Restructured proposal for implementation of Discontinuous Galerkin Methods

Andrey Andreyev
Advisor: Dr. James Baeder

December 21, 2012

Abstract

The initial proposal submitted in September of 2012 discussing the implementation of the Discontinuous Galerkin (DG) methods and strand methods requires restructuring of the schedule and the deliverables due to difficulties encountered. The original proposal indicated that the one dimensional discontinuous Galerkin flow solver was scheduled to be completed by October 31, 2012, the two dimensional flow solver by February 2012, and the stand method grid generator by May 2013. As mentioned a substantial amount of difficulties were encountered in the implementation of the one dimensional solver. Currently the one dimensional version is implemented up to second order. Current work is focused on fixing the third order implementation. After the completion of the third order method focus will shift on implementation of the two dimensional flow solver. This proposal seeks to outline the restructure work schedule and deliverables.

1. Introduction

As mentioned, a series of difficulties with the implementation of the one dimensional flow solver has caused a substantial delay in the schedule of the proposed work. The original delivery schedule is no longer feasible. This proposal outlines a new schedule and deliverables eliminating the stand method work entirely and proposing a new systematic development of the two-dimensional Discontinuous Galerkin flow solver.

The difficulties encountered in the one dimensional implementation were initial conceptual problems with understanding the quadrature of the numerical fluxes, and the implementation of the projection flux limiters. The mathematical details of the quadrature and projection limiter are presented in the mid year report. When the proposal was initially drafted the author did not foresee the difficulties in the quadrature implementation. It was not until an example code was found from an online source at the end of October 2012 that the quadrature implementation became clear. However by this point in time the work flow was behind schedule. Once the quadrature was implemented and the weak form developed properly, the flux limiter presented quite a few issues because the references [2,3] used proved to be somewhat ambiguous in the implementation and it took some time and experimentation to arrive at a non-oscillatory method for limiting the flux. It should be noted that there are still issues in the 3rd order implementation.

Currently the one dimensional 2nd order Discontinuous Galerkin method has been implemented successfully. There is a bug in the 3rd order implementation which is currently the focus of the work. Once the bug is fixed, a grid convergence study will be performed to test the formal order of accuracy of the scheme completing the one dimensional implementation. Details of the one dimensional

Discontinuous Galerkin Method can be found in the mid-year report submitted earlier. Once the one dimensional solver is complete focus will be shifted to two-dimensional implementation.

The motivation behind the two-dimensional Discontinuous Galerkin flow solver is the same to that of the one-dimensional version. It allows for shock capturing abilities, does not require a wider stencil for higher order spatial discretization by simulating the higher order gradients with shape functions. The elimination of the higher order stencil is known as element locality [1] and also makes boundary treatment easier because there is no need to modify any stencils around domain boundaries.

2. Mathematical Background

Before beginning the background section of the two-dimensional DG method, it should be noted that there is still some work that requires to be completed on the one-dimensional GD method; mainly finding the bug in the third order implementation and doing a grid convergence study to confirm formal order of accuracy. The details of this will be given in the validation section. The mathematical background for the one-dimensional has been presented earlier in the mid-year report.

The development of the two-dimensional DG method follows the same procedure as the one-dimensional DG method.
To develop the two-dimensional DG method we first define the spatial discretization by seeking an approximate solution $u_h(x,t)$ to the two dimensional Euler Equations

$$\boldsymbol{U}_t + \nabla \cdot \boldsymbol{f} = \mathbf{0}$$

$$\boldsymbol{U} = [\rho, \rho u, \rho v, E]^T \quad \boldsymbol{F} = [\rho u, \rho u^2 + p, \rho uv, u(E+p)]^T \qquad (2.1)$$

$$\boldsymbol{G} = [\rho v, \rho uv\ \rho v^2 + p, v(E+p)]^T$$

by projecting the unknown solution **U** onto a finite element space of discontinuous functions [1] on a rectangular element

$$V_h = \{v_h \in L^\infty(\Omega) : v_h|_K \in V(K), \forall K \in \mathcal{T}_h\}, \qquad (2.2)$$

with the basis functions given in 2.3. Note that ϕ and ψ are v_h in this formulation.[1]

$$\phi_i(x) = \frac{x - x_i}{\Delta x_i/2}, \quad \psi_j(y) = \frac{y - y_j}{\Delta y_j/2}, \qquad (2.3)$$

We define $u_h(x,t)$ as

$$u_h(x,y,t) = \bar{u}(t) + u_x(t)\phi_i(x) + u_y(t)\psi_j(y) \qquad 2^{nd} \text{ order} \qquad (2.4)$$

$$u_h(x,y,t) = \bar{u}(t) + u_x(t)\phi_i(x) + u_y(t)\psi_j(y) + u_{xy}(t)\phi_i(x)\psi_j(y \qquad 3^{rd} \text{ order}$$

All terms other than the shape functions are the degrees of freedom of the system. We are now ready to define the weak formulation of the Euler Equations 2.1

$$\frac{d}{dt}\int_K u(x,t)\,v(x)\,dx + \sum_{e\in\partial K}\int_e \mathbf{f}(u(x,t))\cdot n_{e,K}\,v(x)\,d\Gamma - \int_K \mathbf{f}(u(x,t))\cdot \operatorname{grad} v(x)\,dx = 0 \qquad (2.5)$$

Note that the second term, just like in the one-dimensional case is the Riemann problem across the cell interface. The difference is that the interface is no longer a point, but a cell edge. In the case of rectangular elements they correspond to the edges of the rectangle. In the one-dimensional set up the variables at the cell interfaces were designated as – or +. In this case they are designated as *int or ext* meaning interior of the element (analogy -) and exterior of the (analogy +). The Riemann problem is solved as a one dimensional problem assuming that there is no flux variation along the cell edge. The third term is also similar to the one-dimensional case only this time the quadrature is performed in two dimensions. Also worth noting is that now there are cell normals which are required the flux perpendicular to the cell edge. In this implementation the elements will be rectangles making cell normal calculation trivial.

Time stepping is done using the exact same procedure as in the one-dimensional case using 3rd order Runge Kutta technique. [3]

$$(\mathbf{u}^h)^{(i)} = \sum_{l=0}^{i-1} [\alpha_{il}(\mathbf{u}^h)^{(l)} + \beta_{il}\,\Delta t\,\mathbf{L}_h((\mathbf{u}^h)^{(l)}, t^n + d_l\,\Delta t)], \qquad i = 1, \ldots, r,$$

$$(\mathbf{u}^h)^{(0)} = (\mathbf{u}^h)^n, \qquad (\mathbf{u}^h)^{(r)} = (\mathbf{u}^h)^{n+1}. \qquad (2.6)$$

$$\alpha_{10} = \beta_{10} = 1,\ \alpha_{20} = \tfrac{3}{4},\ \beta_{20} = 0,$$
$$\alpha_{21} = \beta_{21} = \tfrac{1}{4},\ \alpha_{30} = \tfrac{1}{3},\ \beta_{30} = \alpha_{31} = \beta_{31} = 0,$$
$$\alpha_{32} = \beta_{32} = \tfrac{2}{3},\ d_0 = 0,\ d_1 = 1,\ d_2 = \tfrac{1}{2};$$
$$\mathrm{CFL} = 1,$$

The flux limiting in two dimensions is also performed using the characteristic limiter presented in the mid-year report. It is repeated here for completeness. The limiting is done only the liner terms in equation 2.4. For 3rd order implementation, if limiting is required the non-linear term is set to 0.

$$\boldsymbol{u}_{int=}\boldsymbol{u}_j^1 + 6\boldsymbol{u}_j^2 + 30\boldsymbol{u}_j^3 = \boldsymbol{u}_j^1 + \tilde{\boldsymbol{u}}_j \qquad \boldsymbol{u}_{ext=}\boldsymbol{u}_{j+1}^1 - 6\boldsymbol{u}_{j+1}^2 + 30\boldsymbol{u}_{j+1}^3 = \boldsymbol{u}_{j+1}^1 - \tilde{\boldsymbol{u}}_{j+1} \qquad (2.7)$$

where $\boldsymbol{u}_j^l$ are the degrees of freedom. Note that the above equation is for a third order scheme. To obtain second order scheme, the third term is dropped in each equation.

The spatial discretization is complete. However, a local projection limiter still needs to be applied to stabilize the scheme around discontinuities.[2] A characteristic limiter is used. The procedure is outlined below.

Compute $\boldsymbol{u}_{j+1/2} = (\boldsymbol{u}_j + \boldsymbol{u}_{j+1})/2$ and calculate the Flux Jacobian $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ at $\boldsymbol{u}_{j+1/2}$. Calculate the matrices $\overline{Q}, \overline{Q^{-1}}$ which consist of right eigenvectors as columns and left eigenvectors as rows respectively. [3]

Calculate $\widetilde{\boldsymbol{u}_j}, \breve{\boldsymbol{u}}_{j+1}^1,\ \boldsymbol{u}_j^1,\ \boldsymbol{u}_{j+1}^1,\ \Delta_- \boldsymbol{u}_j^1,\ \Delta_+ \boldsymbol{u}_j^1,\ \Delta_+ \boldsymbol{u}_{j+1}^1$ and multiply each by Q^{-1} to project each of the variables onto the left eigenspace.

$\boldsymbol{a}^p = Q^{-1} \cdot \boldsymbol{a}$ where $\boldsymbol{a}$ represents all of the above variables. The goal is to limit $\widetilde{\boldsymbol{u}_j},\ \widetilde{\boldsymbol{u}_{j+1}}$ in the left eigenspace by applying the minmod limiter. A minmod function

$minmod(a, b, c)$ returns the lowest value unless there is a sign difference in which case it returns zero.[1]

$$\widetilde{\boldsymbol{u}}_j^{mod(p)} = minmod(\widetilde{\boldsymbol{u}}_j^{(p)}, \Delta_+ \boldsymbol{u}_j^{1(p)}, \Delta_- \boldsymbol{u}_j^{1(p}\ )$$

$$\breve{\boldsymbol{u}}_{j+1}^{mod(p)} = minmod(\breve{\boldsymbol{u}}_{j+1}^{(p)} \Delta_+ \boldsymbol{u}_j^{1(p)}, \Delta_+ \boldsymbol{u}_{j+1}^{1(p} \qquad 2.8)$$

The modified values are now used to project $\boldsymbol{u}_j^{1(p)},\ \boldsymbol{u}_{j+1}^{1(p)}$ onto the cell interface

$$\boldsymbol{u}_{j+1/2}^{-(p)} = \boldsymbol{u}_j^{1(p)} + \widetilde{\boldsymbol{u}}_j^{mod(p)} \qquad \boldsymbol{u}_{j+1/2}^{+(p)} = \boldsymbol{u}_{j+1}^{1(p)} - \breve{\boldsymbol{u}}_{j+1}^{mod(p)}$$

and project back on to component space by

$$\boldsymbol{u}_{j+1/2}^{\pm} = \overline{Q} \boldsymbol{u}_{j+1/2}^{\pm(p)}$$

This completes the development of the two-dimensional DG method.

3. Implementation

To avoid the issues encountered in the one-dimensional implementation, the development of the two-dimensional DG method will be much more systematic with focus on solving simpler sub-problems and adding complexity once each problem is solved. The method is outlined below.
The two dimensional implementation presents some difficulties although it is seemingly very similar to its one-dimensional counterpart. The first problem is the two dimensional gauss quadrature integration which is not as straight forward as the one-dimensional version. Thus the first step in the implementation is to develop independently the two-dimensional gauss quadrature integrator and test it against known and validated implementation such as Matlab.

Once this is complete the mesh will be generated. Initially, it was believed that the triangular mesh would be the easiest to implement. Upon further research, it was decided that a rectangular mesh

would be simpler and less error prone. This has many simplifying implications such as no need for a connectivity matrix and the fact that cell normals are now trivially in either the x or y direction. This is shown in figure 1 with the two arrows representing the cell normals. In the systematic approach complex geometries are not considered to make sure the underlying mathematics and numerics are implemented correctly.

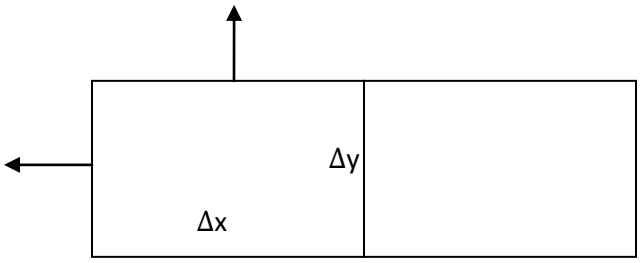

Figure 1. Basic schematic of the rectangular mesh

Following this the development of the two-dimensional DG method will continue with the cell interface Riemann problem and the calculation of the Gauss quadrature using the validated Gauss integrator mentioned above. Initially the projection limiter will not be considered and the solver will only be applied to problems with a smooth solution to validate the correct solution of the weak form (2.5) and Runge Kutta time-stepping (2.6). The problem used for this will the ideal vortex convection The test of the accuracy of the solution will be done using the mesh refinement study by measuring the reduction in the error as the element size is reduced. If the error is reduction is consistent with the order of accuracy of the scheme. . This will be discussed in more detail in the validation section.

Once the smooth solution is validated and there is confidence in the solver, the projection limiter will be applied to test shock capturing abilities of the method. The test problem for this will be the vortex-shock interaction problem. Again, the same validation procedure will be followed.

Once this is complete, and time permitting more complex geometry and boundary treatment will be added to allow for modeling of airfoils. This will only be done if the above two steps are completed before the middle of April 2013.

The final pseu-code implementations will look the same as the one-dimensional version with only the details of each step varying.

Step 1 calculate:

$$\frac{d\boldsymbol{u}_j}{dt} = -\boldsymbol{R}\boldsymbol{e}_j \qquad 3.1)$$

```
Calculate the initial degrees of freedom using the initial condition, DOFs are given by equation 2.1.2
time=0
do time=0: final_time
calculate time step dt using  equation 2.1.7
    do rk=1:nRungeKuttaSteps (3)
            Apply Flux Limiting
            Calculate the Residual according to equation 3.1
            Update solution to the rk^th step
    end do
    time=time+dt
end do
```

4. Validation

As mentioned earlier validation will be performed at each step outlined in the implementation section. As an aside, some validation needs to be performed on the one dimensional case first. This work will be performed. The third order method is encountering problems even for smooth solutions so the first step is to find the cause of this. After this is complete validation will be performed using an entropy wave convection problem which is given below.

$$\boldsymbol{U}(x,o) = \begin{Bmatrix} \rho_\infty + Asin(\pi x) \\ u_\infty \\ p_\infty \end{Bmatrix} \quad 4.1)$$

$$\boldsymbol{U}(x,t) = \begin{Bmatrix} \rho_\infty + Asin(\pi(x - u_\infty t)) \\ u_\infty \\ p_\infty \end{Bmatrix}$$

The solution is just the density (entropy wave) convecting to the right at velocity u_∞. [4]

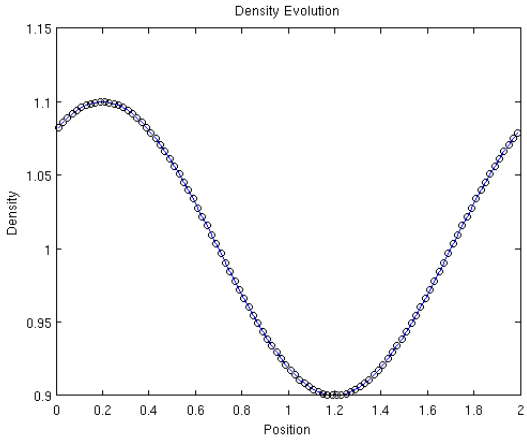

Figure 2. Entropy convection with numerical and exact solution

Defining the error as the difference between the calculated and exact solution we calculate the time average norm of the error using

$$E = \frac{1}{\Delta t}\int |u - u_h|_2 dt \approx \frac{1}{\Delta t}\sum |u - u_h|_2 \Delta t \qquad (4.2)$$

To confirm the accuracy of the schme, the element size will be systematically reduced while noting the trend in error reduction. The error should decrease as $O(\Delta x)$, $O(\Delta x^2)$, and $O(\Delta x^3)$ respectively. This is the trend that is sought in the two-dimensional version as well. Equation 4.2 will be used for all error calculations.

This validation is the very first step of the work going forward. Once this is complete, the implementation of the two-dimensional DG method will begin with the steps outlined above.

The validation of the two dimensional method will be carried out using the vortex convection problem without a limiter.[4] This is the two dimensional analog of the entropy wave convection problem. The vortex convection problem is defined as [4]

$$\begin{aligned} \rho &= \left[1 - \frac{(\gamma-1)b^2}{8\gamma\pi^2} e^{1-r^2}\right]^{\frac{1}{\gamma-1}} ;\ p = \rho^\gamma \\ \delta u &= -\frac{b}{2\pi} e^{\frac{1-r^2}{2}} (y - y_c) \\ \delta v &= \frac{b}{2\pi} e^{\frac{1-r^2}{2}} (x - y_c) \end{aligned} \qquad (4.3)$$

This vortex will simply convect downstream the same way as the entopy wave in the one dimensional case. The solution is shown in figure 3.

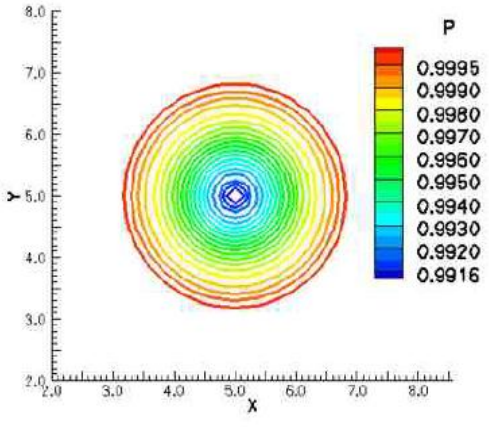

Figure 3. Vortex Convection problem [4]

The grid convergence study will be done the exact same way as the one dimensional case only this time the element reduction is measured in area instead of length of the element. Other than this detail, the study is carried out according to equation 4.2.

In the two dimensional case some numerical dissipation of the vortex as it convects downstream is expected. This should be captured in the error estimates; this does bring up an interesting aside of the dissipative nature of the scheme. In theory, the higher order methods should be less dissipative.

After this validation is complete and the flux limiter is implemented is tested using the vortex shock interaction. A schematic of the problem is shown in figure 4

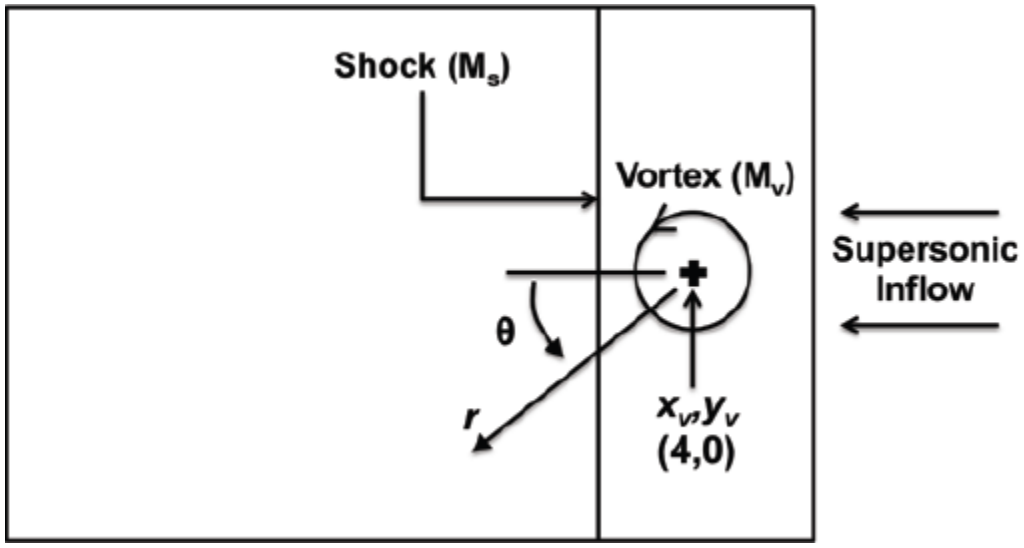

Figure 4. Vortex-shock interaction [4]

The solution to the problem is given by

$$\begin{aligned} \rho &= \left(1-\frac{1}{2}(\gamma-1)M_v^2 e^{1-(r/R)^2}\right)^{\frac{1}{\gamma-1}} \\ \delta u &= -M_v e^{\frac{1}{2}(1-(r/R)^2)}(y-y_v) \\ \delta v &= M_v e^{\frac{1}{2}(1-(r/R)^2)}(x-x_v) \end{aligned} \tag{4.4}$$

Once again the validation will be carried out using a mesh refinement study outlined in the beginning of this section.

The two vortex problems, once validated will provide sufficient confidence that the scheme is working properly on the interior points making it safe to start implementing boundary treatment. Since there is no guarantee that there will be enough time to implement it.

5. Revised Timeline and Deliverables

The original timeline and deliverables are shown below for reference:

10/31/12- Implementation and testing of the one dimensional version of the DG method

12/15/12- Implementation of two dimensional flow solver past an airfoil

02/15/13- Validation of the results using experimental and computational results

03/15/13- Parallelization of the code using MPI. Validation will be done by comparing the results from the serial version.

04/15/13- Implementation of the strand mesh generation. Validation is trivial since the problem is geometric in nature and visual inspection of the resulting mesh will suffice.

Time Permitting- Integration of the strand methods into the DG Flow Solver

End of Semester- Final Report

Deliverables:

- Parallelized code and executable
- Report Summarizing the results and accuracy
- Code and executable for mesh generation
- Final Report summarizing the results

It is clear with only the first step complete only for the 2^{nd} order one-dimensional DG flow solver, the timeline is no longer feasible.

A new timeline reflecting the encountered difficulties and lessons learned is given below

01/20/13- Complete and validated one dimensional Discontinuous Galerkin Code up to 3^{rd} order accurate in space

03/01/13- Two dimensional solution of the vortex convection problem to test spatial order of accuracy without limiting (analogous to entropy convection in one dimension). With validation.

04/01/13?- Two dimensional solution of vortex-shock interaction problem to test the shock capturing capabilities as well as the limiter. (Estimate delivery date because of possible unforeseen complications). With validation

Time permitting- Boundary treatment such as inviscid wall applied to allow for solutions to airfoils

Delivarables:

- Complete and validated flow solver for one dimensional Euler Equations
- Summary of the vortex convection problem including grid convergence studies

- Summary of vortex shock interaction problem
- Final Report and two dimensional code. Time permitting with boundary treatments.

6. Conclusion

With the difficulties encountered in the one-dimensional Discontinuous Galerkin Method, it is not feasible to complete the work initially proposed. The timeline had to be restructured to reflect a more realistic set of deliverables focusing only on the DG methods in two-dimensions while removing the strand mesh portion of the project entirely. The new timeline focuses on a systematic development of the two-dimensional method reducing the chance for error because each step is validated along the way.

References:

1 . Bernardo Cockburn, Chi-Wang Shu, *The Runge-Kutta Discontinuous Galerkin Method for Cnservation Laws V, Multidimensional Systems,* Journal of Computational Physics **141** 199-224 (1997)

2. Bernardo Cockburn, Chi-Wang Shu, *TVB Runge-Kutta Local Projection Discontinuous Galerkin Method for Conservation Laws III: One-Dimensioanl System,* Journal of Computational Physics **84** 90-133 (1989)

3. Bernardo Cockburn, Chi-Wang Shu, *TVB Runge-Kutta Local Projection Discontinuous Galerkin Method for Conservation Laws II: General Frame Work* Mathematics of Computation Volume **52** 186 (1989)

4. D. Gosh, *Compact-Reconstruction Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws.* Doctor of Philosophy Thesis, University of Maryland, College Park 2012.