MPI Summer School 2024
Lecture 3: Multi-D DG Methods

Francis X. Giraldo

Department of Applied Mathematics
Naval Postgraduate School
Monterey CA. 93943-5216

September 2-6, 2024

Table of contents

Summary of DG Lectures 1 and 2

- We covered the background (theoretical) for DG methods.
- We saw that DG uses an integral (weak) form.
- We saw that DG requires good interpolation and good integration.
- One can construct a modal DG approach (using orthogonal polynomials) or a nodal DG approach (using Lagrange polynomials).
- We covered the construction of the 1D elemental matrices (Mass and Differentiation)
- This lecture can be downloaded here:

`https://www.mis.mpg.de/events/series/summer-school-on-mathematics-of-geophysical-flows`

Outline

1D Resulting Element Equations

- The resulting equation which must be satisfied within each DG element is

$$\frac{\Delta x}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\frac{d}{dt}\begin{pmatrix} q_0 \\ q_1 \end{pmatrix}^{(e)} - \frac{1}{2}\begin{pmatrix} -1 & -1 \\ +1 & +1 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \end{pmatrix}^{(e)} + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \end{pmatrix}^{(*)} = 0$$

  where the subscript 0 and 1 denote the current element's grid points.

- This, in fact, is nothing more than the matrix problem

$$M_{ij}\frac{dq_j^{(e)}}{dt} - \widetilde{D}_{ij}f_j^{(e)} + F_{ij}f_j^{(*)} = 0$$

  where $f^{(*)}$ denotes the numerical flux function.

Outline

- 1D Resulting Element Equations
- Numerical Flux
  - Rusanov Flux
- Volume Flux
- Results for the 1D Equations

Rusanov Flux

- The single most common numerical flux function is the Rusanov (or local Lax-Friedrichs) flux which is a generalized upwinding method.
- The Rusanov flux is defined as

$$f^{(*,k)} = \frac{1}{2}\left[f^{(e)} + f^{(k)} - \widehat{\boldsymbol{n}}_{\Gamma_e}^{(e,k)}|\lambda_{max}|\left(q^{(k)} - q^{(e)}\right)\right]$$

  where $\widehat{\boldsymbol{n}}_{\Gamma_e}^{(e,k)}$ denotes the outward pointing normal to the interface of the element e and its neighbor k, and λ_{max} is the maximum wave speed of your system. In our example, $\lambda_{max} = u$ but in general it represents the maximum eigenvalue of the Jacobian matrix of the governing equations of motion.
- The Rusanov flux is just the average value between the two elements sharing an edge with the addition of a dissipation term ($|\lambda|$).

Rusanov Flux

- This dissipation term will allow the flux function to modify itself based on the flow conditions in order to construct an upwind-biased method.
- Let's consider the following figure to see what the Rusanov flux would look like for the center element.

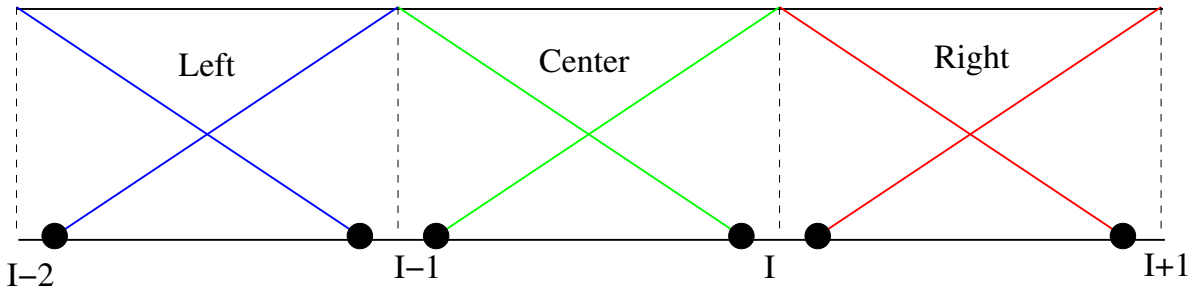


Figure: Contribution to the gridpoint I from the left $(I-2, I-1)$, center $(I-1, I)$ and right $(I, I+1)$ elements.

- Since $f = qu$ we can rewrite the Rusanov flux as

$$f^{(*,k)} = \frac{1}{2}\left[f^{(e)} + f^{(k)} - \widehat{\boldsymbol{n}}_{\Gamma_e}^{(e,k)}\left(f^{(k)} - f^{(e)}\right)\right]$$

Rusanov Flux

- At the interface $(e, k) = (C, L)$, that is, between the center and left elements, the outward pointing normal vector from C to L is $\widehat{\boldsymbol{n}}_{\Gamma_e}^{(e,k)} = -1$.
- This gives

$$f^{(*,C,L)} = \frac{1}{2}\left[f^{(C)} + f^{(L)} - (-1)\left(f^{(L)} - f^{(C)}\right)\right]$$

that can be simplified to

$$f^{(*,C,L)} = f^{(L)}.$$

- At the interface $(e, k) = (C, R)$, that is, between the center and right elements, the outward pointing normal vector from C to R is $\widehat{\boldsymbol{n}}_{\Gamma_e}^{(e,k)} = +1$.

Rusanov Flux

- This gives

$$f^{(*,C,R)} = \frac{1}{2}\left[f^{(C)} + f^{(R)} - (+1)\left(f^{(R)} - f^{(C)}\right)\right]$$

that can be simplified to

$$f^{(*,C,R)} = f^{(C)}.$$

- Using these flux values in our canonical equation gives for the center element equation

$$\frac{\Delta x}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\frac{d}{dt}\begin{pmatrix} q_{I-1}^{(C)} \\ q_I^{(C)} \end{pmatrix} - \frac{1}{2}\begin{pmatrix} -1 & -1 \\ +1 & +1 \end{pmatrix}\begin{pmatrix} f_{I-1}^{(C)} \\ f_I^{(C)} \end{pmatrix} + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} f_{I-1}^{(L)} \\ f_I^{(C)} \end{pmatrix} = 0$$

Outline

- 1D Resulting Element Equations
- Numerical Flux
- Volume Flux
  - Split Form
  - Two-Point Flux
- Results for the 1D Equations
- Multi-D Interpolation
- Multi-D Metric Terms
- Integral Form of the Multi-D Equations

Split Form

- To stabilize, e.g., the Euler equations some (Kennedy-Gruber JCP 2008, Kopriva et al. SISC 2014, Gassner et al. JCP 2016, Coppola et al. JCP 2019) proposed to use split forms as follows

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \quad \rightarrow \frac{\partial \rho}{\partial t} + \frac{1}{2} \nabla \cdot (\rho \boldsymbol{u}) + \frac{1}{2} \left[\boldsymbol{u} \cdot \nabla \rho + \rho \nabla \cdot \boldsymbol{u} \right] = 0$$

  which has been shown to preserve kinetic energy (so called Kinetic-Energy-Preserving, KEP schemes).
- This idea can be carried out for the momentum and thermodynamic equations.

Two-Point Flux

- The KEP scheme can be replicated by simply modifying the volume flux in a clever way.
- To show how this is done, let us first write the continuity equation in a simplified discrete form as follows

$$M_{ij}\frac{d\rho_j}{dt} + \boldsymbol{D}_{ij}^T \boldsymbol{f}^{(\#)} = 0$$

where $\boldsymbol{f}^{(\#)}$ is a new flux function.

- Let us use $\boldsymbol{f}^{(\#)} = 2\{\{\rho\}\}\{\{\boldsymbol{u}\}\}$ where $\{\{\rho\}\} = \frac{1}{2}(\rho_i + \rho_j)$.
- Subbing into the discrete form yields

$$M_{ij}\frac{d\rho_j}{dt} + \frac{1}{2}\boldsymbol{D}_{ij}^T(\rho_i\boldsymbol{u}_i + \rho_j\boldsymbol{u}_j + \boldsymbol{u}_i\rho_j + \rho_i\boldsymbol{u}_j) = 0.$$

Two-Point Flux

- Using the zero row-sum property of $\boldsymbol{D}$, i.e., $\sum_{j=0}^{N} D_{ij} = 0$ allows us to simplify the discrete form as follows

$$M_{ij}\frac{d\rho_j}{dt} + \frac{1}{2}\boldsymbol{D}_{ij}^{T}\left(\rho_j \boldsymbol{u}_j\right) + \frac{1}{2}\left[\boldsymbol{u}_i \boldsymbol{D}_{ij}\rho_j + \rho_i \boldsymbol{D}_{ij}^{T}\boldsymbol{u}_j\right] = 0$$

- This is recovers to the KEP scheme defined using the split form.
- This idea (generally called *Flux Differencing*) is very powerful because we can replicate many split forms found in the literature by just redefining the volume flux. For DG we also have to be mindful of how we define the boundary flux (what we called *Numerical Flux* above).
- Flux Differencing has been used to construct *Entropy-Conservative* and *Entropy-Stable* DG discretizations for numerous types of hyperbolic PDEs.

Outline

Results for 1D Wave Equation

- Suppose we wish to solve the continuous partial differential equation

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0 \qquad \forall x \in [-1, +1]$$

  where $f = qu$ and $u = 2$ is a constant.
- Thus, an initial wave $q(x, 0)$ will take exactly $t = 1$ time in order to complete one full revolution (loop) of the domain.
- Since the governing PDE is a hyperbolic system, then this problem represents an initial value problem (IVP or Cauchy Problem).
- We, therefore, need an initial condition.
- Let it be the following Gaussian

$$q(x, 0) = e^{-cx^2}$$

Boundary Conditions and Norms

- This problem also requires a boundary condition: let us impose periodic boundary conditions, meaning that the domain at $x = +1$ should wrap around and back to $x = -1$.
- Let us define the normalized L^2 error norm as follows

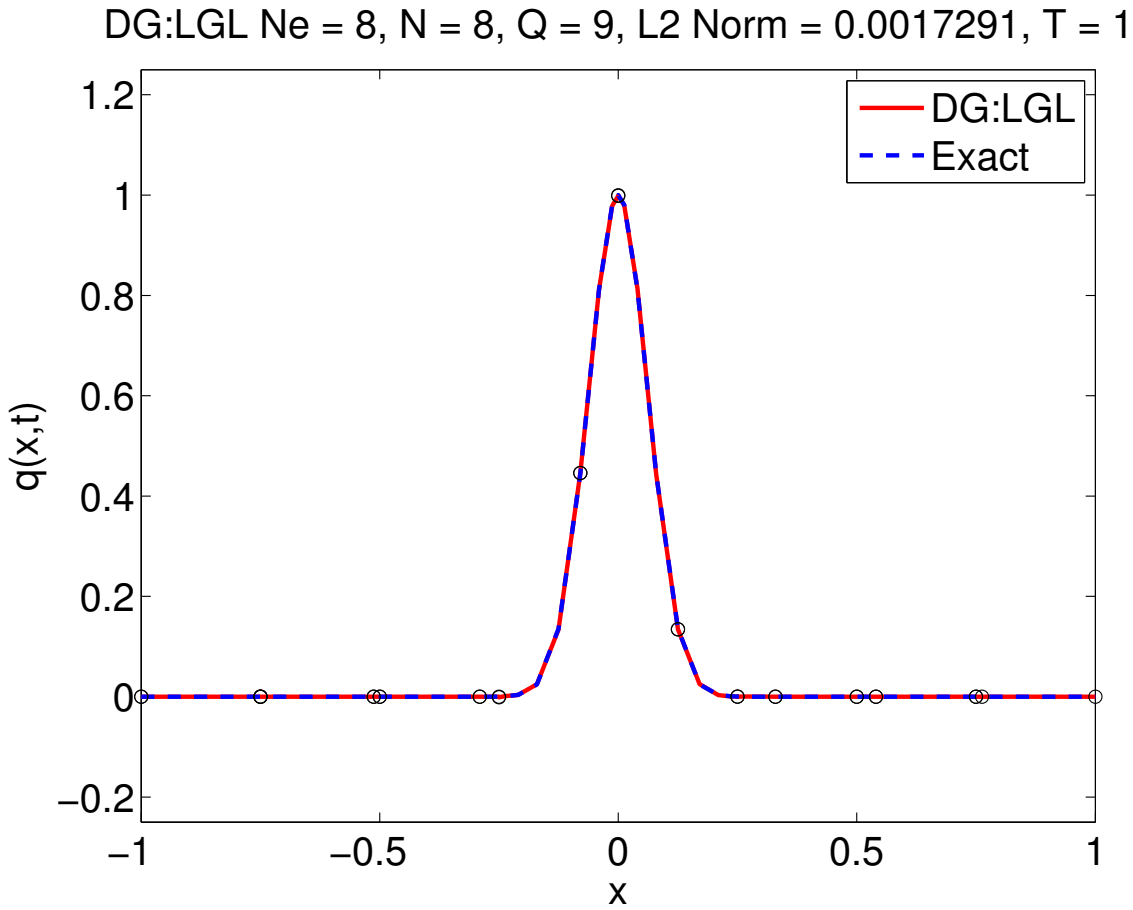
$$L^2 = \sqrt{\frac{\sum_{k=1}^{N_p} \left(q_k^{numerical} - q^{exact}(x_k)\right)^2}{\sum_{k=1}^{N_p} q^{exact}(x_k)^2}}$$

where $k = 1, ..., N_p$ are $N_p = N_e(N+1)$ global gridpoints and $q^{numerical}$ and q^{exact} are the numerical and exact solutions after one full revolution of the wave.

- Note that the wave should just stop where it began without changing shape (in a perfect world).

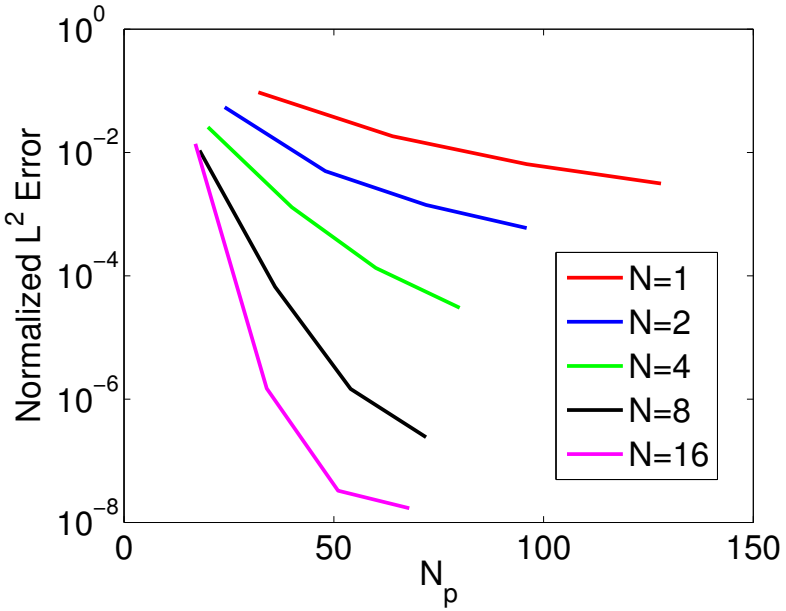
Solution Accuracy

- The figure below shows the snapshot of the exact and DG numerical solutions (with Rusanov flux) after one revolution ($t = 1$) using $N = 8$ order polynomials and $N_e = 8$ elements for a total of $N_p = 72$ gridpoints.



Convergence Rates: Lobatto Points

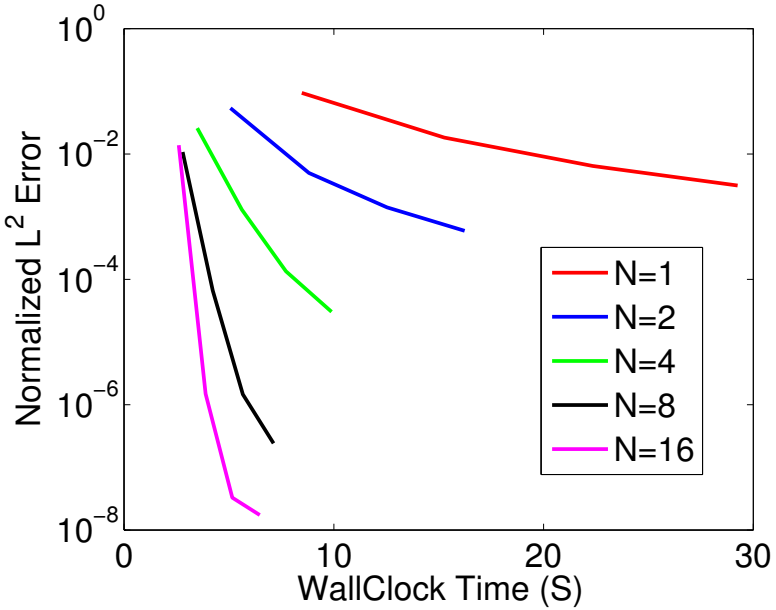
- The figure below shows the convergence rates of various polynomial orders, N, for a total number of gridpoints N_p where, for 1D, $N_p = N_e(N+1)$.



- Is high-order worthwhile?

Convergence Rates

- The figure below shows the L^2 error norm as a function of wallclock time in seconds (work-precision diagram).



- The high-order methods are, in fact, more efficient to reach a certain level of accuracy than the low-order methods.
- To achieve an accuracy of 10^{-4} or 10^{-8} is most efficiently reached with $N = 16$; $N = 1$ and $N = 2$ would require prohibitively large computational times to achieve these levels of accuracy.

1D Euler Equations

- Let us consider the 1D Euler equations written in conservation (flux) form

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) = 0$$
$$\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}(\rho u u + P) = 0$$
$$\frac{\partial \rho e}{\partial t} + \frac{\partial}{\partial x}\left[(\rho e + P)\, u\right] = 0$$

  where $e = c_v T + \frac{1}{2}u^2$ is the total energy and $P = \rho R T$ is the pressure.
- Let us define the initial condition

$$(\rho, u, P) = \begin{cases} (1, 0, 1) & x < 0, \\ (0.125, 0, 0.1) & x \geq 0 \end{cases}$$

  for the domain $x \in [-\frac{1}{2}, +\frac{1}{2}]$, and Dirichlet boundary conditions on both endpoints defined by the values from the initial conditions, and run for time $t \in [0, 0.2]$.

1D Euler Equations

- Since these initial conditions form a shock, we need to stabilize the DG method.
- Let us discretize the Euler equations as follows

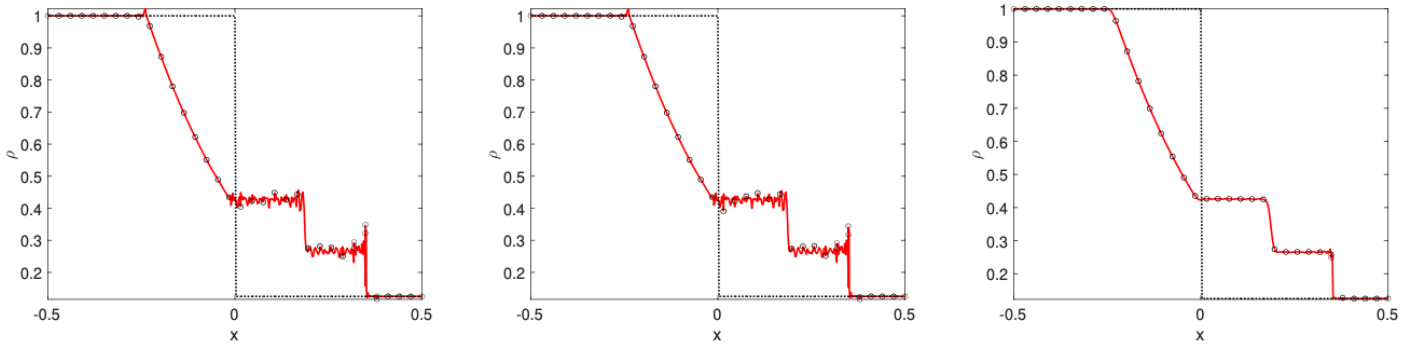
$$M\frac{d\boldsymbol{q}}{dt} + D\boldsymbol{f}^{(\#)} = 0$$

- Let us use the Kinetic-Energy-Preserving (KEP) volume flux defined as follows

$$\boldsymbol{f}^{(\#)} \equiv \boldsymbol{f}^{(KEP)} = \begin{pmatrix} \{\{\rho\}\}\{\{u\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{u\}\} + \{\{P\}\} \\ (\{\{\rho\}\}\{\{e\}\} + \{\{P\}\})\{\{u\}\} \end{pmatrix}$$

1D Euler Equations

- Sod Shock-tube Problem ($N = 16$ order polynomials)



(a) ES (b) KEP (c) Artificial Diffusion

Fig. 18.33: Density plot for the Sod shock tube problem at time=0.2 using 33 elements and 16th order polynomials for the following methods: (a) entropy-stable, (b) kinetic-energy-preserving, and (c) standard DG with $\mu = 1 \times 10^{-4}$.

- The ES flux is given in, e.g., Ismail-Roe JCP 2009, Renac JCP 2019, Waruszewski et al. JCP 2022. Omitted for brevity.

Outline

- 1D Resulting Element Equations
- Numerical Flux
- Volume Flux
- Results for the 1D Equations
- Multi-D Interpolation
- Multi-D Metric Terms
- Integral Form of the Multi-D Equations

Multi-D Interpolation

- Recall that a 1D nodal interpolation of a function $f(x)$ is given as

$$f(x) = \sum_{i=0}^{N} L_i(x) f_i$$

where L are Lagrange polynomials and $f_i = f(x_i)$ is the value of the function $f(x)$ evaluated at $x = x_i$.

- The multi-D nodal interpolation of a function $f(\boldsymbol{x})$ is given as

$$f(\boldsymbol{x}) = \sum_{i=0}^{N} L_i(\boldsymbol{x}) f_i$$

- In 3D, the Lagrange polynomials are constructed as follows

$$L_{ijk}^{3D}(x, y, z) = L_i(x) L_j(y) L_k(z), \ \forall \ (i, j, k) = 0, ..., N.$$

- The popularity of tensor-product elements is very much a direct consequence of the ease with which interpolation functions can be constructed.

Multi-D Interpolation

- Another reason for their ubiquity is due to their computational efficiency.
- Methods for multi-dimensions that are based on tensor products tend to be very fast because a very nice trick (called sum factorization) can be used to reduce the cost of the method to $\mathcal{O}(N^{d+1})$ where N is the order of the interpolation functions and d is the dimension of the space.
- A naive implementation of a derivative computation would cost $\mathcal{O}(N^{2d})$.

Multi-D Interpolation

- The tensor-product form of the basis functions is written

$$\psi_{ijk}(\xi,\eta,\zeta) = L_i(\xi)L_j(\eta)L_k(\zeta),\ \forall\ i = 0,...,N_\xi,\ j = 0,...,N_\eta,\ k = 0,...,N_\zeta$$

- The monolithic form is

$$\psi_I(\xi,\eta,\zeta) = L_i(\xi)L_j(\eta)L_k(\zeta),$$

$$\forall\ i = 0,...,N_\xi,\ j = 0,...,N_\eta,\ k = 0,...,N_\zeta,$$

with

$$I = i + 1 + jN_\eta + kN_\eta N_\zeta,\ M_N = (N_\xi + 1)(N_\eta + 1)(N_\zeta + 1)$$

- In practice, we use the tensor-product form but the monolithic form is great for describing the approach in general dimensions.

Multi-D Interpolation

- With the basis functions defined, we can now expand the solution variable q as follows

$$q_N^{(e)}(\boldsymbol{x}, t) = \sum_{j=1}^{M_N} \psi_j(\boldsymbol{x}) q_j^{(e)}(t)$$

- which implies the approximation of the gradient operator to be

$$\boldsymbol{\nabla} q_N^{(e)}(\boldsymbol{x}, t) = \sum_{j=1}^{M_N} \boldsymbol{\nabla} \psi_j(\boldsymbol{x}) q_j^{(e)}(t)$$

- where the partial derivatives are defined as follows

$$\frac{\partial q_N^{(e)}(\boldsymbol{x}, t)}{\partial x_i} = \sum_{j=1}^{M_N} \frac{\partial \psi_j(\boldsymbol{x})}{\partial x_i} q_j^{(e)}(t), \ i = 1, ..., d.$$

- Since we will perform all of our computations in the reference element with coordinates (ξ, η, ζ) then we must transform the derivatives from (x, y, z) to (ξ, η, ζ).

Outline

- 1D Resulting Element Equations
- Numerical Flux
- Volume Flux
- Results for the 1D Equations
- Multi-D Interpolation
- Multi-D Metric Terms
- Integral Form of the Multi-D Equations

Multi-D Metric Terms

- Using the chain rule, we write the derivatives of the basis functions as

$$\frac{\partial \psi(\boldsymbol{x}(\boldsymbol{\xi}))}{\partial x_i} = \frac{\partial \psi(\boldsymbol{\xi})}{\partial \xi}\frac{\partial \xi(\boldsymbol{x})}{\partial x_i} + \frac{\partial \psi(\boldsymbol{\xi})}{\partial \eta}\frac{\partial \eta(\boldsymbol{x})}{\partial x_i} + \frac{\partial \psi(\boldsymbol{\xi})}{\partial \zeta}\frac{\partial \zeta(\boldsymbol{x})}{\partial x_i}$$

- Which we write as

$$\frac{\partial \psi}{\partial \boldsymbol{x}} = \frac{\partial \psi}{\partial \xi}\frac{\partial \xi}{\partial \boldsymbol{x}} + \frac{\partial \psi}{\partial \eta}\frac{\partial \eta}{\partial \boldsymbol{x}} + \frac{\partial \psi}{\partial \zeta}\frac{\partial \zeta}{\partial \boldsymbol{x}}$$

- For $i = 1, ..., d$ where $x = x_1, y = x_2, z = x_3$.
- We can compute these derivatives after we build $\frac{\partial \psi}{\partial \xi_i}$ and $\frac{\partial \xi_i}{\partial x_j}$.

Multi-D Metric Terms

- To construct the metric terms in 3D, we start with $\boldsymbol{x} = \boldsymbol{x}(\xi, \eta, \zeta)$ and apply the chain rule as follows

$$d\boldsymbol{x} = \boldsymbol{x}_\xi d\xi + \boldsymbol{x}_\eta d\eta + \boldsymbol{x}_\zeta d\zeta.$$

- This now yields the following matrix form

$$\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{pmatrix} \begin{pmatrix} d\xi \\ d\eta \\ d\zeta \end{pmatrix}$$

- With the Jacobian defined as

$$\boldsymbol{J} = \begin{pmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{pmatrix} \tag{2}$$

Multi-D Metric Terms

- With determinant defined as such

$$\mathcal{J} \equiv \boldsymbol{x}_\xi \cdot (\boldsymbol{x}_\eta \times \boldsymbol{x}_\zeta)$$

- With the contravariant metric terms

$$\nabla\xi = \frac{1}{\mathcal{J}}\left(\frac{\partial \boldsymbol{x}}{\partial \eta} \times \frac{\partial \boldsymbol{x}}{\partial \zeta}\right)$$

$$\nabla\eta = \frac{1}{\mathcal{J}}\left(\frac{\partial \boldsymbol{x}}{\partial \zeta} \times \frac{\partial \boldsymbol{x}}{\partial \xi}\right)$$

$$\nabla\zeta = \frac{1}{\mathcal{J}}\left(\frac{\partial \boldsymbol{x}}{\partial \xi} \times \frac{\partial \boldsymbol{x}}{\partial \eta}\right)$$

- Which, we refer to as the *cross-product* form of the metric terms.
- The curl-invariant form are defined as follows

$$\nabla\xi^i = \frac{1}{2\mathcal{J}}\left[\left(\boldsymbol{x}_{\xi^j} \times \boldsymbol{x}\right)_{\xi^k} - \left(\boldsymbol{x}_{\xi^k} \times \boldsymbol{x}\right)_{\xi^j}\right] \tag{4}$$

Multi-D Metric Terms

- The final step required to compute these metric terms is to approximate the derivatives $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}$ using the basis functions.
- Let us approximate the physical coordinates by the basis function expansion

$$\boldsymbol{x}_N(\xi, \eta, \zeta) = \sum_{j=0}^{N} \psi_j(\xi, \eta, \zeta) \boldsymbol{x}_j$$

which then yields the derivatives

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\xi, \eta, \zeta) = \sum_{j=0}^{N} \frac{\partial \psi_j}{\partial \boldsymbol{\xi}}(\xi, \eta, \zeta) \boldsymbol{x}_j.$$

Outline

- 1D Resulting Element Equations
- Numerical Flux
- Volume Flux
- Results for the 1D Equations
- Multi-D Interpolation
- Multi-D Metric Terms
- Integral Form of the Multi-D Equations

Integral Form of the Multi-D Equations

- We begin with the PDE in continuous form with suitable initial and boundary conditions, conservation law form

$$\frac{\partial q}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f} = 0.$$

- To construct the weak integral form, we begin by expanding the solution variable q using the monolithic form as follows

$$q_N^{(e)}(\boldsymbol{x}, t) = \sum_{j=1}^{M_N} \psi_j(\boldsymbol{x}) q_j^{(e)}(t)$$

with

$$\boldsymbol{u}_N^{(e)}(\boldsymbol{x}) = \sum_{j=1}^{M_N} \psi_j(\boldsymbol{x}) \boldsymbol{u}_j^{(e)}$$

and $\boldsymbol{f}_N^{(e)} = q_N^{(e)} \boldsymbol{u}_N^{(e)}$ where $\boldsymbol{u}(\boldsymbol{x}) = u(\boldsymbol{x})\widehat{\boldsymbol{i}} + v(\boldsymbol{x})\widehat{\boldsymbol{j}} + w(\boldsymbol{x})\widehat{\boldsymbol{k}}$ is the velocity vector with (u, v, w) the components along $(\widehat{\boldsymbol{i}}, \widehat{\boldsymbol{j}}, \widehat{\boldsymbol{k}})$.

Integral Form of the Multi-D Equations

- Recall that the integer M_N in the basis function approximation

$$q_N^{(e)}(\boldsymbol{x}, t) = \sum_{j=1}^{M_N} \psi_j(\boldsymbol{x}) q_j^{(e)}(t)$$

  is a function of N which determines the number of points inside each element Ω_e.
- For tensor-product bases $M_N = (N+1)^d$ where d is the spatial dimension.
- Which we can generalize to $M_N = (N_\xi + 1)(N_\eta + 1)(N_\zeta + 1)$.

Integral Form of the Multi-D Equations

- Next, we substitute $q_N^{(e)}$, $\boldsymbol{u}_N^{(e)}$, and $\boldsymbol{f}_N^{(e)}$ into the PDE, multiply by a test function and integrate within the local domain Ω_e yielding the weak integral form: find $q \in L^2$ such that

$$\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} \, d\Omega_e + \int_{\Gamma_e} \psi_i \widehat{\boldsymbol{n}}^{(e,k)} . \boldsymbol{f}_N^{(*,k)} \, d\Gamma_e - \int_{\Omega_e} \boldsymbol{\nabla} \psi_i \cdot \boldsymbol{f}_N^{(e)} \, d\Omega_e = 0$$

$$\forall \psi \in L^2$$

- where $= \boldsymbol{f}^{(*,k)}$ is the numerical flux function which we will assume to be the Rusanov flux

$$\boldsymbol{f}^{(*,k)} = \frac{1}{2} \left[\boldsymbol{f}^{(k)} + \boldsymbol{f}^{(e)} - \mid \lambda \mid \widehat{\boldsymbol{n}}^{(e,k)} \left(q^{(k)} - q^{(e)} \right) \right]$$

Integral Form of the Multi-D Equations

- In standard DG notation, the Rusanov flux is written as

$$\boldsymbol{f}^{(*,k)} = \left\{\left\{ \boldsymbol{f}^{(e,k)} \right\}\right\} - \mid \lambda \mid \widehat{\boldsymbol{n}}^{(e,k)} \left[\left[q^{(e,k)} \right]\right]$$

- where the superscript (k) denotes the face neighbor of (e), $\widehat{\boldsymbol{n}}^{(e,k)}$ is the unit normal vector of the face shared by elements e and k, and λ is the maximum wave speed of the system.
- For the wave equation λ is just the maximum normal velocity $\widehat{\boldsymbol{n}} \cdot \boldsymbol{u}|_{\Gamma_e}$.

Integral Form of the Multi-D Equations

- The equation that we need to solve is the element equation

$$\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} \, d\Omega_e + \int_{\Gamma_e} \psi_i \widehat{\boldsymbol{n}} \cdot \boldsymbol{f}_N^{(*)} \, d\Gamma_e - \int_{\Omega_e} \boldsymbol{\nabla} \psi_i \cdot \boldsymbol{f}_N^{(e)} \, d\Omega_e = 0$$

$$\forall \psi \in L^2$$

- At this point we have to decide whether or not we are going to evaluate the integrals using co-located quadrature (which results in inexact integration) or non-colocated quadrature (that, for a specific number of quadrature points, results in exact integration).
- Let us first see what happens when we use exact integration.

Integral Form of the Multi-D Equations

- Let $M_Q = (Q+1)^d$ where $Q = \frac{3}{2}N + \frac{1}{2}$ will integrate $3N$ polynomials exactly (note that the advection term is a 3N degree polynomial).
- This quadrature rule yields the following matrix-vector problem

$$M_{ij}^{(e)} \frac{dq_j^{(e)}}{dt} + \sum_{k=1}^{N_{face}} \left(\boldsymbol{F}_{ij}^{(e,k)} \right)^{\mathcal{T}} \boldsymbol{f}_j^{(*,k)} - \left(\widetilde{\boldsymbol{D}}_{ij}^{(e)} \right)^{\mathcal{T}} \boldsymbol{f}_j^{(e)} = 0$$

- where the above matrices are defined as follows:

$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j \, d\Omega_e = \sum_{k=1}^{M_Q} w_k \mathcal{J}_k \psi_{ik} \psi_{jk}$$

is the mass matrix where w_k and $\mathcal{J}_k$ are the quadrature weight and determinant of the Jacobian evaluated at the quadrature point ξ_k.

Integral Form of the Multi-D Equations

- The flux matrix is defined as follows

$$\boldsymbol{F}_{ij}^{(e,k)} = \int_{\Gamma_e} \psi_i \psi_j \widehat{\boldsymbol{n}}^{(e,k)} \, d\Gamma_e = \sum_{l=1}^{M_Q^{(F)}} w_l^{(k)} \mathcal{J}_l^{(k)} \psi_{il} \psi_{jl} \widehat{\boldsymbol{n}}_l^{(e,k)}$$

  where the superscript (k) denotes the face variables,

- and

$$\widetilde{\boldsymbol{D}}_{ij}^{(e)} = \int_{\Omega_e} \boldsymbol{\nabla} \psi_i \psi_j \, d\Omega_e = \sum_{k=1}^{M_Q} w_k \mathcal{J}_k \boldsymbol{\nabla} \psi_{ik} \psi_{jk}$$

  is the differentiation matrix, which is nothing more than the multi-D version of the weak form differentiation matrix we have already seen.

- There is no problem integrating the above matrices exactly, except for the issue of having to deal with a non-diagonal mass matrix.

Integral Form of the Multi-D Equations

- However, in the DG method having a non-diagonal mass matrix poses very little difficulty since this matrix is small and local.
- Inverting the mass matrix yields the final matrix problem

$$\frac{dq_i^{(e)}}{dt} + \sum_{k=1}^{N_{face}} \left(\widehat{\boldsymbol{F}}_{ij}^{(e,k)}\right)^{\mathcal{T}} \boldsymbol{f}_j^{(*,k)} - \left(\widehat{\widetilde{\boldsymbol{D}}}_{ij}^{(e)}\right)^{\mathcal{T}} \boldsymbol{f}_j^{(e)} = 0$$

- where

$$\widehat{\boldsymbol{F}}_{ij}^{(e,k)} = \left(M_{il}^{(e)}\right)^{-1} \boldsymbol{F}_{lj}^{(e,k)}$$

and

$$\widehat{\widetilde{\boldsymbol{D}}}_{ij}^{(e)} = \left(M_{ik}^{(e)}\right)^{-1} \widetilde{\boldsymbol{D}}_{kj}^{(e)}$$

are the flux and differentiation matrices premultiplied by the inverse mass matrix.

Summary of DG Lecture 3

- At this point, we covered all of the necessary ingredients to build a DG model in multi-dimensions.
- Lecture 4 will show how we can use these tools to build a 3D compressible Euler solver for atmospheric modeling.
- Some things that will be discussed includes some subtle issues with solving Euler on spherical domains.
- Will also show some simulations of interesting atmospheric flows such as 3D hurricane and supercell simulations.

Resources

- Video lectures can be found here:
  `https://frankgiraldo.wixsite.com/mysite/ma4245`
- With PDF of Lectures and both Matlab and Julia code found here:
  `https://github.com/fxgiraldo/Element-based-Galerkin-Methods`