries, this term will remain). To ensure the continuity and communication between the elements in the CG method, we perform an operation traditionally called *direct stiffness summation* (DSS), which involves a mapping of element-local degrees of freedom to their global counterparts followed by summation (see [21] for details). The implementation of the DSS on the non-conforming interfaces will be presented in Sec. 2.3 along with the discussion of the data storage. Additional discussions regarding the DSS operation on non-conforming edges can be found in [8, 6, 7]. In [22] the authors introduce continuous global basis functions for non-conforming CG, which intrinsically maintain continuity in general situations in arbitrary dimensions.

To show how the CG and DG methods coexist in the NUMA code, let us assume that we use explicit time integration. In that case the terms on the right-hand-side of (5) will be evaluated at the current time

Algorithm 1 CG/DG algorithm to create RHS

```
function COMPUTE_RHS(q)
    for e = 1, N_e do                                   ▷ For all elements in the grid
        R^e ← COMPUTE_VOLUME_INTEGRALS(q^e)             ▷ both methods compute volume integrals
    end for
    if CG then                                          ▷ CG method
        R ← APPLY_DSS(R)                                ▷ applies DSS
    else
        for e = 1, N_e do                               ▷ DG method for all elements
            R^e ← R^e + COMPUTE_FLUX_INTEGRALS(q^e, q^ne)   ▷ computes flux to neighbors ne
            for i = 1, N_p do                           ▷ goes through nodal points in each element
                R_i^e ← R_i^e / M_i^e                   ▷ and divides by mass
            end for
        end for
    end if
    return R
end function
```

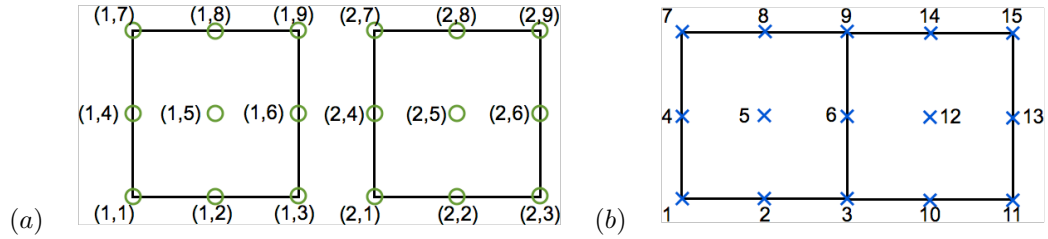

Figure 1: (a) local numbering of points (DG storage), (b) global numbering of points (global storage)

and used to advance $\boldsymbol{q}_N$ in time. Algorithm 1 shows a pseudocode for evaluating the right-hand-side (RHS), including inverting the mass matrix and left-multiplying by $(\boldsymbol{M}^e)^{-1}$. The COMPUTE_VOLUME_INTEGRALS routine evaluates the first two terms of the RHS of (5). This step is common to both CG and DG methods. If the CG method is chosen, the DSS is applied to the evaluated RHS. Otherwise the last term of (5) is computed by COMPUTE_FLUX_INTEGRALS, the result is added to the existing RHS and the sum is divided by the mass. In the case of the CG method the division by mass is handled within the APPLY_DSS routine.

Using this unified formulation of the CG/DG method we can ensure that the only difference between the solutions can come from the different handling of the communication between elements. In the DG method this is done by computing a flux, which is conservative by construction. In Appendix B we present the proof of conservation of the DSS operator used in the CG method; this proof is valid for both conforming and non-conforming meshes. In the following section we discuss the data storage we use in the code, which is closely related to the implementation of the DSS operation for the non-conforming element edges.

2.3. Data storage

In the NUMA code, data is stored in the so-called *DG storage*. This means that instead of storing variables as globally numbered vectors, we index the nodal solution by the element number and the element-local point number. This concept is graphically presented in Fig. 1. Panel (a) shows two elements with nodes numbered in a local DG way, while panel (b) presents global numbering traditionally used for the CG approach. Clearly, the DG storage has a bigger memory footprint, as the points which lie at the interface are numbered separately in each element. It is, however, the only possible storage for a DG method, as it assumes no continuity between the elements. For the CG method the overlapping points at the element edges will hold the same value, even in the DG storage approach.

Fischer et. al. [6] explain the procedure for moving the data between global (CG) and local (DG) storage. In their formulation a Boolean connectivity matrix $\boldsymbol{Q}$ maps global values of $\boldsymbol{u}^{CG}$ to locally stored $\boldsymbol{u}^{DG}$. $\boldsymbol{Q}^T$ sums the local values of $\boldsymbol{u}^{DG}$ from the corresponding nodes and stores the result in a global vector $\boldsymbol{v}^{CG}$. Note that $\boldsymbol{v}^{CG} \neq \boldsymbol{u}^{CG}$, instead we have the following relations

$$\boldsymbol{u}^{DG} = \boldsymbol{Q}\ \boldsymbol{u}^{CG}, \qquad \boldsymbol{v}^{CG} = \boldsymbol{Q}^T \boldsymbol{u}^{DG}. \tag{6}$$

While we do not need to use the $\boldsymbol{Q}$ operation in the DG code, it will be essential in the CG part for the direct stiffness summation. In NUMA this operation is performed in three steps outlined in Algorithm 2. In the code, the matrix $\boldsymbol{Q}$ is never constructed. Instead, we evaluate its action, as well as the action of its transpose. The details of the DSS operation for the non-conforming interface will be discussed in the following section.

Algorithm 2 Direct stiffness summation

function APPLY_DSS($\boldsymbol{u}^{DG}$)
  $\boldsymbol{v}^{CG} \leftarrow \boldsymbol{Q}^T \boldsymbol{u}^{DG}$ ▷ Gather data to CG storage
  for $j = 1, N_g$ **do** ▷ go over all global points
    $u_j^{CG} = v_j^{CG} / m_j^{CG}$ ▷ divide by $\boldsymbol{m}^{CG} = \boldsymbol{Q}^T \boldsymbol{m}^{DG}$, where $\boldsymbol{m}^{DG}$ is a vector of diagonals of $\boldsymbol{M}^e$ for all elements e
  end for
  $\boldsymbol{u}^{DG} \leftarrow \boldsymbol{Q}\ \boldsymbol{u}^{CG}$ ▷ scatter back to DG storage
  return $\boldsymbol{u}^{DG}$
end function

Since the matrix $\boldsymbol{Q}$ is a Boolean map between the local and global node numbering, we implement this map as an interconnectivity array `ig = intma(e,ip)`, where `intma` stands for the INTerconnectivity MAtrix, `ig` is a global node number, `e` is the element number and `ip` is a local point number. The actions of $\boldsymbol{Q}$ and $\boldsymbol{Q}^T$ for conforming elements are detailed in Alg. 3 lines 1-6.

2.4. Non-conforming edges

Non-conforming edges arise when an edge is shared by more than two elements. Fig. 2 shows (a) local and (b) global numbering of nodes in such a situation for linear elements. In panel (b) the red circle (labeled "8") marks a *hanging node*. The value of the solution there is constrained by the C^0 continuity requirement of the CG method. The DSS operation satisfies this constraint in the non-conforming element situation by an appropriate construction of the $\boldsymbol{Q}$ matrix. It is described in detail in [6]. Since it is not constructed in the code, we outline here the implementation of its action in the non-conforming case. The algorithm for the DSS operation is the same for both conforming and non-conforming cases.

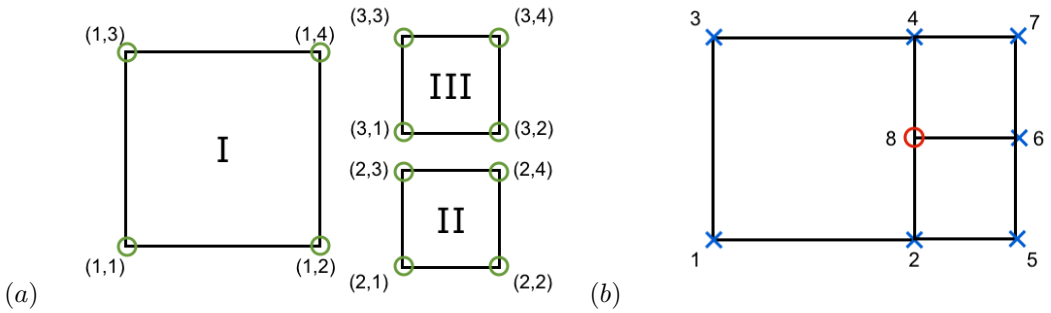

Figure 2: Non-conforming elements: (a) local numbering of points (DG storage), (b) global numbering of points (CG storage). In global numbering, point no. 8 is a so called *hanging node*, as its value is constrained by the continuity condition between the three elements.

Algorithm 3 Scatter and gather operations $\boldsymbol{Q}$ and $\boldsymbol{Q}^T$ in a non-conforming case

$\boldsymbol{Q}$ action (scatter):	$\boldsymbol{Q}^T$ action (gather):
	$\boldsymbol{v}^{CG} = 0$
for e=1,N_e **do**	**for** $e = 1, N_e$ **do**
for ip=1,N_p **do**	**for** $ip = 1, N_p$ **do**
ig = intma(e,ip)	ig = intma(e,ip)
u^{DG}(e,ip) = u^{CG}(ig)	v^{CG}(ig) = v^{CG}(ig) + u^{DG}(e,ip)
end for	**end for**
end for	**end for**
for $\forall$ f $\in$ NCE **do**	**for** $\forall$ f $\in$ NCE **do** ▷ for non-conforming edges
p $\leftarrow$ parent(f)	p $\leftarrow$ parent(f) ▷ parent element
c1 $\leftarrow$ child(f,1)	c1 $\leftarrow$ child(f,1) ▷ child element #1
c2 $\leftarrow$ child(f,2)	c2 $\leftarrow$ child(f,2) ▷ child element #2
for i=1,ngl **do**	**for** i=1,ngl **do**
ip=imap(f,p,i)	ic=imap(f,c1,i) ▷ find nodes on f
ig=intma(p,ip)	$u_1(i) \leftarrow u^{DG}$(c1,ic) ▷ get values
v(i) $\leftarrow u^{CG}$(ig)	$u_2(i) \leftarrow u^{DG}$(c2,ic) from edge nodes
end for	**end for**
$\boldsymbol{u}_1 = \boldsymbol{L}_1 \boldsymbol{v}$	$\boldsymbol{v}_{1+2} = \boldsymbol{L}_1^T \boldsymbol{u}_1 + \boldsymbol{L}_2^T \boldsymbol{u}_2$ ▷ interpolate & sum
$\boldsymbol{u}_2 = \boldsymbol{L}_2 \boldsymbol{v}$	
	ic=imap(f,c1,1) ▷ Correct corner #1
	ig=intma(ic)
	v^{CG}(ig) = v^{CG}(ig) − u^{DG}(c1,ic)
	ic=imap(f,c2,ngl) ▷ Correct corner #2
	ig=intma(ic)
	v^{CG}(ig) = v^{CG}(ig) − u^{DG}(c2,ic)
for i=1,ngl **do**	**for** i=1,ngl **do**
ic=imap(f,c1,i)	ip=imap(f,p,i)
u^{DG}(c1,ic) = u_1(i)	ig=intma(p,ip)
u^{DG}(c2,ic) = u_2(i)	v^{CG}(ig) = v^{CG}(ig) + v_{1+2}(i)
end for	**end for**
end for	**end for**

Algorithm 3 provides pseudocode for the routines evaluating the action of $\boldsymbol{Q}$ (gather operation - left) and $\boldsymbol{Q}^T$ (scatter operation - right). The first six lines perform a typical conforming gather/scatter operation using the `intma` array. In the non-conforming case we need to introduce adjustments to accommodate the hanging nodes. In line 7 the index `f` loops over all non-conforming edges. In lines 8-10 we identify the parent and children elements. The parent element is the large element sharing the edge, while the children are the small elements on the other side of this edge. The naming convention comes from the fact that the parent element will ensure continuity, which will be enforced on the children edges. In Fig. 2 the parent is the element I, while the children are the smaller elements II, III. Indices `p`, `c1` and `c2` correspond to appropriate element numbers (I, II and III respectively).

The section of the algorithm in lines 7-10 is responsible for creating a vector of edge values for both scatter ($\boldsymbol{v}$) and gather ($\boldsymbol{u_1}, \boldsymbol{u_2}$) operations. Throughout the algorithm N_e is the number of elements, `ngl` is the number of points along an element edge and N_p is the number of points inside each element (for quadrilateral elements with tensor-product basis functions $N_p =$ `ngl` $\times$ `ngl`). The array `imap` finds the element local number of the *ith* point that belongs to a given element (`p,c1,c2`) and lies on the edge `f`. In the example of Fig. 2 $\boldsymbol{v} = [u_2^{CG}, u_4^{CG}]$ and for the gather operation $\boldsymbol{u}_1 = [u^{DG}_{(2,1)}, u^{DG}_{(2,3)}]$ and $\boldsymbol{u}_2 = [u^{DG}_{(3,1)}, u^{DG}_{(3,3)}]$.

In lines 16 and 17 for the scatter operation we perform the interpolation of the vector $\boldsymbol{v}$ from the parent side of the edge to two children vectors $\boldsymbol{u}_1$, $\boldsymbol{u}_2$. The matrices $(L_1)_{ij} = \psi_i(\xi_j^{\mathtt{c1}})$ and $(L_2)_{ij} = \psi_i(\xi_j^{\mathtt{c2}})$ are the interpolation matrices from the parent side of the edge to each of the two children sides respectively. Their entries are the values of the 1D parent edge basis functions ψ_i at the children side nodal points $\xi_j^{\mathtt{c1,c2}}$. In Fig. 2 example $\boldsymbol{\xi}^{\mathtt{c1}} = [\xi_{(2,1)}, \xi_{(2,3)}]$ and $\boldsymbol{\xi}^{\mathtt{c2}} = [\xi_{(3,1)}, \xi_{(3,3)}]$ or, using global node numbering, $\boldsymbol{\xi}^{\mathtt{c1}} = [\xi_2, \xi_8]$, $\boldsymbol{\xi}^{\mathtt{c2}} = [\xi_8, \xi_4]$. The ξ coordinate is measured in the parent element coordinate system. All the interpolations are evaluated on the standard element in the computational space $\xi \in [-1, 1]$, therefore, the matrices $\boldsymbol{L}_{1,2}$ are computed only once and used for all elements.

For the gather operation, in line 16 we use the transpose of the interpolation matrix to compute the contributions from the children elements to the parent side of the edge. The contributions are summed, but before they can be applied to the solution gathered in the CG storage, we need to account for the fact that some contributions from the corner nodes were already added in the conforming step. In Fig. 2 example, data from points $(2, 1)$ and $(3, 3)$ are gathered to global points 2 and 4 respectively. The summed contributions $\boldsymbol{v}_{1+2}$ account for contributions of local points $(2, 1), (2, 3), (3, 1), (3, 3)$ to points 2 and 4 in global storage. We avoid double counting by subtracting $u^{DG}_{(2,1)}$ and $u^{DG}_{(3,3)}$ from u_2^{CG} and u_4^{CG} respectively, which is executed in lines 18-23 of the gather operation. There are other possibilities for executing these operations but, we feel, this is the most sensible way since the standard conforming DSS operation is first used with the correction for non-conformity.

In lines 24-29 of the scatter operation we copy the interpolated vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ to the local storage. For the gather operation the summed solution $\boldsymbol{v}_{1+2}$ is added to appropriate locations in the global CG storage. This concludes both operations.

In the non-conforming case, the DSS algorithm outlined in Alg. 2 remains unchanged. The only difference lies in the construction of the $\boldsymbol{Q}$ matrix. In the example shown in Fig. 2, we first perform the gather operation, to account for the contributions of nodes $(2, 1), (2, 3), (3, 1), (3, 3)$ to global nodes $2, 4$. Once complete, we divide the gathered solution by the gathered mass matrix. Since both the solution and the mass matrix are gathered in the same way, this division is equivalent to a weighted average of multiple contributions from local nodes to a single global node. The weight of each local node is its corresponding mass matrix entry. The final DSS step is to scatter the data back to the local storage using the $\boldsymbol{Q}$ matrix. Since the scatter operation is essentially an interpolation, this procedure ensures that the solution is continuous across the non-conforming interfaces.

A good check for the correct implementation of the $\boldsymbol{Q}$ matrix is to construct it explicitly for a small problem and inspect its entries[1]. For comparison, in Appendix A we provide the $\boldsymbol{Q}$ matrix for both conforming and non-conforming examples presented in Fig. 1 and Fig. 2. See also [8] for a similar analysis.

[1]We are grateful to Lucas Wilcox for suggesting this idea.

2.5. Computation of the mass

One of the objectives of this paper is to discuss the mass conservation properties of the continuous and discontinuous Galerkin methods with non-conforming mesh refinement. We measure the mass conservation error as follows:

$$M(t) = \frac{|m(0) - m(t)|}{m(0)}, \tag{7}$$

where $m(t) = \int_\Omega \rho(t)\,\mathrm{d}\Omega$ is the total mass of the system at time t. The way the total mass $m(t)$ is computed is important. As explained in [1], simple sequential summation of mass contributions from all the nodal points can result in a considerable roundoff error, which blurs the outcome of the mass conservation study. In order to control the roundoff error in the mass computation, we use a pairwise summation algorithm outlined in [23][2].

3. Results

3.1. Test cases

In this study we use two standard atmospheric test cases: the density current [24] and the rising thermal bubble [16]. They both display features that move dynamically across the entire domain, which allows for using adaptive mesh refinement (AMR) effectively. The difference between the cases is that the rising thermal bubble allows for using larger Courant numbers[3]. Following the findings of the previous paper ([1]) we use an IMEX ARK3 (3^{rd} order additive Runge-Kutta method) time integrator for its robustness with AMR simulations. For both test cases, we use artificial viscosity to ensure stability of the simulations. We run both tests for a much longer time than usual to be able to reliably report the mass conservation properties. The cases are identical, apart from the simulation time, to the ones used for our previous study [1]. Here we outline them for the sake of completeness.

3.1.1. Case 1: Density current

The case consists of a bubble of cold air dropped in a neutrally stratified atmosphere. The bubble eventually hits the lower boundary of the domain (no flux wall) and moves horizontally shedding Kelvin-Helmholtz rotors 4. In order to obtain the grid-converged solution we apply artificial viscosity $\mu = 75\mathrm{m}^2/\mathrm{s}$.

The initial condition is defined in terms of potential temperature perturbation

$$\theta' = \begin{cases} 0 & \text{for} \quad r > r_c \\ \frac{\theta_c}{2}\left(1 + \cos\left(\frac{\pi r}{r_c}\right)\right) & \text{for} \quad r \leq r_c \end{cases}, \tag{8}$$

where $\theta_c = -15\mathrm{K}$, $r = \sqrt{\left(\frac{x-x_c}{x_r}\right)^2 + \left(\frac{z-z_c}{z_r}\right)^2}$ and $r_c = 1$. The domain is defined as $(x, z) \in [0, 25600]\mathrm{m} \times [0, 6400]\mathrm{m}$ and the center of the bubble is at $(x_c, z_c) = (0, 3000)\mathrm{m}$ with the axes of the bubble defined by $(x_r, z_r) = (4000, 2000)\mathrm{m}$. The boundary conditions for all four boundaries are no-flux walls, i.e. $\mathbf{u}|_\Gamma = 0$. The velocity field is initially set to zero everywhere. In order to obtain credible mass conservation data we run the simulation up to $t = 10000\mathrm{s}$, which is over 10 times longer than the usual final time for this test case (typically $t = 900\mathrm{s}$). This time is enough for the bubble to traverse the length of the domain three times.

[2]We are grateful to Jeremy Kozdon for this suggestion.

[3]The reason for this is due to the disparity of the size of the acoustic waves compared to the other waves in the system. Since we use IMEX (implicit-explicit) methods, we can use time-steps that allow us to step completely over the acoustic waves.

3.1.2. Case 2: Rising thermal bubble

In this test case a warm bubble rises in a constant potential temperature atmosphere (the reference potential temperature $\bar{\theta} = 300K$, $\theta = \bar{\theta} + \theta'$). As it rises, it deforms until it forms a mushroom shape and hits the top wall at a later time. Initially, the air is at rest and in hydrostatic balance. The initial potential temperature perturbation is given by Eq. (8) with $\theta_c = 0.5$K and $r_c = 250$m. The domain has dimensions $(x, z) \in [0, 1000]$m $\times [0, 1000]$m and the bubble is positioned at $(x_c, z_c) = (500, 350)$m. The boundary conditions for all sides are no-flux. The simulation runs until $t = 5000$s. In this case we try to use as little artificial viscosity as possible, just enough to stabilize the simulation. We provide the discussion on the artificial viscosity used in Sec. 3.2.2.

3.2. Mass conservation

3.2.1. Case 1 - density current

Figure 3 shows the time evolution of the mass conservation error for Case 1 on different mesh configurations. Fig. 3a presents the results for a uniform mesh of 128×32 elements with polynomial orders of 5 and 7 (blue and red markers, respectively), and 64×16 elements with polynomial order 10 (green markers). In this way the simulations for polynomial order 5 and 10 (nop=5 and nop=10) have the same effective resolution, while the nop=7 simulations have slightly increased resolution. For each polynomial order, the results are reported using both CG and DG (circular and triangular markers, respectively). The dashed black line shows the machine precision level of 10^{-16}.

The uniform mesh test is designed to be a benchmark against which we will test the mass conservation properties of non-conforming simulations. Figure 3a shows that for all the simulations the mass error is contained within 10^{-15}. Moreover, the mass conservation error is virtually identical for both methods for all polynomial orders.

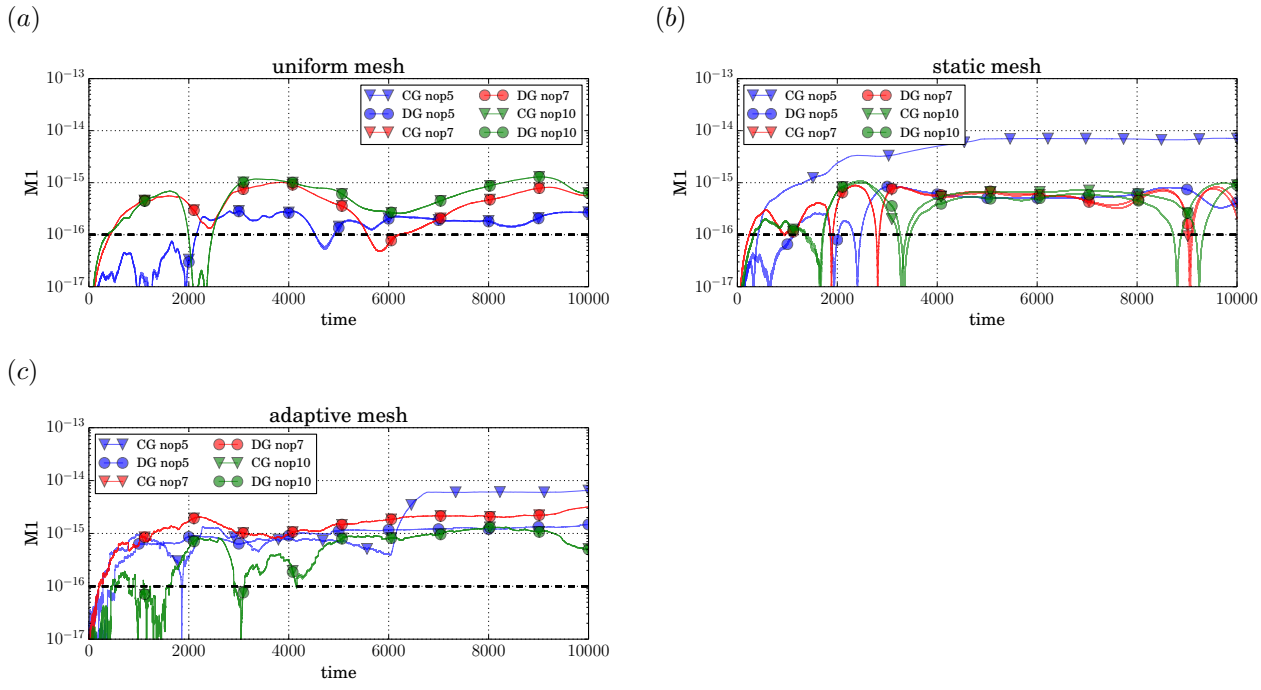

Figure 3: Case 1: Mass conservation as a function of simulation time for (a) uniform mesh, (b) statically refined mesh, (c) dynamically adaptive mesh. Triangular markers denote the CG results and circular markers represent the DG results. Colors represent different polynomial orders.

The static mesh test is designed to check how the mass is conserved when the cold front is moving through the non-conforming mesh interfaces. The mesh is refined near the vertical domain boundaries and coarsened in the middle (Fig. 4). The intermediate region is created following the 2:1 balancing constraint described in [1]. The results are reported in Fig. 3b using the same color and marker convention as for the uniform test. Similarly as in the uniform simulations, the mass conservation error remains under 10^{-15} for

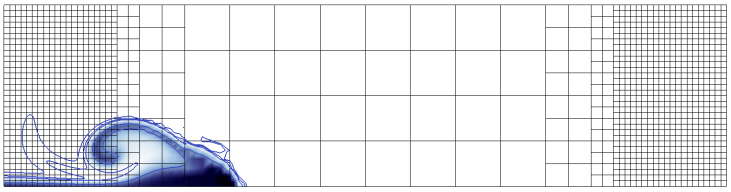

Figure 4: Static mesh for the density current case. The mesh is refined near the vertical edges of the domain to a 5^{th} level mesh (32 elements in the vertical direction, which corresponds to the resolution of the reference simulation). The middle of the domain is kept at 2^{nd} level mesh (coarsened three times to 4 elements in the vertical dimension). The results for 10^{th} order polynomials are obtained on the mesh one level coarser for both refined and coarse regions to keep the effective resolution comparable between 5^{th} and 10^{th} order. The color contours show the potential temperature perturbation.

the majority of the tests. The worst exception is the CG nop=5 simulation, where the mass conservation error is approximately a level of magnitude higher. Apart from this result, other polynomial orders show similar, albeit not identical, mass conservation properties between CG and DG.

Fig. 3c presents the results of dynamically adaptive simulations, where the refined mesh tracks the areas where the absolute value of potential temperature perturbation exceeds 0.1K. Comparing to the static mesh simulations, here the mass conservation error is slightly increased but still bounded in time. The only simulation with significantly different behavior is, again, the CG nop=5 simulation. For all the other cases the CG and DG methods yield virtually the same mass errors.

For all three results presented in Fig. 3 it is important to bear in mind that we operate at error levels at or near machine precision. For that reason it is difficult to make definitive statements about which result is better, unless the difference is an order of magnitude. Even though great care was taken to make sure the mass conservation is computed accurately, we cannot guarantee it is free of roundoff error (see [1]). All the results have a similar level of mass error, therefore we can conclude that at this level of precision the AMR simulations have similar conservation properties as the uniform simulations. There is no significant difference between the CG and DG methods, even though the treatment of the non-conforming interface is rather different. The only exception is the nop=5 non-conforming simulation (both for static and adaptive mesh), which shows different behavior between the CG and DG methods. Since this behavior was consistent between static and adaptive mesh simulations, it may be an indication that lower order non-conforming meshes with CG may be less robust with regards to mass conservation than their DG counterparts. It is worthwhile to mention here that we use inexact integration, which may be the cause of this issue. [25] shows that in the conforming mesh case one should not go below nop=4 to avoid inexact integration error. The presence of the non-conforming edges may amplify this effect for the CG method. Another explanation of this issue may be that even though the nop=5 and nop=10 simulations have the same effective resolution, the higher order method in fact resolves the problem better. It is possible that the nop=5 simulation with mesh refinement has marginal or too low resolution at certain times in this problem and this manifests itself in the mass conservation error being slightly higher.

For adaptive and uniform meshes we have run a series of simulations with different effective resolutions to check if the mass conservation properties are affected by the mesh resolution. In Fig. 5 we plot the time-average of the mass conservation error against the effective resolution of the simulation for different polynomial orders (colors) and methods (markers). The results for the adaptive mesh show slightly higher mass conservation error. In both cases there is no visible effect of the resolution on the average mass conservation error.

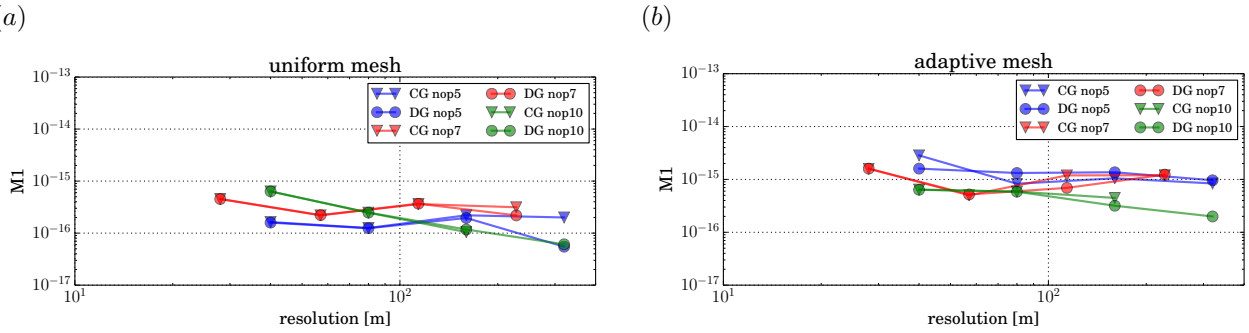

Figure 5: Case 1: Mass conservation against the effective resolution for (a) uniform mesh and (b) adaptive mesh. Triangular markers denote the CG results and circular markers represent the DG results. Colors represent different polynomial orders.

3.2.2. Case 2 - rising thermal bubble

The density current test case discussed in Sec. 3.2.1 (Case 1) is characterized by very high viscosity, which was introduced to guarantee convergence (see [24]). In order to investigate the mass conservation errors for lower artificial viscosity levels, we use the rising thermal bubble test. For usual simulation times of t = 700s and effective resolutions in the range of single meters, the value of viscosity was typically set to $\mu = 0.1\text{m}^2/\text{s}$ (see [16, 26]). In this study, unless otherwise stated, we chose $\mu = 0.5\text{m}^2/\text{s}$ to stabilize the adaptive bubble simulations until $t = 5000$s.

Figure 6a presents the reference uniform simulation results. The mesh consists of 32×32 elements with two different polynomial orders (nop=5 blue markers and nop=7 red markers). Both polynomial orders show the mass conservation error limited by 10^{-14}. The results are identical for both the CG and DG methods.

For the adaptive mesh simulations, in addition to polynomial orders 5 and 7 we have run a nop=10 simulation with the smallest element size corresponding to the elements of the uniform 16×16 mesh. This gives the same effective resolution as the 32×32 nop=5 simulation. Figure 6b shows the comparison of the adaptive mesh results for the CG and DG methods (triangular and circular markers, respectively) for different polynomial orders. For all the results the mass conservation error is below 10^{-14}. Unlike in the previous test case, the results are not identical between the DG and CG methods for all polynomial orders[4]. For CG with polynomial order 5 (blue triangles), the simulation did not complete successfully (for 5000 seconds) for the artificial viscosity $\mu = 0.5\text{m}^2/\text{s}$.

Bearing in mind that in the previous test case (Case 1), the CG simulations with nop=5 behaved differently than the rest, we increased the viscosity to $\mu = 5.0\frac{m^2}{s}$ (we tried a range of viscosity values from 0.1 to $5.0\text{m}^2/\text{s}$ and only the highest one resulted in a successfully completed simulation at 5000 seconds - note that all the tests were successful at the usual final time of $t = 700$s, regardless of the viscosity). The result of this simulation is reported in Fig. 6c. With significantly larger viscosity the nop=5 CG simulation (red triangles) completed and had mass conservation properties consistent with the results presented in panel (b). The DG simulation with increased viscosity (red circles) behaved similarly, yet not identically, to the lower viscosity one (blue circles).

An alternative method of stabilization is to use a filter which removes high frequency oscillations from the solution. We designed an experiment where for polynomial order 5 simulations with viscosity $\mu = 0.5\frac{m^2}{s}$ we use the Boyd-Vandeven filter (see [27, 16]). By design this filter is not supposed to change the mean of the filtered field, trimming only the higher order oscillations. The result is presented in Fig. 6d. The filtered CG simulation (red triangles) completed for lower viscosity, while the mass conservation error for

[4]The different behavior between CG and DG is not unexpected for this case. The initial condition used for both tests is non-smooth (not infinitely differentiable) but the key difference is that the large amount of viscosity used in the previous case was sufficient to suppress discontinuities whereas in this case (due to the small amount of diffusion) it is not.

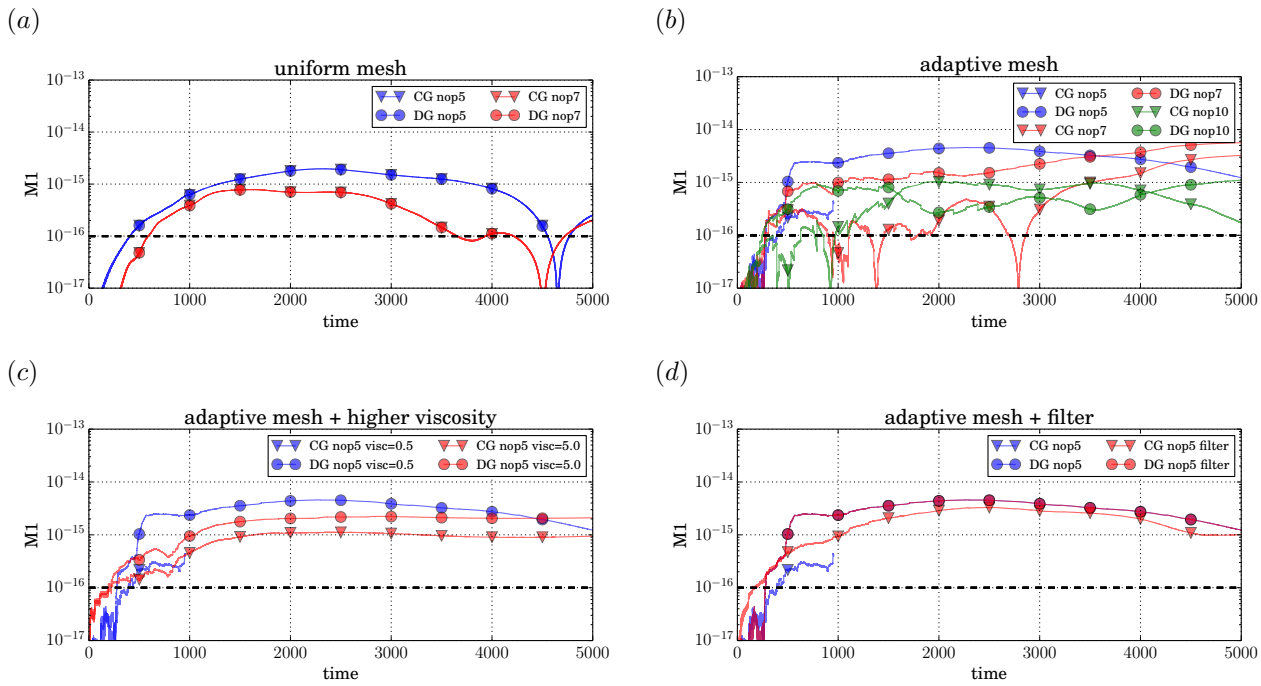

Figure 6: Case 2: Mass conservation error as a function of time for (a) uniform mesh, (b) adaptive mesh, (c) adaptive mesh with increased viscosity and (d) adaptive mesh with filtering. Triangular markers denote the CG results and circular markers represent the DG results. Colors represent different polynomial orders (panel (a) and (b)), different viscosity parameter (c) or the use of filtering (d).

the DG method remained unchanged (blue and red circles overlap). We have to report, however, that for the original viscosity of $\mu = 0.1\frac{m^2}{s}$ the CG nop=5 simulation did not complete even with the filter (for $t = 5000$s). It is not too surprising, perhaps, since the CG method has no inherent stabilization while our DG uses a Rusanov numerical flux, which is highly dissipative and thus more robust.

To further investigate the stability issue discovered in Fig. 6 we ran a set of simulations with various artificial viscosity values and different polynomial orders (without a filter). The simulation times of those runs are reported in Table I, where the final time of 5000s (in bold type) marks a successfully completed simulations, and any smaller value (italic) represents a time at which the simulations broke down [5]. All

[5]In our case the simulation break-down happened by either the simulation producing an NaN value or GMRES exceeding the iteration count limit.

Table I: Case 2: CG simulation time (in seconds) for different polynomial orders and artificial viscosity parameters. Bold numbers symbolize simulations that completed successfully. Values in italic mark the time at which the simulations broke-down.

$\mu[\frac{m^2}{s}]$	nop=5	nop=6	nop=7
0.1	*716*	*897*	*1110*
0.5	*956*	*994*	**5000**
2.0	*1034*	**5000**	**5000**
3.0	*1578*	**5000**	**5000**
5.0	**5000**	**5000**	**5000**

the results were obtained for the CG method with dynamic AMR for Case 2, for the DG method all the simulations completed successfully. Note that for the usual final time of $t = 700$s all CG simulations would be considered successful as well. It is clear that with increased artificial viscosity, the simulations become more stable. As a general rule we can say that for this case all the simulations with polynomial order greater than or equal to 6 and artificial viscosity of at least $\mu = 2.0\text{m}^2/\text{s}$ are stable until $t = 5000$s. There seems to be a critical time after which the simulation never breaks down. The times in Table I indicate that this critical time is around $t \approx 1000 - 1500$s, which coincides with the bubble impinging on the top domain boundary. It is possible that the complexity of the flow impinging at the domain boundary is not fully resolved by certain choices of mesh and polynomial order. High artificial diffusion stabilizes the solution even for under-resolved flows.

The question that naturally arises is why is DG more robust than CG for AMR with low viscosity and lower polynomial order? We believe the reason is due to the additional dissipation mechanism offered by the Rusanov flux. Recall that this flux can be written as

$$F^* = \{F^{(e,k)}\} - |\lambda|\hat{n}[q^{(e,k)}], \tag{9}$$

where { } denotes the mean value and [ ] is the jump in standard DG notation. The jump term is essentially an adaptive viscosity that gets stronger with large jumps (the difference in the solutions on each side of the edge). Therefore, to make the comparison more fair between CG and DG would require the inclusion of the additional adaptive viscosity to the CG scheme. We reserve the study of how best to construct such a dissipative mechanism for future work.

4. Conclusion

In this paper we have investigated the mass conservation properties of a unified CG/DG method with non-conforming adaptive meshes using two standard atmospheric test cases. The unification of the CG and DG method allowed us to compare directly the different treatment of the non-conforming interfaces, because every other aspect of the code executes the same routines for both methods.

The study showed that the mass conservation error is not significantly affected by the non-conforming mesh compared with the uniform mesh. At the levels of error near machine precision we cannot be sure if the small difference we see in the plots comes from the presence of the non-conforming faces, the roundoff error in the algorithm, or the roundoff error committed while computing the mass. We attempted to minimize the mass computation error by using a pairwise sum algorithm and have included a mathematical proof on the conservation of mass for CG with non-conforming meshes.

The difference between the CG and DG mass conservation error is small and due to a different treatment of the non-conforming edges. For conforming uniform meshes there is no difference in mass conservation between the CG and DG methods, unless the resolution is very coarse (order of 200-300m in the density current case).

Lower order non-conforming CG simulations are less robust than the DG simulations and require more stabilization. Polynomial order 5 simulations require artificial viscosity of $\mu = 5.0\text{m}^2/\text{s}$ to complete 5000s simulation, while higher polynomial orders require $\mu = 2.0\text{m}^2/\text{s}$. We were also successful with running polynomial order 7 with $\mu = 0.5\text{m}^2/\text{s}$. None of the DG simulations required any additional stabilization, even for the very long simulation times used in this study. We showed that the stability problem in the CG non-conforming method can also be solved using filtering or a combination of filtering and artificial viscosity. We are currently investigating other stabilization methods as well. It is possible that adaptive local viscosity can stabilize the non-conforming simulations without applying artificial viscosity in the entire domain.

Another possible explanation of the stability issue is that certain regions of the domain are under-resolved and therefore the CG method is unstable. Our refinement criterion is designed to follow certain physical features and is not an error indicator, which would pick up any growing instabilities.

In the scope of this paper only the issue of mass conservation was discussed. In that comparison both CG and DG non-conforming AMR methods perform equally well. We will address the issues of efficiency and accuracy in a separate paper.

Acknowledgements

The authors gratefully acknowledge the support of the Office of Naval Research through program element PE-0602435N, the National Science Foundation (Division of Mathematical Sciences) through program element 121670, and the Air Force Office of Scientific Research through the Computational Mathematics program.

The authors are grateful to Paul Fischer for insightful discussions regarding the implementation of the CG non-conforming algorithm. Part of this research was conducted at the Isaac Newton Institute for Mathematical Sciences during the Multiscale Numerics for the Atmosphere and Ocean Program.

The authors gratefully acknowledge the assistance of Carlos Borges on some Linear Algebra issues in the proof in Appendix B.

[1] M. A. Kopera, F. X. Giraldo, Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations, J. Comput. Phys. 2014.
URL http://dx.doi.org/10.1016/j.jcp.2014.06.026

[2] Y. Maday, C. Mavriplis, A. T. Patera, Nonconforming mortar element methods: Application to spectral discretizations, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, 1988.

[3] C. Mavriplis, Nonconforming discretizations and a posteriori error estimators for adaptive spectral element techniques, Ph.D. thesis, Massachusetts Institute of Technology (1989).

[4] R. D. Henderson, G. E. Karniadakis, Unstructured spectral element methods for simulation of turbulent flows, J. Comput. Phys. 122 (2) (1995) 191–217.

[5] R. Henderson, Dynamic refinement algorithms for spectral element methods, Comput. Method. Appl. M. 175 (3) (1999) 395–411.

[6] P. F. Fischer, G. W. Kruse, F. Loth, Spectral element methods for transitional flows in complex geometries, J. Sci. Comput. 17 (1-4) (2002) 81–98.

[7] C. Sert, A. Beskok, Spectral element formulations on non-conforming grids: A comparative study of pointwise matching and integral projection methods, J. Comput. Phys. 211 (1) (2006) 300–325.

[8] D. Rosenberg, A. Fournier, P. Fischer, A. Pouquet, Geophysical–astrophysical spectral-element adaptive refinement (GASpAR): Object-oriented h-adaptive fluid dynamics simulation, J. Comput. Phys. 215 (1) (2006) 59–80.

[9] A. St-Cyr, C. Jablonowski, J. M. Dennis, H. M. Tufo, S. J. Thomas, A comparison of two shallow-water models with nonconforming adaptive grids., Mon. Weather Rev. 136 (6) (2008) 189801922.

[10] T. J. R. Hughes, G. Engel, L. Mazzei, M. G. Larson, The continuous Galerkin method is locally conservative, J. Comput. Phys. 163 (2) (2000) 467–488.

[11] T. J. R. Hughes, G. Engel, L. Mazzei, M. G. Larson, A comparison of discontinuous and continuous Galerkin methods based on error estimates, conservation, robustness and efficiency, in: Discontinuous Galerkin Methods, Springer, 2000, pp. 135–146.

[12] M. A. Taylor, J. Edwards, S. Thomas, R. Nair, A mass and energy conserving spectral element atmospheric dynamical core on the cubed-sphere grid, in: J. Phys. Conf. Ser., Vol. 78, IOP Publishing, 2007, p. 012074.

[13] M. A. Taylor, A. Fournier, A compatible and conservative spectral element method on unstructured grids, J. Comput. Phys. 229 (17) (2010) 5879–5895.

[14] J. F. Kelly, F. X. Giraldo, Continuous and discontinuous Galerkin methods for a scalable 3D nonhydrostatic atmospheric model: limited area mode, J. Comp. Phys. 231 (2) (2012) 7988–8008.

[15] F. X. Giraldo, J. F. Kelly, E. M. Constantinescu, Implicit-explicit formulations for a 3D Nonhydrostatic Unified Model of the Atmosphere (NUMA), SIAM J. Sci. Comp. 35 (5) (2013) B1162–B1194.

[16] F. X. Giraldo, M. Restelli, A study of spectral element and discontinuous Galerkin methods for the Navier-Stokes equations in non-hydrostatic mesoscale atmospheric modeling: equation sets and test cases, J. Comput. Phys. 227 (1) (2008) 3849–3877.

[17] M. Yu, F. X. Giraldo, J. Kim, Z. J. Wang, Localized artificial viscosity stabilization of discontinuous Galerkin methods for non-hydrostatic atmospheric modeling, J. Comput. Phys. in review.

[18] S. Marras, M. Nazarov, F. X. Giraldo, A conservative, consistent, and grid adaptive stabilized spectral element method based on a dynamic LES model for the compressible Euler equations, J. Comp. Phys.

[19] F. X. Giraldo, Continuous and discontinuous Galerkin methods for atmospheric modeling, in: Proceedings of the Recent Developments in Numerical Methods for Atmosphere and Ocean Modelling Seminar, ECMWF, 2013, pp. 167–183.
URL http://old.ecmwf.int/publications/library/ecpublications/_pdf/seminar/2013/Giraldo.pdf

[20] V. V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, USSR Comp. Math. and Math. Phys. 1 (1961) 267–279.

[21] F. X. Giraldo, T. E. Rosmond, A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP: Dynamical core tests., Mon. Weather Rev. 132 (1) (2004) 133–153.

[22] A. Fournier, D. Rosenberg, A. Pouquet, Dynamically adaptive spectral-element simulations of 2D incompressible Navier–Stokes vortex decays, Geophysical and Astrophysical Fluid Dynamics 103 (2-3) (2009) 245–268.

[23] N. J. Higham, The accuracy of floating point summation, SIAM J. Sci. Comput. 14 (4) (1993) 783–799.

[24] J. Straka, R. B. Wilhelmson, J. R. Anderson, K. K. Droegemeier, Numerical solutions of a non-linear density current: a benchmark solution and comparisons, Int. J. Numer. Meth. Fl. 17 (1993) 1–22.

[25] F. X. Giraldo, The Lagrange–Galerkin spectral element method on unstructured quadrilateral grids, Journal of Computational Physics 147 (1) (1998) 114–146.
[26] A. Mueller, J. Behrens, F. X. Giraldo, V. Wirth, An adaptive discontinuous Galerkin method for modeling atmospheric convection, J. Comput. Phys. 235 (1) (2012) 371–393.
[27] J. P. Boyd, The erfc-log filter and the asymptotics of the Euler and Vandeven sequence accelerations, in: A. V. Ilin, L. Scott (Eds.), Proceedings of the Third International Conference on Spectral and High Order Methods, Houston, Texas, 1996, pp. 267–276.

Appendix A

In this appendix we provide explicitly the **Q** arrays for both conforming (Fig. 1) and non-conforming (Fig. 2) examples.

Conforming $\boldsymbol{Q}$ matrix:

$$u^{DG} = \boldsymbol{Q} u^{CG} \quad \rightarrow \quad \begin{bmatrix} u_{(1,1)} \\ u_{(1,2)} \\ u_{(1,3)} \\ u_{(1,4)} \\ u_{(1,5)} \\ u_{(1,6)} \\ u_{(1,7)} \\ u_{(1,8)} \\ u_{(1,9)} \\ u_{(2,1)} \\ u_{(2,2)} \\ u_{(2,3)} \\ u_{(2,4)} \\ u_{(2,5)} \\ u_{(2,6)} \\ u_{(2,7)} \\ u_{(2,8)} \\ u_{(2,9)} \end{bmatrix} = \begin{bmatrix} 1 & & & & & & & & & & & & & & \\ & 1 & & & & & & & & & & & & & \\ & & 1 & & & & & & & & & & & & \\ & & & 1 & & & & & & & & & & & \\ & & & & 1 & & & & & & & & & & \\ & & & & & 1 & & & & & & & & & \\ & & & & & & 1 & & & & & & & & \\ & & & & & & & 1 & & & & & & & \\ & & & & & & & & 1 & & & & & & \\ & 1 & & & & & & & & & & & & & \\ & & & & & & & & & 1 & & & & & \\ & & & & & & & & & & 1 & & & & \\ & & & & & 1 & & & & & & & & & \\ & & & & & & & & & & & 1 & & & \\ & & & & & & & & & & & & 1 & & \\ & & & & & & & & 1 & & & & & & \\ & & & & & & & & & & & & & 1 & \\ & & & & & & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \end{bmatrix}$$

Non-conforming $\boldsymbol{Q}$ matrix:

$$u^{DG} = \boldsymbol{Q} u^{CG} \quad \rightarrow \quad \begin{bmatrix} u_{(1,1)} \\ u_{(1,2)} \\ u_{(1,3)} \\ u_{(1,4)} \\ u_{(2,1)} \\ u_{(2,2)} \\ u_{(2,3)} \\ u_{(2,4)} \\ u_{(3,1)} \\ u_{(3,2)} \\ u_{(3,3)} \\ u_{(3,4)} \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ 1 & & & & & & \\ & & & & 1 & & \\ \frac{1}{2} & & \frac{1}{2} & & & & \\ & & & & & 1 & \\ \frac{1}{2} & & \frac{1}{2} & & & & \\ & & & & & 1 & \\ & & & 1 & & & \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}$$

Note that the non-Boolean entires come from the interpolation between parent and children elements.

5. Appendix B - Proof of Mass Conservation by the DSS

Let us introduce the following theorem

Theorem 1. *Global mass is conserved by the DSS operator in the CG discretization regardless of whether the mesh is conforming or non-conforming.*

The proof of this theorem is accomplished across two subsections. In Sec. 5.1 we define some essential properties (Lemmas) that are used in the proof described in Sec. 5.2.

5.1. Definitions and Properties of Matrices

Let us define $\boldsymbol{Q} \in R^{m \times n}$ as the *scatter* matrix that scatters the CG ($\in C^0$) solution of dimension n to the DG solution of dimension m. Let $\boldsymbol{Q}^T \in R^{n \times m}$ define the *gather* matrix that gathers the DG ($\notin C^0$) of dimension m, to the CG solution ($\in C^0$) of dimension n. The dimension m is defined as $m = N_e(N+1)^d$ where N_e are the number of elements in the grid, N is the polynomial order in the CG method, and d is the spatial dimension. The dimension n is such that $n < m$ and defines the distinct global gridpoints. Examples of simple Q matrices are given in Appendix A.

Let us define a diagonal matrix with integration weights and Jacobians as $\boldsymbol{W} \in R^{m \times m}$, which is built from the local element-wise mass matrices in a DG method. The corresponding CG mass matrix is defined as follows:

$$\boldsymbol{M} = \boldsymbol{Q}^T \boldsymbol{W} \boldsymbol{Q} \in R^{n \times n}. \tag{10}$$

Let us now review some properties of the matrices $\boldsymbol{Q}$ and $\boldsymbol{W}$. The rows i and columns j of the matrix $\boldsymbol{Q}$ are in fact defined as follows: $Q_{ij} = L_j(\boldsymbol{\xi}_i)$ where $i = 1, ..., m$, $j = 1, ..., n$, $L(\boldsymbol{\xi})$ are the Lagrange polynomials that we use for interpolation, and $\boldsymbol{\xi}$ are the interpolation points in multiple dimensions. To make the following exposition easier to follow, (although not necessary for the proof), one can assume that the multi-dimensional Lagrange polynomials are derived from the tensor-product bases in one-dimension. This further simplifies the exposition by allowing us to assume that we are only interested in a single non-conforming edge (a one-dimensional construct). If we define the one-dimensional Lagrange polynomials as $h(\xi)$ we can then rewrite $\boldsymbol{Q}_{ij} = h_j(\xi_i)$.

Lagrange polynomials enjoy the property of the partition of unity, i.e., $\forall \xi_i \in [-1, +1] \sum_{j=1}^{n} h_j(\xi_i) = 1$. Therefore, all of the rows ($i = 1, ..., m$) of $\boldsymbol{Q}$ sum to 1. This means that $\boldsymbol{Q}^T_{ji} = h_j(\xi_i)$ has the property that all the columns ($i = 1, ..., m$) sum to 1. Let us call this the *Summation Property of* $\boldsymbol{Q}^T$ (see also [22]).

Next, we need to show that the matrices $\boldsymbol{Q}^T\boldsymbol{W}$ and $\boldsymbol{M} = \boldsymbol{Q}^T\boldsymbol{W}\boldsymbol{Q}$ are of full rank. The matrix $\boldsymbol{W} \in R^{m \times m}$ is of full rank m since the diagonal elements are the weights and Jacobians of the integration - the LGL weights are always positive and the Jacobians are positive for well-defined grids (since they represent the volume of the element). For the matrix $\boldsymbol{Q}^T \in R^{n \times m}$ we note that it has n linearly-independent rows by construction (since it defines the n distinct global gridpoints). Therefore, $\boldsymbol{Q} \in R^{m \times n}$ has n linearly-independent columns and, therefore, the products $\boldsymbol{Q}^T\boldsymbol{W} \in R^{n \times m}$ and $\boldsymbol{M} = \boldsymbol{Q}^T\boldsymbol{W}\boldsymbol{Q} \in R^{n \times n}$ will both have full rank n. Since $\boldsymbol{M}$ is square and of full rank, then it is invertible.

5.2. Proof of Mass Conservation

To prove that the mass conservation is unchanged by the Direct-Stiffness-Summation (DSS) operator, we begin by assuming that we have a solution vector (say, for density) defined as q^{DG} which is strictly a local element-wise vector such that $q^{DG} \in R^m$ and $q^{DG} \notin C^0$.

Step 1 We apply local element-wise integration which is represented as the matrix-vector operation

$$\boldsymbol{W} q^{DG} \in R^{m \times 1}.$$

Step 2 We gather the solution to all the n distinct global gridpoints via the matrix multiplication

$$\boldsymbol{Q}^T \boldsymbol{W} q^{DG} \in R^{n \times 1}.$$

This is the global assembly operation typical of continuous finite element methods.

Step 3 We multiply by the inverse global mass matrix to bring the integral solution of Step 2 to a global gridpoint solution as follows

$$\boldsymbol{M}^{-1}\boldsymbol{Q}^T\boldsymbol{W}q^{DG} \in R^{n\times 1}.$$

The resulting vector lives in the n distinct global gridpoints and is in C^0 (CG space).

Step 4 Finally, we scatter the solution to the m non-distinct local gridpoints as follows

$$q^{DG}_{C^0} = \boldsymbol{Q}\boldsymbol{M}^{-1}\boldsymbol{Q}^T\boldsymbol{W}q^{DG} \in R^{m\times 1}. \tag{11}$$

The resulting vector on the left-hand-side lives in the DG space (of dimension m) but is in C^0. The original vector q^{DG} need not live in C^0.

Step 5 Let us now replace the mass matrix in Eq. (11) using Eq. (10) which results in

$$q^{DG}_{C^0} = \boldsymbol{Q}\left(\boldsymbol{Q}^T\boldsymbol{W}\boldsymbol{Q}\right)^{-1}\boldsymbol{Q}^T\boldsymbol{W}q^{DG}. \tag{12}$$

Equation (12) is defined in [13] as the projection and self-adjoint matrix (see also [22]).

Step 6 Since $\boldsymbol{Q}^T\boldsymbol{W}$ is of full rank, then we can left-multiply Eq. (12) by it to obtain

$$\boldsymbol{Q}^T\boldsymbol{W}q^{DG}_{C^0} = \left(\boldsymbol{Q}^T\boldsymbol{W}\boldsymbol{Q}\right)\left(\boldsymbol{Q}^T\boldsymbol{W}\boldsymbol{Q}\right)^{-1}\boldsymbol{Q}^T\boldsymbol{W}q^{DG}. \tag{13}$$

which simplifies to

$$\boldsymbol{Q}^T\boldsymbol{W}q^{DG}_{C^0} = \boldsymbol{Q}^T\boldsymbol{W}q^{DG}. \tag{14}$$

Step 7 Introducing the vector $\boldsymbol{e}_n = (1, ..., 1)^T \in R^{n\times 1}$ and taking the scalar product of this vector with Eq. (14) yields

$$\boldsymbol{e}_n^T\boldsymbol{Q}^T\boldsymbol{W}q^{DG}_{C^0} = \boldsymbol{e}_n^T\boldsymbol{Q}^T\boldsymbol{W}q^{DG} \tag{15}$$

where $\boldsymbol{e}_n^T\boldsymbol{Q}^T = \boldsymbol{e}_m^T$, by virtue of the *Summation Property of* $\boldsymbol{Q}^T$. This simplifies Eq. (15) to

$$\boldsymbol{e}_m^T\boldsymbol{W}q^{DG}_{C^0} = \boldsymbol{e}_m^T\boldsymbol{W}q^{DG} \tag{16}$$

which states that the global sum, weighted by $\boldsymbol{W}$ is identical for both the DG solution and the DG solution after it has been made to be C^0. The role of $\boldsymbol{W}$ is to perform the local integration of q^{DG}, while the scalar product with $\boldsymbol{e}$ is the global sum. Therefore, the global mass is conserved.