AIAA SciTech
5-9 January 2015, Kissimmee, Florida
53rd AIAA Aerospace Sciences Meeting

Mesh deformation and shock capturing techniques for high-order simulation of unsteady compressible flows on dynamic meshes

Abhishek Sheshadri*, Jacob Crabill†, Antony Jameson‡
Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305

53rd AIAA Aerospace Sciences Meeting

High-order CFD has great potential for solving flows of industrial interest that are not quite possible today, such as the highly separated, vortex-dominated flows of high-lift systems. However, while conventional 2nd-order finite-volume codes have greatly matured in their range of capabilities and applications, the same cannot be said of high-order methods. The simulation of unsteady flows on arbitrarily deforming domains is one such capability that is yet to be fully adopted into high-order CFD. Likewise, shock-capturing schemes for high-order CFD are also in their infancy. In this work, we propose to combine the two capabilities in the high-order CFD code HiFiLES,[1] and demonstrate their functionality and utility on representative test cases.

Traditional shock capturing methods which use physical quantities and their gradients to detect the presence of a shock are not suitable in unsteady flows with vortices and boundary layers, especially when mesh deformation is involved. Several methods which detect discontinuities locally in an element using the polynomial spectral data have been proposed [2] [3] [4] . These methods also have the added advantage that they seamlessly adapt to the deforming mesh environment we plan to implement. The method of concentration used in signal edge detection has shown great potential as an effective shock indicator, even in the presence of various flow features [2] . We evaluate the performance of the concentration method coupled with the exponential modal filter in presence of moving meshes in this paper.

Elastic mesh deformation could also facilitate the possibility of a mesh-deformation mechanism that is coupled with the shock capturing scheme to produce finer shocks by increasing the resolution near them. This would remove the requirement of having additional elements and the resulting mesh storage structures required in the traditional mesh adaptation and enable us to simulate complex flows using coarser meshes by being more efficient. Moreover, we believe that the idea of detecting flow structures or patterns using the polynomial modal data in the elements and using this information to selectively enhance resolution in those areas has a great potential in improving the efficiency and capability of these schemes. For example, this concept could be used as a stabilization mechanism in turbulent flows, drastically reducing the simulation effort through relaxed meshing requirements. The main focus of this paper is to investigate different mesh adaptation algorithms for their suitability to being adopted into the 2D and 3D Flux Reconstruction scheme. In this paper we propose and compare 3 different meshing algorithms in 1D and discuss their advantages and disadvantages. We also show preliminary 2D results for the adaptation algorithm.

*Ph.D. Candidate, Department of Aeronautics and Astronautics, Stanford University; abisheks@stanford.edu;
†Ph.D. Candidate, Department of Aeronautics and Astronautics, Stanford University; jcrabill@stanford.edu;
‡Thomas V. Jones Professor of Engineering, Department of Aeronautics and Astronautics, Stanford University, AIAA Member

I. Introduction

The ability to handle deforming meshes is imperative in order to simulate many flows of practical interest. Many aerodynamics applications also require the ability to handle compressible flows effectively. One important motivating case for the use of high order CFD is the computation of the highly unsteady and vortex-dominated flow field around a helicopter. Such a simulation would require both these capabilities. Other examples would be the simulation of turbine blades, wing flutter in transonic flows, or even birds and insects for bio-inspired flight concepts. Mesh deformation could also complement shock capturing by allowing us to elastically deform the mesh to have more elements near the shock so as to capture it in a finer fashion.

Currently we have developed shock capturing capability in 2D on both quads and triangles in HiFiLES.[21] We use the method of concentration[2] in quads and use the method proposed by Persson and Peraire[5] on triangles for shock detection. We have modal filtering as well as addition of artificial viscosity as tools for capturing the shock effectively. The shock detection is made up only of element-local operations which can be parallelized effectively on multiple GPUs. If addition of artificial viscosity is used to capture the shock, enforcement of continuity of artificial viscosity coefficients has a footprint over a few elements neighboring those containing shocks as well. The artificial viscosity method also requires quite a few fine-tuned input parameters and mesh motion might increase the complexity of choosing these parameters. Due to this we believe that modal filtering might be a lot more suitable in such cases.

In order to simulate fluid flows on arbitrarily deforming domains, the Arbitrary Lagrangian-Eulerian (ALE) formulation of the Navier-Stokes equations must be utilized, and has also been implemented into HiFiLES. This form of the equations can be reached through derivation of the conservation equations on a moving control volume, or through the use of a coordinate transformation from a static to a dynamic domain. The usual approach has been the first option, as the resulting integral form is most suitable for standard finite-volume codes. On the other hand, the coordinate transformation approach - although mathematically equivalent - is more directly applicable to high-order codes, and will be described in detail in the following sections. Mesh deformation capability has also been implemented into HiFiLES, using an adaptation of the linear-elasticity method implemented in the SU^2 code package[6] and first described by Dwight.[7]

Traditionally mesh adaptation in high-order CFD has been done using either h or p adaptation or both. However, for unstructured complex meshes, h-adaptation is not easy to implement as a continuous adaptation procedure since it needs complete re-meshing and re-interpolation. It also needs additional storage structures as the number of elements might change significantly. With this in view, we propose to investigate the strategy of adapting the existing mesh by just moving elements around to reinforce areas of interest in the flow at the expense of far-field regions. This method known as r-adaptation is in its infancy in the high-order CFD domain and we explore some of ideas in this direction. In particular, we investigate the approach of using an elasticity model for the mesh elements and applying body forces on them to obtain equilibrium solutions of the mesh-structure that are adapted to the finer flow features in the domain. Beyond shock capturing, this sensor-based body-force has the possibility of being applied to other flows of interest, easing the meshing requirements for generating accurate results. For example, it could be possible to develop a sensor based on vorticity which would refine the vortex structure behind a high-lift system, or one which would detect the presence of a boundary layer and cluster points there. This method is much simpler and potentially more computationally efficient than methods based on the usual h- p-adaptation, in which degrees of freedom are added by either adding additional elements or increasing the solution polynomial order. Ou[8] has already shown the promise of an optimization-based r-adaptation method for the high-order Spectral Difference method in the context of a 1D scalar conservation law, and we expect to extend this idea to 2D and 3D Euler and Navier Stokes simulations using the Flux Reconstruction method.

II. Shock Capturing Technique

The idea of elemental shock detection and capturing we propose to use has been presented in.[2] Shocks are detected in each element using the method of concentration, wherein the polynomial modes are summed up along with certain concentration factors in order to "concentrate" on the points of discontinuity. In order to

exxagerate the difference between a point of discontinuity and one which is not, a technique called non-linear enhancement can be further employed. The maximum value of the calculated enhanced kernel inside an element is then allocated the elemental shock-sensor value. We then use an exponential modal filter of the form used in[?] but modified to incorporate the boundary information from the neighboring elements. The advantage of this system for shock detection and capture is that it is completely element local and can be parallelized. Also, this formulation does not interfere with the added flux term coming from mesh motion. In fact, the added capability of deformable meshes only provides us with an additional tool to enable finer shock capturing.

III. Mesh Motion

There are several approaches with which we can incorporate mesh motion into these high order FEM schemes, but the most common among them is Arbitrary Lagrangian Eulerian Formulation (ALE). The advantage of ALE is that it provides a single formulation or framework for structured and unstructured 3D meshes and remains mostly unaltered irrespective of the particular high order scheme in use. The ALE method modifies the governing flow equations to incorporate flux and source terms coming from the mesh motion. Sometimes, in order to avoid addition of source terms to the scheme, the method can also be re-cast in conservative form with no source terms, but with an additional equation to be solved referred to as the Geometric Conservation Law.

While the ALE method offers a lot of flexibility, since the mesh motion is introduced in the form of fluxes and source terms, the amount of deformation the mesh can undergo in a single time-step is quite limited. In other words, large mesh velocities can cause serious problems, not only in terms of time-step restrictions but could cause its own numerical discontinuities which need to be handled. When we detect a discontinuity in the flow field and want to immediately re-adapt the mesh to have higher resolution near the shock, great care needs to be taken to make sure the mesh velocities are not overwhelming the scheme. Additionally, care needs to be taken during evaluation and interpolation of the mesh velocities in order to preserve accuracy of the high order scheme.

In view of the above we also explore the possibility of interpolation across meshes as an alternative. In theory, this can be a very powerful approach because it is not constrained by mesh velocities or small deformations. However, the biggest disadvantage of this method is that accuracy-preserving interpolation between 2D or 3D unstructured meshes becomes complicated and expensive. One could argue that, since interpolation can handle large mesh movements at once, we need to do this only once in a few time steps and can thus reduce the computational effort. In this paper we use the method of interpolation to experiment with various methods of mesh deformation to conform to a shock since interpolation gives us a robust method for handling large movements. Once a good motion algorithm is decided upon, we could then configure it to work with the ALE formulation which is much more sensitive to large deformations.

In the following sub-sections, we briefly describe both the interpolation technique and the ALE formulation we use.

A. Interpolation

As mentioned earlier, interpolation of the solution across unstructured meshes can be computationally tedious. Also, it is not obvious if the accuracy of the scheme is preserved. We show a few numerical results regarding accuracy but a thorough theoretical investigation is one of our future objectives. In this paper, we mainly use this as an investigative technique to evaluate different mesh-adaption algorithms in 1D.

For interpolating the solution on the original mesh to an adapted mesh, we just find the element in the original mesh which contained a particular solution point of the new mesh and calculate the value of the solution at that point in the original mesh using the polynomial representation of the solution in that element. We use the element-local polynomial representation instead of any other methods of interpolation so that we can hope to preserve the order of accuracy of the solution.

B. ALE Formulation

1. Derivation of the ALE Formulation

As previously stated, the conservation equations must be modified to allow for moving and deformable domains; we will choose the ALE method. For convenience with the Flux Reconstruction (FR) scheme, we will use the coordinate transformation approach to arrive at the ALE equations. First, we shall start with the two-dimensional Navier-Stokes equations in conservation form:

$$\frac{\partial U^p}{\partial t} + \frac{\partial F^p}{\partial x} + \frac{\partial G^p}{\partial y} = 0$$

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho(e+k) \end{bmatrix}, \quad F = F_{inv}(U) + F_{vis}(U, \nabla U)$$

where the p superscript denotes that the quantities are seen in the physical reference frame, which may be moving through space. Next, we will define a time-dependent coordinate transformation to a static reference domain of isoparametric elements:

$$(\xi, \eta) = \mathbf{T}(x, y, t)$$

By applying the chain rule to each term in the conservation equations, the equations in terms of reference-domain derivatives become:

$$\frac{\partial U^p}{\partial t} + \frac{\partial F^p}{\partial x} + \frac{\partial G^p}{\partial y} = \frac{\partial U^p}{\partial t} + \xi_t \frac{\partial U^p}{\partial \xi} + \eta_t \frac{\partial U^p}{\partial \eta} + \frac{\partial U^p}{\partial t} + \xi_x \frac{\partial F^p}{\partial \xi} + \eta_x \frac{\partial F^p}{\partial \eta} + \xi_y \frac{\partial G^p}{\partial \xi} + \eta_y \frac{\partial G^p}{\partial \eta} = 0$$

After rearranging the above terms on the right hand side, the governing equations in the fixed reference domain can be shown as:

$$\frac{\partial J U^p}{\partial t} + \frac{\partial J(\dot{\xi} U^p + \xi_x F^p + \xi_y G^P)}{\partial \xi} + \frac{\partial J(\dot{\eta} U^p + \eta_x F^p + \eta_y G^P)}{\partial \eta} - \frac{\partial (J)}{\partial t} - \frac{\partial (J\dot{\xi})}{\partial \xi} - \frac{\partial (J\dot{\eta})}{\partial \eta} = 0$$

where J is the determinant of the transformation Jacobian matrix:

$$\mathbf{G}(t) = \begin{bmatrix} \frac{\partial X}{\partial \xi} & \frac{\partial Y}{\partial \xi} \\ \frac{\partial X}{\partial \eta} & \frac{\partial Y}{\partial \eta} \end{bmatrix}, \quad J(t) = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}$$

By defining the transformed solution and flux vectors in the reference domain as follows:

$$U^r = J U^p$$

$$\begin{bmatrix} F^r \\ G^r \end{bmatrix} = J \begin{bmatrix} \xi_x F^p + \xi_y G^p \\ \eta_x F^p + \eta_y G^p \end{bmatrix} - J U^p \begin{bmatrix} \dot{\xi} \\ \dot{\eta} \end{bmatrix} = J \left[G\right]^{-1} \left(\begin{bmatrix} F^p \\ G^p \end{bmatrix} - U^p \begin{bmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{bmatrix} \right)$$

where $\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}$ refer to the motion of the grid in the physical domain, then the transformed system of equations becomes:

$$\frac{\partial U^p}{\partial t} + \frac{\partial F^p}{\partial x} + \frac{\partial G^p}{\partial y} = \left(\frac{\partial U^r}{\partial t} + \frac{\partial F^r}{\partial \xi} + \frac{\partial G^r}{\partial \eta} \right) - \left(\frac{\partial (J)}{\partial t} + \frac{\partial (J\dot{\xi})}{\partial \xi} + \frac{\partial (J\dot{\eta})}{\partial \eta} \right) = 0$$

If we assume for the moment that the determinant of the transformation matrix (the transformation Jacobian; effectively the cell volume) remains constant in time, then the transformed ALE Navier-Stokes equations in the reference domain map back to the same equation:

$$\frac{\partial U^r}{\partial t} + \frac{\partial F^r}{\partial \xi} + \frac{\partial G^r}{\partial \eta}$$

meaning that, with the addition of a single term in the flux definitions, the Flux Reconstruction scheme (and other schemes which utilize a coordinate transformation) is applicable to solutions on arbitrarily moving domains, so long as the Jacobian and the transformed velocities remain constant.

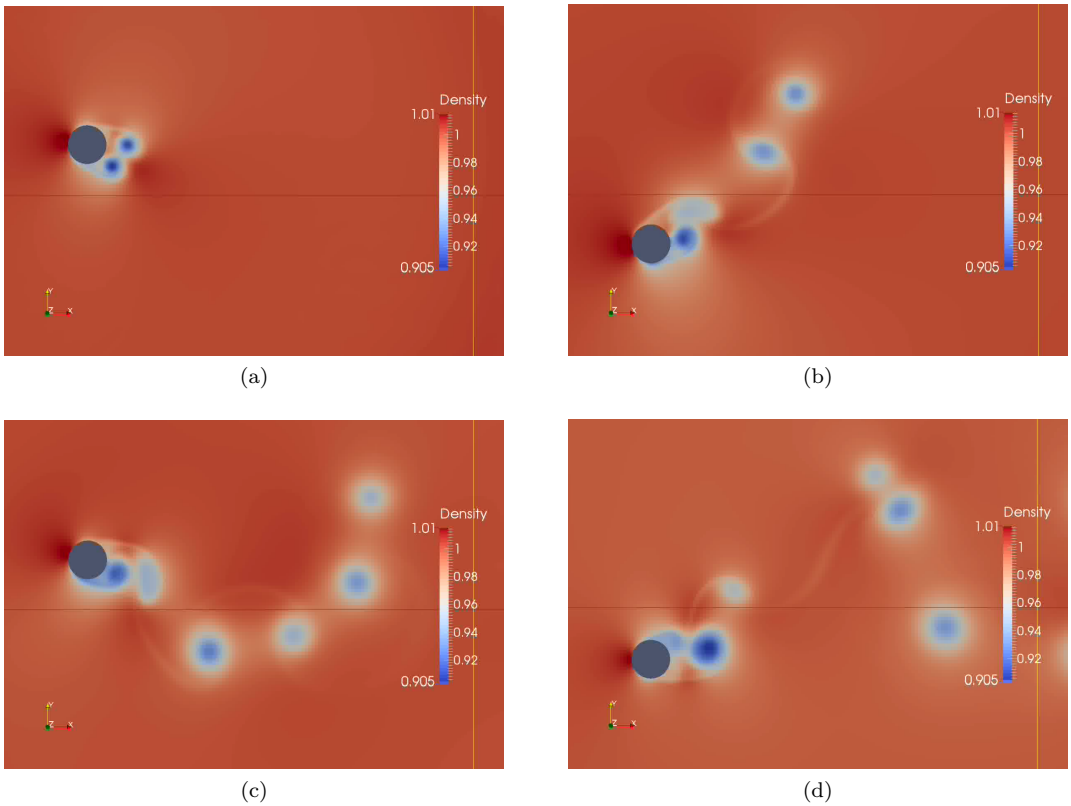


Figure 1: Density contours in viscous subsonic flow at M = .2, Re = 400, of a cylinder of diameter 1, oscillating with the prescribed motion $y(t) = \frac{4}{3}\sin(\pi t/5)$. The polynomial order is 4, with a quadrilateral mesh of 5716 elements. The mesh is rigidly moved. (a) t=.4 s, (b) 1.2s, (c) 2.0s, (d) t=2.8s

C. Satisfying the Geometric Conservation Law

The assumption of constant Jacobians is a reasonable approximation in many cases; for instance, rigid rotation and translation satisfy this exactly, and for small deformations, mesh element volumes may remain nearly constant. However, for simulations involving large deformations of the mesh - which is likely to occur, for example, with multiple bodies in relative motion - these geometry-dependant sources may produce large errors in the results. These terms can also be seen as relating to freestream preservation on an arbitrarily deforming grid, meaning that a constant solution will not be maintained without the GCL. This is described in more detail in.[9] In order to make these source terms disappear, it must be treated as an additional conservation equation, known as the Geometric Conservation Law (GCL):

$$\frac{\partial J}{\partial t} + \nabla_r \bullet \left(J\mathbf{G}^{-1} \begin{bmatrix} x_t \\ y_y \end{bmatrix} \right) = 0$$

The GCL must be integrated in time along with the solution of the Euler or Navier-Stokes equations. In fact, for explicit time-stepping, this equation can and, in general, must be integrated ahead of the Navier-Stokes terms, as the updated value of J is required throughout the residual calculation process. Additionally,

the same spatial discretization must be used for the GCL as for the fluid equations in order to properly satisfy the full system of equations. More details on the proper enforcement of the GCL, along with successful applications to similar high-order methods, have been discussed by Persson[10] and Ou.[9]

The ALE formulation has been implemented into the Flux Reconstruction code HiFiLES and preliminary results are encouraging. Here we show the ALE method working for a simple rigid motion of the mesh without any shocks. Figure 1 shows the flow around an oscillating cylinder simulated at a Mach number of 0.2 and Reynolds number of 400. We have used 4th order quadrilateral elements and rigid mesh motion for the simulation. It can be seen from the figure that the no-slip boundary conditions are satisfied. Although the GCL has not yet been implemented, freestream preservation has been demonstrated for arbitrary rigid motion. The linear elasticity mesh deformation method has also been successfully implemented and applied to simple cases of boundary motion.

IV. Mesh Deformation for Improved Shock Capturing

Intelligent adaptation of the mesh by repositioning points to cluster in areas of interest, referred to as r-adaptation or r-refinement, has great potential for increasing the accuracy of both steady and unsteady flows. The linear elasticity method for mesh deformation has already shown to be highly robust for even large boundary displacements.[7] In this section we explore several approaches to adapting a mesh by just moving the elements instead of adding new ones. We explain these methods in 1D for simplicity and discuss the ease of adopting these to 2D and 3D. Each of these methods can theoretically be combined with either Interpolation or the ALE formulation, but the use of ALE needs a lot more care in order to not incur very high mesh velocities.

A. Stretching function

The simplest and most direct ideas in 1D to refine the mesh towards a particular known location would be to use to stretching function to re-mesh the domain. The stretching/contraction offered by the function can be tuned and we can investigate both local accuracy near the shock and global accuracy for different degrees of contraction. We have tried various stretching functions including $tanh(x)$ and $x^{1/3}$.

The stretching functions are relatively easy to design for the case of single shock location. Our shock sensing mechanism gives us a way to measure the 'strength' of the shock and can also be normalized. So the intensity of contraction can be scaled by the strength of the shock so that stronger shocks cause higher degree of refinement of the mesh towards the shock than weaker ones.

We show some results for the 1D Burgers equation

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} = 0 \quad x \in [-1, 1]; t \geq 0$$

with the initial condition

$$u(x, 0) = \begin{cases} 2 & \text{for } x \leq -0.5 \\ 1 & \text{for } x > -0.5 \end{cases}$$

The exact solution for this case has the shock advecting through the domain, i.e. the exact solution is

$$u(x, t) = \begin{cases} 2 & \text{for } x - 3t \leq -0.5 \\ 1 & \text{for } x - 3t > -0.5 \end{cases}$$

Figure 2 shows the numerical solution compared to the exact one for $t_f = 0.3$ for cases with and without mesh adaptation by stretching. We use the 5th order Flux Reconstruction method with just 5 elements in order to emphasize the differences between the various cases clearly. As we can see from the figure, the stretching function clearly helps in capturing a sharp shock. In this simple case, the possible loss of accuracy away from the shock due to larger elements is not readily evident. But we use this as a simple test case to

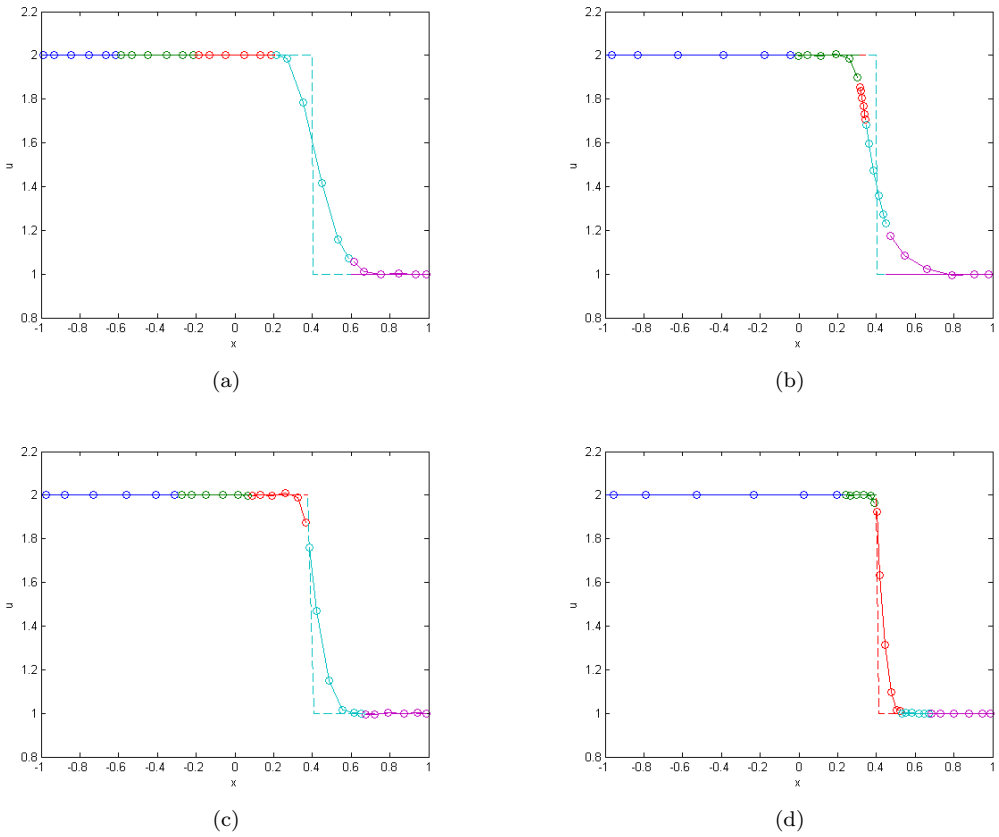


Figure 2: Figure shows the solution of Burgers' equations at $t = 0.3$ with (a) No stretching or adaptation (b) using a cube root ($x^{\frac{1}{3}}$) based stretching function (c) using a $tanh(x)$ based stretching function (d) using a $tanh(3x)$ function

see the performance of the different methods in terms of stretching.

The stretching function method is very intuitive and easy for a single shock location. It also provides very precise control over the mesh refinement process. However, it is not easy to extend this to multiple shock locations even in 1D. In 2D and 3D this becomes an extremely complicated task even if we were to know the exact orientations and locations of the shocks. So this method is just used as a baseline case to compare other methods in 1D.

B. Mesh adaptation as an optimization problem

Mesh adaptation can be cast as an optimization problem which minimizes the volume of a cell where the gradients (or any other chosen weights) are large. Consider the following optimization problem in 1D

$$\underset{x}{\text{minimize}} \quad I(x) = \int_a^b w(\xi)\frac{\partial x}{\partial \xi}dx$$

This tries to reduce the jacobian $J = \frac{\partial x}{\partial \xi}$ wherever $w(\xi)$ is large, thereby giving a distribution of points adapted to regions where the weights are large. A natural option for the weighting function is the gradient of the solution. We choose the normalized magnitude of the gradient as our weight function i.e.

$$w(\xi) = \frac{\left|\frac{\partial u}{\partial \xi}\right|}{max(\left|\frac{\partial u}{\partial \xi}\right|)}$$

Instead of trying to solve the above optimization problem exactly, we can take a step in a direction which ensures reduction of $I(x)$ as done by Jameson et al. We can write $I(x)$ as

$$I(x) = \int_a^b w(\xi)\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \xi}d\xi = \int_a^b w(\xi)\left(\frac{\partial x}{\partial \xi}\right)^2 d\xi$$

If we let $f(x_\xi) = w(\xi){x_\xi}^2$ we have

$$\delta I = \int_a^b [f(x_\xi + \delta x_\xi) - f(x_\xi)]d\xi = \int_a^b \frac{\partial f}{\partial x_\xi}\delta x_\xi d\xi$$

.

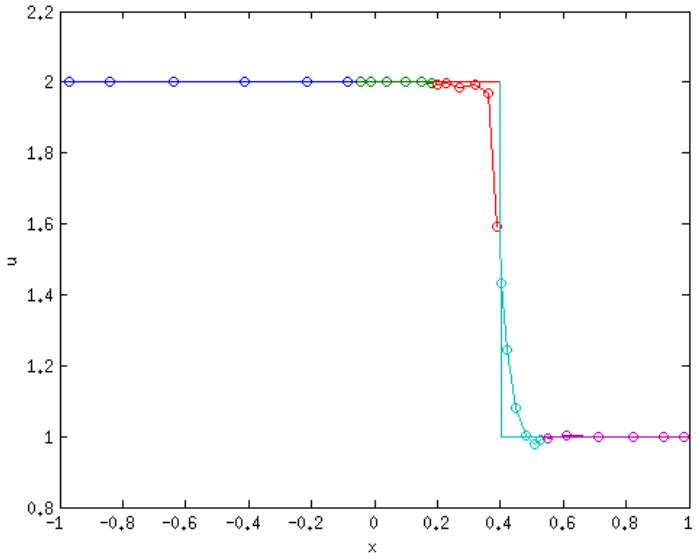


Figure 3: Figure shows the Burgers' equation solution at $t = 0.3$ using the gradient-based optimization algorithm

If we integrate the above by parts we get

$$\delta I = \frac{\partial f}{\partial x_\xi}\delta x\Big|_a^b - \int_a^b \frac{\partial}{\partial \xi}\left(\frac{\partial f}{\partial x_\xi}\right)\delta x d\xi$$

Now if we constrain the left and right boundary points from moving, the $\delta x|_{x=a} = \delta x|_{x=b} = 0$, which gives us

$$\delta I = -\int_a^b \frac{\partial}{\partial \xi}\left(\frac{\partial f}{\partial x_\xi}\right)\delta x d\xi$$

Therefore if we take

$$\delta x = \frac{\partial}{\partial \xi}\left(\frac{\partial f}{\partial x_\xi}\right)$$

We get

$$\delta I = -\int_a^b \left(\frac{\partial}{\partial \xi}\left(\frac{\partial f}{\partial x_\xi}\right)\right)^2 d\xi$$

which guarantees that we reduce I with every step δx we take. Therefore the mesh deformation at each step is given by

$$x^{new} - x^{old} = \lambda \delta x_{min} = \lambda\frac{\partial}{\partial \xi}\left(\frac{\partial f}{\partial x_\xi}\right) = \lambda\frac{\partial}{\partial \xi}\left(2w(\xi)x_\xi\right) = 2\lambda\frac{\partial}{\partial \xi}\left(w(\xi)J\right) = \lambda G$$

where J is the Jacobian and λ is the step size in the objective-reducing direction δx_{min}. We could theoretically use this directly to update our mesh, however notice that G needs the calculation of the second gradient of the solution (since $w(\xi)$ already contains the gradient of the solution). This could make G eccentric and non-smooth even if we start with a smooth solution, especially since we need to extrapolate the values from within the element to the boundary points in Flux Reconstruction. So we can further smooth this G to obtain a smoothed version $\bar{G}$ by solving the following second order ODE with a diffusion term for $\bar{G}$ as suggested by Jameson et al

$$\bar{G} - \epsilon \frac{\partial^2 \bar{G}}{\partial x^2} = G$$

We can solve for $\bar{G}$ using either a tridiagonal or pentadiagonal implicit system over the boundary/node points of the elements in the domain. We can then update the mesh as

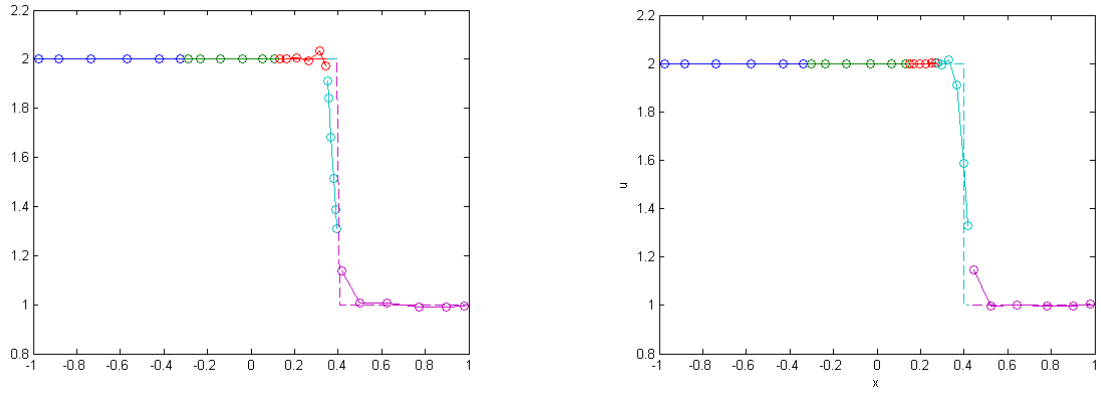
$$x^{new} = x^{old} + \lambda \bar{G}$$

We are currently working on extending this approach to 2D and 3D. Here we show a result with this method for the 1D Burgers' equation case discussed in the previous section. We set $\lambda = \Delta t$ so that we will have $\frac{\Delta x}{\Delta t} = \bar{G}$, where $\frac{\Delta x}{\Delta t}$ signifies the mesh velocity. The larger the value of ϵ, the smoother the $\bar{G}$ will be. However, if the ϵ value is too large, then the weighting function will be smoothed significantly and the mesh adaptation will be limited strongly.

C. Spring-attractor system

In this approach, we consider each shape point (or vertex) as a mass and each edge as a spring. In 1D, the element itself is the spring and the vertices are masses. The springs are assumed to have a fixed spring constant k which could either be a constant or might itself depend on the extension or contraction of the spring. Once the elements with a shock are detected, we place an 'attractor' in those elements which apply an attractive force towards itself. We have considered 2 models for the attractor:

1. Local attractor model: The attractor is placed at the centers of the elements with discontinuities and a constant attraction force (based on the shock strength) acts only upon the 2 masses neighboring masses. As these masses move towards the attractor, they in turn pull the other masses.
2. Global attractor model: In this model, we place the attractor at the exact shock location within the element as determined by the sensor and the attractor applies a force proportional to $^1/_r$ on the masses, where r is the distance of the mass from the attractor.



(a) (b)

Figure 4: (a) shows the Burgers' equation solution at $t = 0.3$ for the local spring-attractor model (b) shows the same for the global model

We then solve for a force-equilibrium solution for the spring-mass system by using a time-marching technique. Let $\vec{x_0}$ represent a uniform distribution of points in the domain. Let $\vec{x}$ be the current location of

the vertices (masses). Then we can find the equilibrium locations of the massesby finding the steady state solution to the following system

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x})$$

where

$$\vec{F} = \vec{F}_{spring} + \vec{F}_{attractor}$$

The spring force is given by $k(L - L_0)$ where L is the new length of the spring and L_0 is the original length. The attractor strength is set proportional to the shock strength from the concentration method. The spring constant k is made inversely proportional to the Jacobian so that, as we march the system towards steady state or equilibrium, as the element gets smaller, its stiffness increases preventing it from inverting or getting too small. Figure 4 shows the result for the Burgers' equation using both the local and global spring-attractor models. Both the local and global attractor models work quite well although it generally takes longer to find the equilibrium solution using the global attractor model.

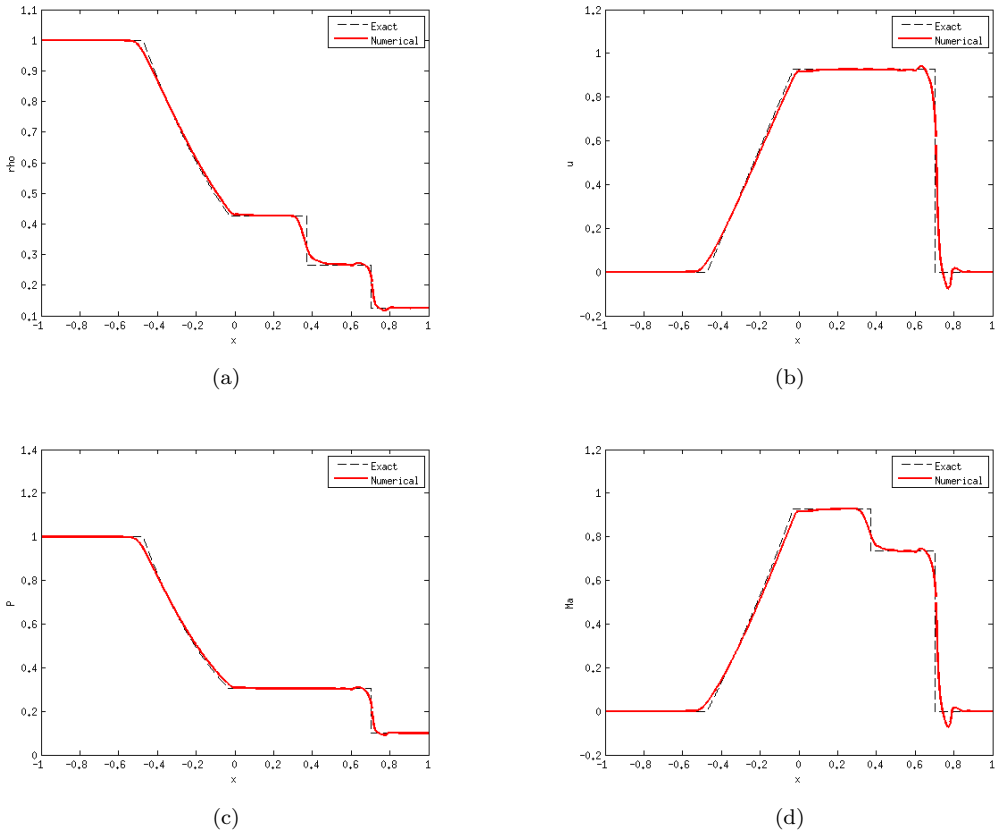


Figure 5: Figure shows the solution of the Sod Shock tube problem at $t = 0.4$ with 20 elements and 5th order Flux Reconstruction method. The global spring-attractor model is used for mesh adaptation to the shock locations (a) shows the density (b) shows the velocity (c) shows the Pressure (d) shows the Mach number. The dashed line in black shows the exact solution

The major advantage of this method is that it can be used for problems with multiple shocks and higher dimensions. In Figure 5 we show an application of this method to Sod's shock tube problem[11] with multiple discontinuities. Again, we use a 5th order method with only 20 elements in order to magnify the fact that it is possible to capture fine shocks with a coarse mesh by adaptation of the mesh. Tables 1 and 2 show the errors, maximum overshoot and total variation of the solution for the cases with and without mesh adaptation. It is evident that in this case, the error is significantly reduced. The additional cost of the mesh adaptation algorithm can be minimized by performing the adaptation once in several time steps. The

	2-norm of error	Infinity norm	Maximum overshoot	Total Variation
rho	0.02355	0.13025	0.13025	0.63016
u	0.11909	0.88995	0.88995	2.24477
P	0.02557	0.18369	0.18369	0.55807
Mach	0.09976	0.71345	0.71345	2.10184

Table 1: Errors for the Sod Shock Tube Case without mesh adaptation

	2-norm of error	Infinity norm	Maximum overshoot	Total Variation
rho	0.02086	0.10065	0.10065	0.60465
u	0.07353	0.67881	0.67881	1.95400
P	0.01863	0.13166	0.13166	0.49226
Mach	0.06522	0.56622	0.56622	1.91312

Table 2: Errors for the Sod Shock Tube Case with mesh adaptation using spring-attractor model

interpolation method for handling the mesh motion is ideal for this as it can handle large mesh motions.

This method works quite well and the advantage of this method is that it can easily be extended to higher dimensions. We now show some 2D results using an extension of the force-equilibrium method to 2D using a similar force-equilibrium approach on an elastic system.

D. 2D Methods: Linear Elasticity and "Marching" Displacement-Blending

For 2-dimensional problems, we are currently working on two different methods for adapting the mesh towards a chosen set of elements. The first method is the linear-elasticity method, in which the plane-stress equations of elastostatics are solved while applying point forces to force points towards the selected "shock cells". The second method involves marching outwards, along edges, from the vertices of the selected cells and applying a displacement based upon the length of each edge and its distance from the shock.

For the linear-elasticity method, we utilize a simple first-order continuous Galerkin finite-element method to solve the equations of elastostatics (in 2D, simplified to the plane-stress equations) on arbitrary unstructured grids. The method has the advantage of also being able to include arbitrary motion of any boundaries at the same time, allowing for a unified framework of mesh motion.

In order to prevent any elements from collapsing too quickly or inverting, the stiffness of each element is taken to be proportional to $area^{-q}$, where q is a parameter which can be tuned to suit the current case; in the current test, we use $k = .7$. Additionally, multiple iterations are performed, so as not to destroy the quality of the mesh by moving points too far at once. To force points towards the shocks, the force applied to each point is:

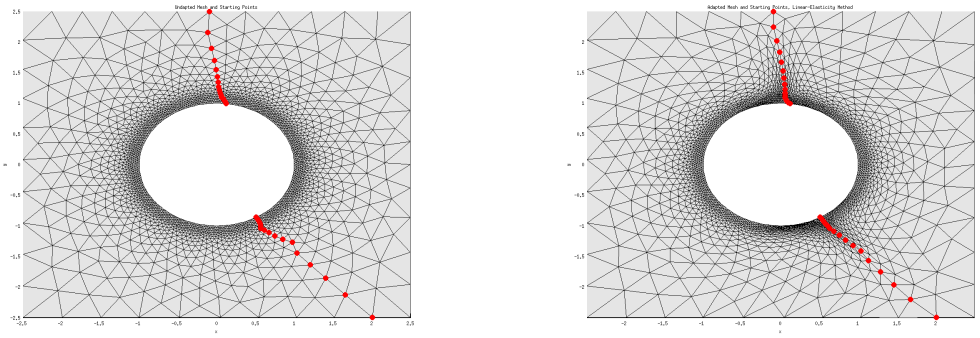
$$\vec{F} = F_{mag} F_{blend} \frac{\vec{D}}{|\vec{D}|}$$

$$F_{blend} = \begin{cases} \sqrt{\frac{|\vec{D}|}{B}} - \frac{|\vec{D}|}{B}, & : |D| < B \\ 0 & : |D| \geq B \end{cases}$$

where F_{mag} is an overall scaling factor which can be tuned for more or less adaption, $\vec{D}$ is the distance vector to the nearest shock point, and B is the distance over which the force is to be blended. The above function provides a smooth blending of the force away from the shock, with points very near the shock being left alone.

Figures 6 and 7 show the results of adapting to simulated "shocks" around a circular cylinder.

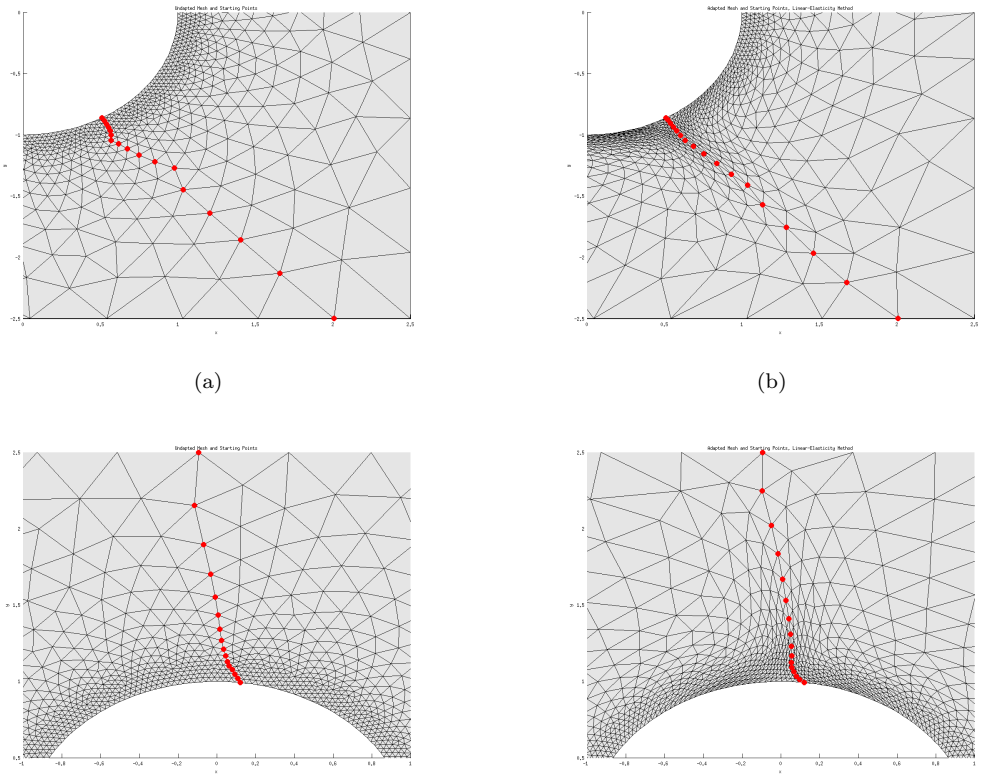
The marching displacement-blending method is named after the Fast-Marching Method (FMM), used to compute level-set functions on structured or unstructured grids. The idea is to take the vertices of the shock



(a) (b)

Figure 6: Adaption of an unstructured triangular mesh around a cylinder to pre-specified shock locations, using the linear elasticity method. The original mesh is shown in (a), and the final mesh in (b).

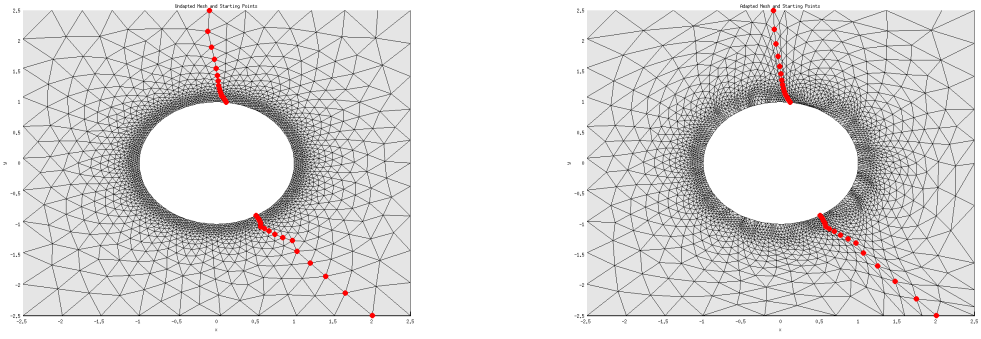
cells as starting points, and march outwards along edges in a manner similar to FMM. At each "active" node, the displacement is specified by scaling the average displacement of the "dead" nodes it is attached to, along with the averaged direction vector of the edges to those nodes, by use of the same blending function utilized in the linear-elasticity method. As shown in Figures 8 and 9, this method has the benefit of providing excellent refinement near the shock; however, the stretching which occurs near the boundary is problematic, and will be an area of improvement for the future.



(a) (b)

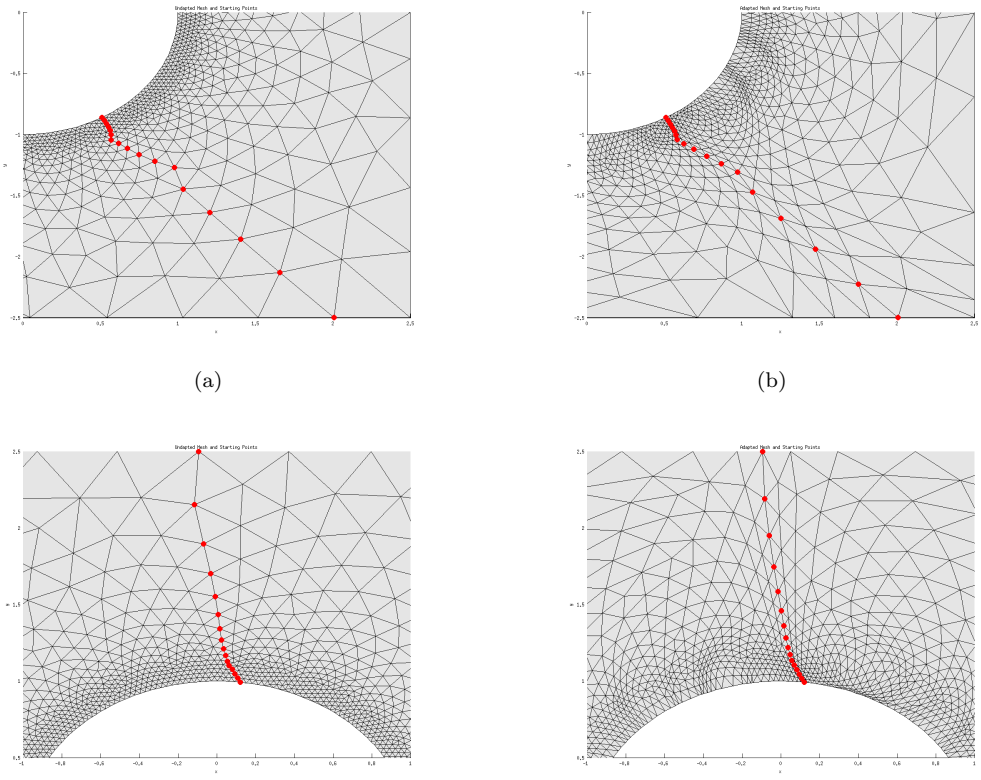
(c) (d)

Figure 7: Close-up view of the linear-elasiticy mesh adaptation. (a) and (b) show the 'shock' to the bottom-right of the circle, before and after adaption; (c) and (d) likewise show the top 'shock' before and after adaption.



(a) (b)

Figure 8: Adaption of an unstructured triangular mesh around a cylinder to pre-specified shock locations, using the marching method. The original mesh is shown in (a), and the final mesh in (b).



(a) (b)

(c) (d)

Figure 9: Close-up view of the marching method of mesh adaptation. (a) and (b) show the 'shock' to the bottom-right of the circle, before and after adaption; (c) and (d) likewise show the top 'shock' before and after adaption.

V. Summary and Future Work

Shock capturing in moving meshes as well as real-time mesh refinement is a much needed feature for any CFD solver. In this paper different mesh-motion algorithms as well as methods to incorporate such motion seamlessly into a high-order code were investigated. The effectiveness of these algorithms to 1D problems with multiple discontinuities and some preliminary 2D results are shown. The pros and cons of the different approaches in their extension to 2D and 3D unstructured meshes were discussed. The optimization method works well in 1D but has not yet been used in higher dimensions effectively and is a focus of our future work. The spring-attractor model where we use force-equilibrium to solve for the adapted mesh has been tested in 1D and shows good promise. In 2D and 3D, the linear-elasticity method shows great promise as a unified framework for both adaptation and boundary motion. While adopting ALE formulation for general rigid mesh motion has been successful in 2D, using ALE in combination with these adaptation algorithms has been difficult probably due to high mesh velocities necessary to adapt to moving shocks, as well as difficulties in ensuring geometric conservation when element volumes change drastically.

Acknowledgments

The authors would like to acknowledge the support for this work provided by the Stanford Graduate Fellowship program, National Science Foundation (grant number 1114816) and Air Force Office of Scientific Research (grant number FA9550-10-1-0418).

References

[1]Manuel R. López-Morales, Abhishek Sheshadri, K. A. J. C. T. E. D. M. J. R. J. W. D. W. D. F. P. P. A. J., "Verification and Validation of HiFiLES: a High-Order LES unstructured solver on multi-GPU platforms," 2014.

[2]Sheshadri, A. and Jameson, A., "Shock detection and capturing methods in high order Discontinuous Galerkin Finite Element Methods," 2014.

[3]Persson, P., "Shock Capturing for High-Order Discontinuous Galerkin Simulation of Transient Flow Problems," 21st aiaa computational fluid dynamics conference proceedings 2013-3061, 2013.

[4]A. Klöckner, T. W. b. and Hesthaven, J. S., "Viscous Shock Capturing in a Time-Explicit Discontinuous Galerkin Method," Tech. rep., 2010.

[5]Persson, P. and Peraire, J., "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods," Tech. rep., 2006.

[6]F. Palacios, M. R. Colonno, A. C. A. A. C. S. R. C. T. D. E. A. K. L. T. W. L. T. W. R. T. and Alonso, J. J., "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design," 2013.

[7]Dwight, R., "Robust Mesh Deformation using the Linear Elasticity Equations," *Computational Fluid Dynamics 2006*, Springer Berlin Heidelberg, 2009, pp. 401–406.

[8]Ou, K., "Dynamic Mesh Deformation for Adaptive Grid Refinement with Staggered Spectral Difference and Finite Volume Mesh," 2011.

[9]Ou, K., *High Order Methods for Unsteady Flows on Unstructured Dynamic Meshes*, Ph.D. thesis, Stanford University, 2012.

[10]Persson, P., "Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains," *Computer Methods in Applied Mechanics and Engineering*, 2009.

[11]Sod, G. A., "A survey of several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws," *Journal of Computational Physics*, 1978.