Stabilization of High-Order Methods for Unstructured Grids with Local Fourier Spectral Filtering: high-Re Simulations in Coarse Meshes

Manuel R. López-Morales * and Antony Jameson†
Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305

One of the main barriers to wide adoption of high-order numerical methods in industrial applications is the schemes' low robustness relative to low-order methods. HiFiLES, an open-source, high-order, Navier-Stokes solver for unstructured grids is not impervious to this problem. Its stability is generally highly dependent on the quality of the grid. This paper describes the implementation of the Local Fourier-spectral (LFS) filters, developed by Asthana and the authors, in HiFiLES, and shows the results of high-Re simulations in coarse, unstructured 2D meshes. The simulations demonstrate the potential for LFS filters to stabilize high-order simulations under extreme conditions: very coarse grids, high-Re, high-Ma, and very low-Ma. We present a formulation of the LFS filters for a general high-order polygon (2D) or polyhedron (3D).

LFS filters are uniquely suited to implementation in highly-parallelizable numerical schemes like Flux Reconstruction (FR) because they operate element-wise, use interface information that is already used by the element to advance the solution, and maintain the operational complexity of the underlying scheme –the filtering operation is two element-local matrix multiplications.

I. Introduction

Low-order methods are ubiquitous in industry and academia. Regardless of the mesh quality and flow conditions, commercial CFD packages output an answer. It is up to the informed user to decide if such answer is believable or accurate to her satisfaction. This is not so true of high-order methods: they are still sensitive to starting conditions, mesh quality, and the non-linearity of the flow, i.e. how high the Reynolds number (Re) is. If any of these parameters is not chosen well, the simulation will halt prematurely and no result, not even a rough estimate, will be provided. Certainly, this is not acceptable in an industrial setting.

The Local Fourier Spectral (LFS) filters being proposed in this paper are an attempt at tackling the low robustness of high-order methods from the perspective of flow physics, rather than the classical frameworks of polynomial order reduction, artificial viscosity, or limiting. Very promising results have been shown by Asthana and the authors [1] for 1D non-linear advection-diffusion problems and 2D inviscid high-Ma flows with quadrilateral, unstructured, coarse grids, and polynomial discretizations of up to order 8. The present paper extends their formulation to arbitrary elements in arbitrary dimensions and shows results in high-Re flows without turbulence modeling.

It is well known that turbulent flows –which tend to be high-Re–, exhibit an energy cascade: the energy from large scales is transferred to smaller scales due to natural dissipation. In very crude terms, large vortices become ever smaller vortices until they reach dimensions proportional to the Kolmogorov length-scale [2]. This phenomenon is well captured by the Navier-Stokes equations. Hence, a good Navier-Stokes solver would see large scales become ever smaller. In general, low-order methods in grids not created for Direct Numerical Simulation (DNS) introduce enough numerical dissipation that the ever shrinking scales are dissipated before they become aliased (or under-sampled).

We speculate that it is precisely this very natural energy cascade which is de-stabilizing high-order numerical methods. Because high-order methods introduce little numerical dissipation, the ever shrinking scales may not be dissipated before they become aliased: they re-appear as larger scales that, naturally, become smaller later on. This

*Ph.D. Candidate, Department of Aeronautics and Astronautics, Stanford University, AIAA Student Member; mlopez14@stanford.edu
†Thomas V. Jones Professor of Engineering, Department of Aeronautics and Astronautics, Stanford University, AIAA Fellow

vicious cycle introduces non-physical energy into the flow until the simulation is de-stabilized. Thus, removing the small scales before they become aliased would, in theory, stabilize the solution.

Local Fourier Spectral filters target scales relative to the element size, as the filtering operation happens in the reference element. The smaller the element, the smaller the scale being filtered. In addition, the filters help satisfy boundary conditions. The results presented here show that the LFS filters can stabilize not only high-Re flows but also moderate-Ma and low-Ma flows in coarse grids, which opens the door to using the filters as pre-conditioners or in multigrid cycles. All simulations being presented ran from start to finish without intervention.

Many stabilization schemes created for high-order methods have focused on shock-capturing. A concise review of the shock-capturing literature can be seen in Section 3.4 in[3]. An artificial viscosity-based shock capturing approach that has gained popularity because of its ease of implementation and increase of robustness was suggested by Persson and Peraire[4]. A similar approach for high-Re flows with turbulence modeling has been proposed by Nguyen et al.[5]. Lodato[6] has used filtering in the formulation of Sub-Grid Scales (SGS) models for Large Eddy Simulations (LES) with high-order Spectral Difference (SD) schemes. His work inspired the formulation of the filters presented here. A stabilization strategy based on optimization was suggested by Guba et al.[7] shows great promise. A limiter-based stabilization stragey easily implemented in Discontinuous-Galerkin-type methods was proposed by Kuzmin[8].

The main reason we decided to find a stabilization strategy that could be posed as a matrix multiplcation and requires a very local stencil arises from the fact that HiFiLES performs best on Graphical Processing Units (GPUs). GPUs require a low-communication, highly-parallel implementation with organized memory accesses and homogeneous computations. As limiting procedures at an element require querying neighboring elements, they require disorganized memory accesses. Performing element-wise optimizations requires heterogeneous computations (each kernel performs a different computation). Because HiFiLES is using explicit time-stepping, adding artificial dissipation explicitly would make the time-step limits even more stringent.

Section II. presents a general description of the Flux Reconstruction (FR) method to show how a filter that costs two small matrix multiplications per element is ideal for this scheme. Section III. describes the properties being sought in the LSF filters and the mechanics of their implementation. Section IV. provides visualizations of the filters and their effects on a polynomial solution. Section V. presents the results of 2D simulations, in unstructured coarse grids, of flows where $\mathrm{Re} = 1e6$. This Reynolds number was selected based on the availability of consistent experimental data for the case of the cicular cylinder.

II. Flux Reconstruction Method

What follows is a brief overview of the flux reconstruction (FR) framework with an emphasis on how the LFS filtering technique can be seamlessly implemented. We start the discussion with the solution of the advection equation in one dimension using the FR approach to illustrate the method. We then proceed to briefly explain how conservation equations can be solved in multiple dimensions. The Navier-Stokes equations are a set of coupled conservation equations in multiple dimensions, so the extension of the FR methodology to them is straightforward. The detailed description of the algorithm used in HiFiLES is given by Castonguay et al.[9].

A. Solution of the Advection Equation in One Dimension using the FR Approach

Consider the one-dimensional conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \tag{1}$$

in domain Ω, where x is the spatial coordinate, t is time, u –the *solution*– is a scalar function of x and t, and f –the *flux*– is a scalar function of u. Note that by letting $f = f(u, \frac{\partial u}{\partial x})$, Equation (1) becomes a model of the Navier-Stokes equations.

Let us partition the domain $\Omega = [x_1, x_{N+1})$ into N non-overlapping elements with interfaces at $x_1 < x_2 < ... < x_{N+1}$. Then,

$$\Omega = \bigcup_{n=1}^{N} \Omega_n \tag{2}$$

and $\Omega_n = [x_n, x_{n+1})$ for $n = 1, ..., N$. To simplify the implementation, let us map each of the physical elements Ω_n

to a standard element $\Omega_s = [-1, 1)$ with the function $\Theta_n(\xi)$, where

$$x = \Theta_n(\xi) = \left(\frac{1-\xi}{2}\right) x_n + \left(\frac{1+\xi}{2}\right) x_{n+1} \tag{3}$$

With this mapping, the evolution of u within each Ω_n can be determined with the following transformed conservation equation

$$\frac{\partial \hat{u}}{\partial t} + \frac{1}{J_n}\frac{\partial \hat{f}}{\partial \xi} = 0 \tag{4}$$

where

$$\hat{u} = u(\Theta_n(\xi), t) \text{ in } \Omega_n \tag{5}$$

$$\hat{f} = f(\Theta_n(\xi), t) \text{ in } \Omega_n \tag{6}$$

$$J_n = \left.\frac{\partial x}{\partial \xi}\right|_{\Omega_n} \tag{7}$$

Now, we introduce polynomials of degree p, $\hat{u}^\delta$ and $\hat{f}^\delta$, to approximate the exact values $\hat{u}, \hat{f}$, respectively. We can write these polynomials as

$$\hat{u}^\delta = \sum_{i=1}^{N_s} \hat{u}_i^\delta l_i(\xi) \tag{8}$$

$$\hat{f}^\delta = \sum_{i=1}^{N_s} \hat{f}_i^\delta l_i(\xi) \tag{9}$$

where N_s is the number of solution points, $\hat{u}_i^\delta$ is the current value of the solution approximation function at the i^{th} *solution point* (or *internal point*) in the reference element, $\hat{f}_i^\delta$ is the current value of the flux approximation function at the i^{th} *flux point* (or *boundary point*) in the reference element, l_i is the Lagrange polynomial equal to 1 at the i^{th} solution point and 0 at the others, and δ denotes that the function is an approximation.

Note that the piecewise polynomials might not be continuous (or C^0) across the interfaces. In the Flux Reconstruction approach, the flux used in the time advancement of the solution is made C^0 by introducing flux correction functions.

This can be achieved by finding interface solution values at each element boundary and then correcting the solution. Let $\hat{f}_L^{\delta I}$ and $\hat{f}_R^{\delta I}$ be the interface flux values at left and right boundaries of some element, respectively. $\hat{f}_L^{\delta I}$ and $\hat{f}_R^{\delta I}$ can be found with a Riemann solver for Discontinuous-Galerkin (DG) methods [10]. Then, select solution correction functions g_L and g_R such that

$$g_L(-1) = 1 \ , \ g_L(1) = 0 \tag{10}$$

$$g_R(-1) = 0 \ , \ g_R(1) = 1 \tag{11}$$

and let

$$\hat{f}^C = \hat{f}^\delta + (\hat{f}_L^{\delta I} - \hat{f}_L^\delta) g_L + (\hat{f}_R^{\delta I} - \hat{f}_R^\delta) g_R \tag{12}$$

where superscript C denotes the function is corrected, and $\hat{f}_L^\delta$, $\hat{f}_R^\delta$ represent the flux approximation evaluated at the left and right boundaries.

The solution can then be advanced at each solution point i in element n. In semi-discrete form, this is

$$\frac{d\hat{u}_i^\delta}{dt} = -\frac{1}{J_n}\frac{\partial \hat{f}^C}{\partial \xi}(\xi_i) \tag{13}$$

The FR scheme can be made provably stable for the linear advection-diffusion equation by selecting special types of correction functions [11]. In general, these correction functions are polynomials of degree $p + 1$ so both sides in Equation (13) are quantities related to polynomials of order p –for consistency [12].

Vincent et al. [13] have shown that in the case of the 1-dimensional, linear advection equation, the Flux Reconstruction approach can be proven to be stable for a specific family of correction functions parameterized by a scalar called c. In addition, they showed that by selecting specific values of c it is possible to recover a particular nodal Discontinuous Galerkin (DG) and Spectral Difference (SD) methods plus a FR scheme that was previously found to be stable by Huynh [14].

B. Extension to Multiple Dimensions

Extension of FR to multiple dimensions requires formulating multi-dimensional interpolation functions and correction functions that satisfy boundary conditions equivalent to those in Equation (10) for each type of element.

Interpolation bases for quadrilaterals and hexahedra can be obtained via tensor products of the 1-dimensional interpolation basis. In HiFiLES, the solution in hexahedra is discretized in the following way

$$\hat{u}^{\delta}(\xi,\eta,\zeta) = \sum_{i=1}^{p+1}\sum_{j=1}^{p+1}\sum_{k=1}^{p+1} \hat{u}^{\delta}_{i,j,k} l_i(\xi) l_j(\eta) l_k(\zeta) \tag{14}$$

where i, j, k index the solution points along the ξ, η, ζ directions, respectively. The flux is discretized similarly.

The interpolation basis for triangles and tetrahedra are described in detail by Hesthaven and Warburton [10]. Figure 3 shows a possible configuration of internal and boundary points. The extension of interpolation polynomials to prisms is obtained via tensor products of the 1-dimensional basis with the triangular basis [9].

The most general polynomial discretization of a k-dimensional solution scalar field and flux vector field in an arbitrary reference element is

$$\hat{u}^{\delta}(\boldsymbol{\xi}) = \sum_{i=1}^{N_s} \hat{u}^{\delta}_i \phi_i(\boldsymbol{\xi}) \tag{15}$$

$$\hat{\mathbf{f}}^{\delta}(\boldsymbol{\xi}) = \sum_{i=1}^{N_s} \hat{\mathbf{f}}^{\delta}_i \phi_i(\boldsymbol{\xi}) \tag{16}$$

where N_s is the number of solution points in an element, $\phi_i(\boldsymbol{\xi})$ is a polynomial basis function associated with solution point i constructed such that $\phi_i(\boldsymbol{\xi}_j) = \delta_{ij}$ and $i, j = 1, \ldots, N_s$.

By letting $\vec{u} =< \hat{u}^{\delta}_1, \hat{u}^{\delta}_2, \ldots, \hat{u}^{\delta}_{N_s} >^T$, $\vec{\mathbf{f}} =< \hat{\mathbf{f}}^{\delta}_1, \hat{\mathbf{f}}^{\delta}_2, \ldots, \hat{\mathbf{f}}^{\delta}_{N_s} >^T$, and $\vec{\phi} =< \phi_1, \phi_2, \ldots, \phi_{N_s} >^T$ the discretization can be written more concisely as

$$\begin{aligned} \hat{u}^{\delta}(\boldsymbol{\xi}) &= \vec{u}^T \cdot \vec{\phi}(\boldsymbol{\xi}) = \vec{\phi}(\boldsymbol{\xi})^T \cdot \vec{u} \\ \hat{\mathbf{f}}^{\delta}(\boldsymbol{\xi}) &= \vec{\mathbf{f}}^T \cdot \vec{\phi}(\boldsymbol{\xi}) = \vec{\phi}(\boldsymbol{\xi})^T \cdot \vec{\mathbf{f}} \end{aligned} \tag{17}$$

Note that we are using the boldface and arrow notation to denote vectors. Boldface vectors have a number of entries equal to the number of dimensions of the problem domain. Arrow vectors are general vectors.

In the general FR approach, the boundary conditions for the correction functions in multiple dimensions can be formulated as

$$\mathbf{h}_i(\boldsymbol{\xi}_j) \cdot \mathbf{n}_j = \delta_{ij} \tag{18}$$

where $\mathbf{h}_i$ is the correction vector function associated with interface point i, $\boldsymbol{\xi}_j$ is the location vector of the j^{th} interface point, and $\mathbf{n}_j$ is the outward unit normal at interface point j. Interface points are located on the boundary of an element.

One of the challenges in the FR approach is finding correction functions that not only satisfy Equation (18) but also guarantee stability in the linear advection-diffusion case. Correction functions that guarantee such stability exist for 1-dimensional segments [13], triangles [15,16], and tetrahedra [17]. FR schemes with these correction functions comprise the ESFR family of schemes.

The update step at solution point i and element n in the FR approach for the multidimensional advection equation $\frac{\partial u}{\partial t} + \nabla \cdot \vec{f} = 0$ becomes

$$\begin{aligned} \frac{du^{\delta}_i}{dt} &= -\frac{1}{\det(\tilde{J}_n)} \nabla \cdot \mathbf{f}^C_i(\boldsymbol{\xi}_i) = -\frac{1}{\det(\tilde{J}_n)} \nabla \cdot \left(\mathbf{f}^{\delta}_i(\boldsymbol{\xi}_i) + \sum_{j=1}^{N_f} \left[\left(\hat{\mathbf{f}}^{\delta I}_j - \hat{\mathbf{f}}^{b}_j \right) \cdot \mathbf{n}_j \right] \vec{\mathbf{h}}_j(\boldsymbol{\xi}_i) \right) \\ &= -\frac{1}{\det(\tilde{J}_n)} \left(\nabla \cdot \mathbf{f}^{\delta}_i(\boldsymbol{\xi}_i) + \sum_{j=1}^{N_f} \left[\left(\hat{\mathbf{f}}^{\delta I}_j - \hat{\mathbf{f}}^{b}_j \right) \cdot \mathbf{n}_j \right] \nabla \cdot \vec{\mathbf{h}}_j(\boldsymbol{\xi}_i) \right) \\ &= -\frac{1}{\det(\tilde{J}_n)} \left(\vec{\nabla}\phi(\boldsymbol{\xi}_i)^T \cdot \vec{\mathbf{f}} + \sum_{j=1}^{N_f} \left[\left(\hat{\mathbf{f}}^{\delta I}_j - \hat{\mathbf{f}}^{b}_j \right) \cdot \mathbf{n}_j \right] \nabla \cdot \vec{\mathbf{h}}_j(\boldsymbol{\xi}_i) \right) \end{aligned} \tag{19}$$

where N_f is the number of interface points, $\tilde{J}_n$ is the Jacobian matrix, $\vec{\nabla}\phi(\boldsymbol{\xi}_i)$ is the vector of gradients of each ϕ_i function evaluated at $\boldsymbol{\xi}_i$, $\vec{\mathbf{f}}^{\delta}$ is a vector of flux vectors, and $\hat{\mathbf{f}}_j^b$ is the flux vector at the j^{th} interface point (obtained via extrapolation).

Note that it is possible to evaluate each of the terms in Equation (19) for all $i = 1, \ldots, N_s$ with a series of matrix-vector multiplications.

III. Local Fourier Spectral Filters

A. Desired Properties

When designing the Local Fourier-spectral Filters presented in this paper, we considered the following properties as desirable:

1. The filter should have spectral interpretation so there is control over the physical scales being smoothed
2. The filtering operation must have a local stencil: only interior and boundary solution values can be used to filter the solution in the interior of the element
3. The filter should preserve boundary conditions
4. Filtering the solution at the interior points should be influenced by the solution values at the element's boundary
5. The influence of a boundary point on the internal point being filtered should be inversely proportional to the distance between them
6. To limit computational cost, the filter should be applied in the reference domain and involve matrix-vector multiplications exclusively
7. Filtering strategy should be generalizable to any type of element in unstructured grids

   In the early stages of the filter design, we noted that Properties 3 and 4 could be overly stringent, so we re-phrased them as the following, more relaxed condition:

8. If solution values at the boundary lie on a hyperplane in the space of spatial coordinates and solution values, the filtering operation should bring the internal solution values closer to such plane. In other words, if the solution values at the boundaries can be determined from a linear function of their location, the filter should bring the internal solution values closer to satisfying such linear function.

Condition 8 may not allow a filtered solution to satisfy the boundary conditions always. Nevertheless, in the limit of infinite mesh refinement the solutions at the boundary of every element will be coplanar, even in the presence of shocks in the Navier-Stokes equations. Hence, Condition 8 satisfies conditions 3 and 4 in a weak sense. This re-formulation was inspired by the Essentially-Local-Extremum-Diminishing (ELED) property of the Jameson-Schmidt-Turkel (JST) scheme[18] for finite volume methods. Whether or not (some) LFS filters satisfy ELED properties shall be left for a future study.

B. Mechanics of LFS Filters

The core idea of the LFS filtering technique is that the solution is filtered with a matrix whose entries depend on the element interface values, the basis functions of the solution, and a selected Fourier filtering kernel.

Suppose we wish to filter the scalar field v inside an arbitrary element. We can represent v, as usual, as a weighted sum of basis functions.

$$v(\boldsymbol{\xi}) = \sum_{i=1}^{N_s} v_i \phi_i(\boldsymbol{\xi}) \tag{20}$$

where v_i is the i^{th}weight, $\boldsymbol{\xi}$ is a vector of coordinates in a reference domain, and $\phi_i(\boldsymbol{\xi})$ is the i^{th}basis function.

The key component of the LFS technique is that the Fourier filtering operation over the element is evaluated exactly at each internal point i with coordinates $\boldsymbol{\xi}_i$. Suppose we wish to use filtering kernel $G(\boldsymbol{\xi})$ to filter $v(\boldsymbol{\xi})$, then the filtered

value at internal point i becomes

$$\begin{aligned}\bar{v}_j &= (G * v)(\boldsymbol{\xi}_j) \\ &= \int_{\mathbb{R}^d} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) v(\boldsymbol{\xi}) d\boldsymbol{\xi} \\ &= \int_{\mathbb{R}^d} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \left(\sum_{i=1}^{N_s} v_i \phi_i(\boldsymbol{\xi}) \right) d\boldsymbol{\xi} \\ &= \sum_{i=1}^{N_s} v_i \int_{\mathbb{R}^d} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi}\end{aligned} \tag{21}$$

where Ω is the entire problem domain, not just the element domain, N_s is the number of solution points, and the overbar means the quantity is filtered. Filtering over the entire domain would be computationally expensive, but would certainly smoothen the solution field and can be done while maintaining the order of accuracy of the underlying numerical scheme[19–21]. In order to use only a local (element-wise) stencil, it is possible to break the integration as follows

$$\int_{\mathbb{R}^d} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi} = \int_{\Omega_n} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi} + \int_{\mathbb{R}^d \setminus \Omega_n} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{22}$$

where Ω_n is the domain of element n (where the ϕ_i values are known) and $\Omega \setminus \Omega_n$ is the complement of Ω_n in Ω. As G and ϕ_i are defined in Ω_n, it is possible to calculate the first integral in Equation (22) analytically or via an adaptive quadrature algorithm like `quanc8`[22].

A design choice arises in defining ϕ_i in the domain $\Omega \setminus \Omega_n$. Asthana and the authors[1] initially decided to let ϕ_i equal a constant in 1D elements, and a combination of constant and polynomial in quadrilaterals. Non-tensor-product elements became problematic for this kind of formulation. This paper presents a more generic filter design method based on the desired properties in Section A..

Let us describe the mechanics of the two components of the integral in Equation (22) in a subsection each. We can call the filter associated with the term $\int_{\Omega_n} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi}$ "Internal Filtering Component" and the filter associated with the term $\int_{\mathbb{R}^d \setminus \Omega_n} G(\boldsymbol{\xi}_j - \boldsymbol{\xi}) \phi_i(\boldsymbol{\xi}) d\boldsymbol{\xi}$ "Boundary Filtering Component".

The filtering operation being sought would follow the following format

$$\vec{\bar{v}} = \alpha \mathcal{T} \vec{v} + (1 - \alpha) \mathcal{B} \vec{v^*} \tag{23}$$

where $\mathcal{T}$ is the internal filtering component, as it acts on the solution at internal points $\vec{v}$, and $\mathcal{B}$ is the boundary filtering component, as it acts on the solution at boundary points $\vec{v^*}$, and α is a scalar between 0 and 1 that determines the relative influence the internal and boundary components have on the internal solution values.

C. Internal Filtering Component

There are two steps to the creation of the internal filtering component $\mathcal{T}$. First we create the filtering matrix with the spectral interpretation, and then we normalize it so if the quantity being filtered is a constant, the filter preserves the constant.

1. Matrix formation

It can be seen from the last line in Equation (22) that finding the vector of solution values filtered in the reference domain can be posed as a matrix-vector multiplication:

$$\begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \vdots \\ \bar{v}_{N_s} \end{bmatrix} = \begin{bmatrix} \int_{\Omega_n} G(\boldsymbol{\xi}_1 - \boldsymbol{\xi}) \phi_1(\boldsymbol{\xi}) d\boldsymbol{\xi} & \int_{\Omega_n} G(\boldsymbol{\xi}_1 - \boldsymbol{\xi}) \phi_2(\boldsymbol{\xi}) d\boldsymbol{\xi} & \cdots & \int_{\Omega_n} G(\boldsymbol{\xi}_1 - \boldsymbol{\xi}) \phi_{N_s}(\boldsymbol{\xi}) d\boldsymbol{\xi} \\ \int_{\Omega_n} G(\boldsymbol{\xi}_2 - \boldsymbol{\xi}) \phi_1(\boldsymbol{\xi}) d\boldsymbol{\xi} & \int_{\Omega_n} G(\boldsymbol{\xi}_2 - \boldsymbol{\xi}) \phi_2(\boldsymbol{\xi}) d\boldsymbol{\xi} & \cdots & \int_{\Omega_n} G(\boldsymbol{\xi}_2 - \boldsymbol{\xi}) \phi_{N_s}(\boldsymbol{\xi}) d\boldsymbol{\xi} \\ \vdots & \vdots & \ddots & \vdots \\ \int_{\Omega_n} G(\boldsymbol{\xi}_{N_s} - \boldsymbol{\xi}) \phi_1(\boldsymbol{\xi}) d\boldsymbol{\xi} & \int_{\Omega_n} G(\boldsymbol{\xi}_{N_s} - \boldsymbol{\xi}) \phi_2(\boldsymbol{\xi}) d\boldsymbol{\xi} & \cdots & \int_{\Omega_n} G(\boldsymbol{\xi}_{N_s} - \boldsymbol{\xi}) \phi_{N_s}(\boldsymbol{\xi}) d\boldsymbol{\xi} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N_s} \end{bmatrix} \tag{24}$$

More compactly,

$$\vec{\bar{v}} = \mathcal{F} \vec{v} \tag{25}$$

we use over-arrow to denote column vectors. Interesting choices for $G(\boldsymbol{\xi})$ depend on a desired filter width h in the physical space:

I. Gaussian function:

$$G(\boldsymbol{\xi}) = \frac{1}{\det(\boldsymbol{\sigma})(2\pi)^{n/2}} e^{-\frac{1}{2}\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{\sigma}^{-\mathrm{T}}\boldsymbol{\sigma}^{-1}\boldsymbol{\xi}}$$

where the covariance matrix $\boldsymbol{\sigma}$ depends on h, and n is the number of dimensions of $\boldsymbol{\xi}$

II. Multidimensional indicator function:

$$G(\boldsymbol{\xi}) = \mathrm{I}_{[0,h]}(||\boldsymbol{\xi}||)$$

where $|\boldsymbol{\xi}|$ is a measure of the length of $|\boldsymbol{\xi}|$.

III. Multidimensional sharp-spectral function:

$$G(\boldsymbol{\xi}) = ||\boldsymbol{\xi}||^{-n/2} J_{n/2}(||\boldsymbol{\xi}||/h)$$

where J_α is a Bessel function of the first kind and n is the number of dimensions of $\boldsymbol{\xi}$.

In these functions, the larger the value of h, the wider the filter is in physical space, so the more wavenumbers it dampens. Note that if $h = 0$, $\mathcal{F}_{ij} = \phi_j(\boldsymbol{\xi}_i)$, so $\vec{\bar{v}} = \vec{v}$. The integrals in Equation (24) can be evaluated to arbitrary accuracy with `quanc8` [22] in a pre-processing stage.

2. *Enforcing Conservation of a Constant Quantity*

The requirement to preserve a constant can be posed in an equation as

$$\bar{v}_i = \sum_{j=1}^{N_s} \mathcal{F}_{ij} v_j = v_i \tag{26}$$

where now $v_j = v_k$ for $j, k = 1, \ldots, N_s$. As a result, each row of $\mathcal{F}$ must satisfy

$$\sum_{j=1}^{N_s} \mathcal{F}_{ij} = 1 \tag{27}$$

where $i = 1, \ldots, N_s$. This constraint can be enforced by dividing each row of $\mathcal{F}$ with the total sum of the row. More explicitly,

$$\mathcal{T}_{ij} = \frac{\mathcal{F}_{ij}}{\sum_{k=1}^{N_s} \mathcal{F}_{ik}} \tag{28}$$

where the normalized matrix $\mathcal{T}$ is the internal filtering component.

3. *Integration Domain*

The normalization of the matrix introduces a bias. As the integration is being performed in some reference domain Ω_n, the integral $\int_{\Omega_n} G(\boldsymbol{\xi}_i - \boldsymbol{\xi})\phi_j(\boldsymbol{\xi}) d\boldsymbol{\xi}$ will tend to have a greater value when $\boldsymbol{\xi}_i$ is well inside the reference element. This is more evident when using the multidimensional indicator function (shown in II) as the filtering kernel.

To ameliorate this bias, we have selected to perform the integration in a reference domain that is symmetric. With this precaution all integrals related to points close to a domain edge have a similar magnitude, and thus the filtering operation does not favor points close to some arbitrary edge over the others.

Figure 1 illustrates the potential introduction of bias in triangular elements that use a right triangle as a reference element.

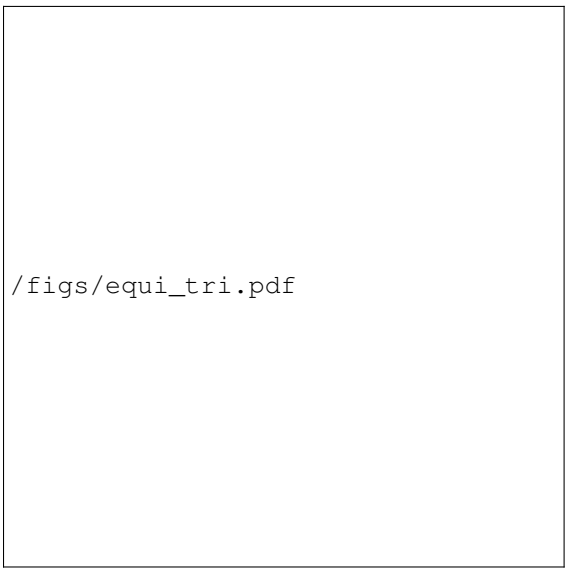
In practice, the basis functions $\phi_j(\boldsymbol{\xi})$ may be defined in non-symmetrical elements. To simplify the evaluation of the integral, the definition of the norm in the filtering kernel $G(\boldsymbol{\xi})$ can be modified so the curve drawn by $|\boldsymbol{\xi} - \boldsymbol{\xi}_0| = 1$ in the non-symmetric reference element would map to a circle in a symmetric reference element.

In the case of a triangle, the basis functions are already defined in a right triangle with vertices $[-1, -1], [1, -1], [-1, 1]$.

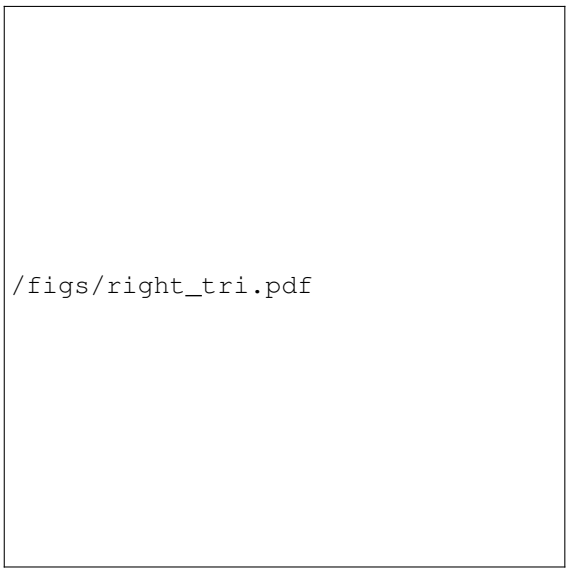
By defining the norm in this right triangle as follows

$$||\boldsymbol{\xi}||_{rt}^2 = \boldsymbol{\xi}_1^2 + \boldsymbol{\xi}_2^2 + \boldsymbol{\xi}_1\boldsymbol{\xi}_2 \tag{29}$$

$||\boldsymbol{\xi} - \boldsymbol{\xi}_0||_{rt} = 1$ maps to a circle in the equilateral triangle with vertices $[-1, -1], [1, -1], [0, \sqrt{3} - 1]$. Figure 2 sketches this mapping. $||\cdot||_{rt}$ is the norm defined in the right triangle.



(a) Equilateral triangle domain



(b) Right triangle domain

Figure 1. Using the same filter width in two different domains introduces different bias. The small, solid circles represent the location of solution points. The large circles represent the filter width acting on two specific internal points in two different domains. Integration is being performed in the area encompassed by the solid, straight lines.

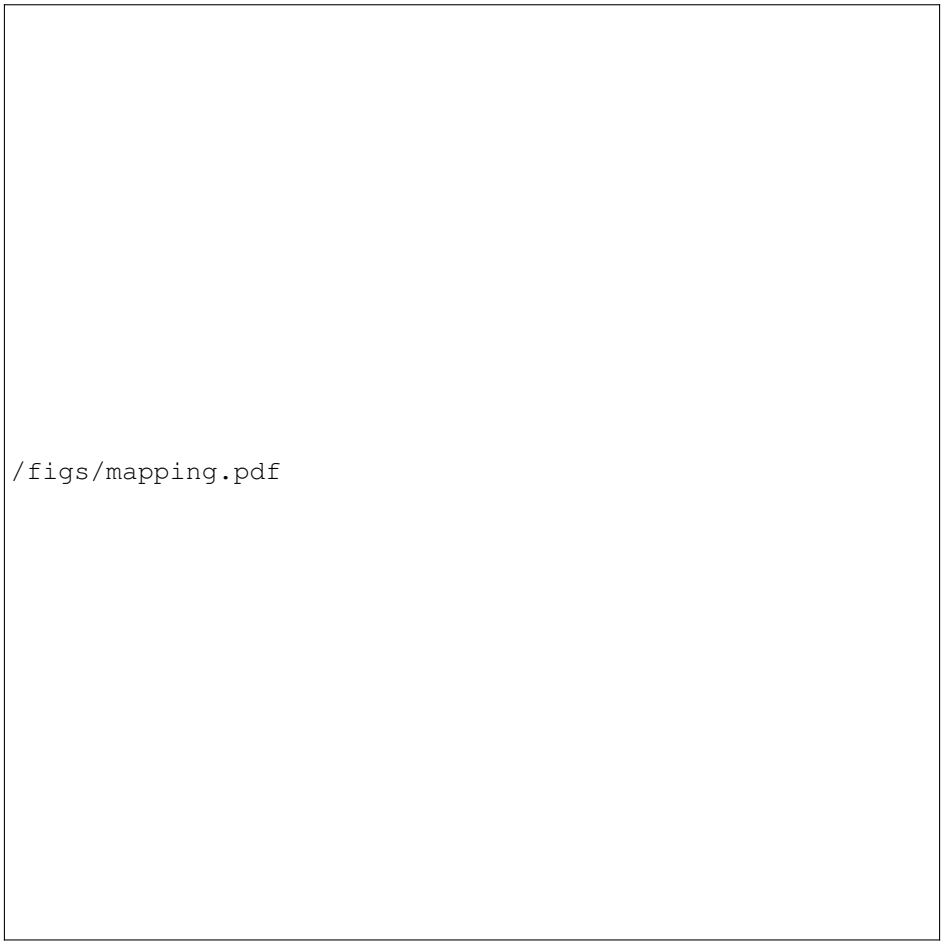


Figure 2. Sketch of ellipsis $|\boldsymbol{\xi} - \boldsymbol{\xi}_0|_{rt} = 1$ defined in the right triangle domain maps to a circle in the equilateral triangle domain

D. Boundary Filtering Component

Formulating the influence of boundary solution values on internal solution values as a matrix is not so straightforward. The approach being suggested here is based on some reasonable desired properties, but some choices are arbitrary.

The creation of this matrix proceeds as follows: place the boundary points and the internal points in a symmetric domain, select an "influence" function with as many unknowns as dimensions, find values for these unknowns such that Condition 8 is satisfied, and create the boundary filtering component.

1. Ensuring Boundary Filtering Component Preserves Linearity of the Solution at the Boundary (Condition 8)

In this sub-section, we seek to form matrix $\mathcal{B}$ in Equation 23. In this analysis, we assume that $\alpha = 0$ in Equation 23 so we can isolate $\mathcal{B}$'s properties from those of $\mathcal{T}$.

The preservation of linearity can be posed in the following way. Assume that values $\vec{v^*}$ at the boundaries are linear in space. More explicitly, for each boundary point $i = 1, \ldots, N_f$

$$v_i^* = \boldsymbol{\xi}_i^{*\mathrm{T}} \vec{a} + b \tag{30}$$

where v_i^* is the solution at the i^{th}boundary point, $\boldsymbol{\xi}_i^*$ are the coordinates of such boundary point, $\vec{a}$ is a vector of constant coefficients, and b is a constant scalar. In vector notation, we are assuming

$$\vec{v^*} = \underline{\boldsymbol{\xi}}^{*\mathrm{T}} \vec{a} + \vec{b^*} \tag{31}$$

where $\vec{b^*} = b\mathbb{1}_{[N_f \mathrm{x} 1]}$ and $\underline{\boldsymbol{\xi}}^{*\mathrm{T}}$ is a matrix of dimensions N_f by N_d –number of dimensions.

To satisfy Condition 8, we seek a matrix $\mathcal{B}$ such that, when $\vec{v^*}$ satisfies Equation 31,

$$\vec{v} = \mathcal{B}\vec{v^*} = \underline{\boldsymbol{\xi}}^{\mathrm{T}} \vec{a} + \vec{b} \tag{32}$$

where $\vec{b} = b\mathbb{1}_{[N_s \mathrm{x} 1]}$ and $\underline{\boldsymbol{\xi}}^{\mathrm{T}}$ is a matrix of dimensions N_s by N_d. The i^{th}row of $\underline{\boldsymbol{\xi}}^{\mathrm{T}}$ is $\boldsymbol{\xi}_i{}^{\mathrm{T}}$, the coordinates of the i^{th}internal point.

Therefore, we seek a matrix $\mathcal{B}$ such that

$$\mathcal{B}\left(\underline{\boldsymbol{\xi}}^{*\mathrm{T}} \vec{a} + \vec{b^*}\right) = \underline{\boldsymbol{\xi}}^{\mathrm{T}} \vec{a} + \vec{b} \tag{33}$$

As a result, for general $\underline{\boldsymbol{\xi}}^{*\mathrm{T}}$, $\underline{\boldsymbol{\xi}}^{\mathrm{T}}$, and $\vec{a}$,

$$\mathcal{B}\underline{\boldsymbol{\xi}}^{*\mathrm{T}} = \underline{\boldsymbol{\xi}}^{\mathrm{T}} \qquad \text{and} \qquad \mathcal{B}\vec{b^*} = \vec{b} \tag{34}$$

$$\implies \qquad \sum_{j=1}^{N_f} \mathcal{B}_{ij}\xi_{jd}^* = \xi_{id} \qquad \text{and} \qquad \sum_{j=1}^{N_f} \mathcal{B}_{ij} = 1 \tag{35}$$

for all $i = 1, \ldots, N_s$ and $d = 1, \ldots, N_d$. Where ξ_{jd}^* is the d^{th} coordinate of the j^{th} boundary point, and ξ_{id} is the d^{th} coordinate of the i^{th} boundary point.

Equation 35 introduces $N_s(N_d + 1)$ constraints for $N_s N_f$ unkonwns in $\mathcal{B}$. Each row in $\mathcal{B}$ needs to satisfy $N_d + 1$ equations. Table 1 shows the number of equations and unknowns for specific elements assuming that the solution whithin each is represented by a polynomial of degree $p \geq 0$.

All elements, except for the line element, have matrices $\mathcal{B}$ with more unknowns than equations. This calls for a reduction of the number of unknowns in a strategic way. This is achieved by invoking Condition 5: the farther away a boundary point is from an internal point, the less influential it should be.

2. Ensuring Influence of Boundary Points on Internal Points is Inversely Proportional to their Distance (Condition 5)

Given this requirement and Equation (35), a reasonable design choice for each entry in $\mathcal{B}$ is:

$$\mathcal{B}_{ij} = \frac{g\left(||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j^*||\right)^{-1}}{\sum_{k=1}^{N_f} g\left(||\boldsymbol{\xi}_i - \boldsymbol{\xi}_k^*||\right)^{-1}} \tag{36}$$

Element type	N_s	N_d	N_f	# of Equations: $N_s(N_d+1)$	# of Unkowns: N_sN_f
Line	$p+1$	1	2	$2(p+1)$	$2(p+1)$
Triangle	$\frac{1}{2}(p+1)\cdot(p+2)$	2	$3(p+1)$	$\frac{3}{2}(p+1)\cdot(p+2)$	$\frac{3}{2}(p+1)^2\cdot(p+2)$
Quadrilateral	$(p+1)^2$	2	$4(p+1)$	$3(p+1)^2$	$4(p+1)^3$
Tetrahedron	$\frac{1}{6}(p+1)\cdot(p+2)$ $\cdot(p+3)$	3	$4\frac{1}{2}(p+1)\cdot(p+2)$	$\frac{2}{3}(p+1)\cdot(p+2)$ $\cdot(p+3)$	$\frac{1}{3}(p+1)^2\cdot(p+2)^2$ $\cdot(p+3)$
Pyramid	$\frac{1}{6}(p+1)\cdot(p+2)$ $\cdot(2p+3)$	3	$4\frac{1}{2}(p+1)\cdot(p+2)$ $+(p+1)^2$	$\frac{2}{3}(p+1)\cdot(p+2)$ $\cdot(2p+3)$	$\frac{1}{6}(p+1)^2\cdot(p+2)$ $\cdot(2p+3)\cdot(3p+5)$
Prism	$\frac{1}{2}(p+1)^2\cdot(p+2)$	3	$2\frac{1}{2}(p+1)\cdot(p+2)$ $+3(p+1)^2$	$2(p+1)^2\cdot(p+2)$	$\frac{1}{2}(p+1)^3\cdot(p+2)$ $\cdot(4p+5)$
Hexahedron	$(p+1)^3$	3	$6(p+1)^2$	$4(p+1)^3$	$6(p+1)^5$

Table 1. Number of internal points N_s, number of boundary points N_f, and number of equations and unknowns in matrix $\mathcal{B}$ for each type of element, assuming the solution is being discretized with a polynomial of degree p

where $\boldsymbol{\xi}_i$ is the location of the i^{th}internal point, $\boldsymbol{\xi}_j^*$ is the location of the j^{th}boundary point, and $g(\cdot)$ is a monotonically increasing function and $g(0) = 0$. The norm $||\cdot||$ is ideally defined in a symmetric element, or a reformulation of a norm in an non-symmetric element as in Equation (29). We note that the requirement that $\sum_{j=1}^{N_f}\mathcal{B}_{ij} = 1$ is immediately satisfied regardless of the exact form of $g(\cdot)$.

To avoid dividing by zero when $||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j^*|| = 0$, we can recast Equation (36) as

$$\mathcal{B}_{ij} = \frac{1}{1 + g\left(||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j^*||\right)\sum\limits_{\substack{k=1\\k\neq j}}^{N_f} g\left(||\boldsymbol{\xi}_i - \boldsymbol{\xi}_k^*||\right)^{-1}} \tag{37}$$

This definition of $\mathcal{B}$ is now too stringent, so the conditions in Equation (35) are not clearly satisfied for any selection of $g(\cdot)$. Each row in $\mathcal{B}$ has N_d equations left to satisfy.

By introducing N_d unknowns in the definition of $g(\cdot)$ per row we can expect to satisfy the remaining N_d equations per row. A choice made here is to let

$$g(x) = \sum_{d=1}^{N_d} a_{id}x^d \tag{38}$$

where x is a scalar, and a_{id} are the unknowns to be found in each row i.

To find the values of a_{id}, a regular non-linear minimization function can be invoked. In this paper, we used the downhill simplex method by Nelder and Mead[23]. The implementation in C++ was taken from Jia[24]. The function being minimized for each row i of $\mathcal{B}$ is

$$J(a_{i1},\ldots,a_{1N_d}) = \sum_{d=1}^{N_d}\left(\sum_{j=1}^{N_f}\mathcal{B}_{ij}\xi_{jd}^* - \xi_{id}\right)^2 \tag{39}$$

where J is the cost function to be minimized. In occasions, the minimum of J does not reach machine zero. However, as it will be shown in the case of triangles, $\mathcal{B}$ still behaves as desired.

IV. Visualization of the LFS Filters in Triangular Elements

In this section we present visualizations of the effect of the LFS filters on a polynomial solution of degree 5 in a reference triangle.

In the plots that follow, the internal and boundary points are located as shown in Figure 3.

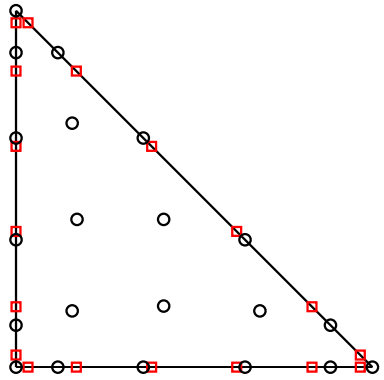


Figure 3. Internal and boundary points in the reference triangular domain when solution is represented with polynomial of degree 5 **(**$p = 5$**). Black circles represent the internal points. Red squares represent the boundary points.**

A. Internal Filtering Components

To illustrate the effect of the internal filtering component, let us filter a solution using matrix $\mathcal{T}$ exclusively. Figure 4 shows the result of filtering using a width of $h = 10$ and the 2-D sharp-spectral filtering kernel (III) using the modified norm (29).

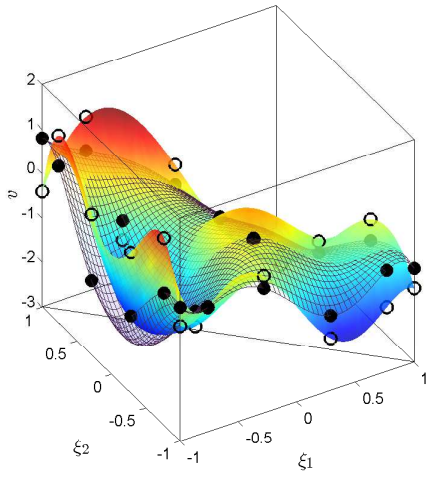


Figure 4. Solution values after filtering with internal component exclusively, $\vec{v} = \mathcal{T}\vec{v}$**, where** $v_i = \sin(k\xi_{i1}) + \cos(k\xi_{i2})$**, where** $k = 500$**. Hollow black circles show the unfiltered solution at the interior points, transparent colored surface is the polynomial representation of the unfiltered solution, filled black circles show the filtered values of the solution at the interior points, and the meshed surface shows the polynomial representation of the filtered solution.**

It can be seen that the polynomial representation of the filtered values is smoother than the polynomial representation of the unfiltered values while maintaining the general shape and curvature.

B. Boundary Filtering Components

To illustrate the effect of the boundary filtering component, let us filter a solution using matrix $\mathcal{B}$ exclusively. Figure 5 shows the result of filtering. Because only $\mathcal{B}$ is acting on the solution, the unfiltered values of the solution at the internal points are not plotted.

We have made all the values of the solution at the boundaries co-planar to illustrate how effectively Condition 8 is satisfied. It is clear the operation of filtering with $\mathcal{B}$ does bring the filtered solution closer to a plane if the boundary values are co-planar.

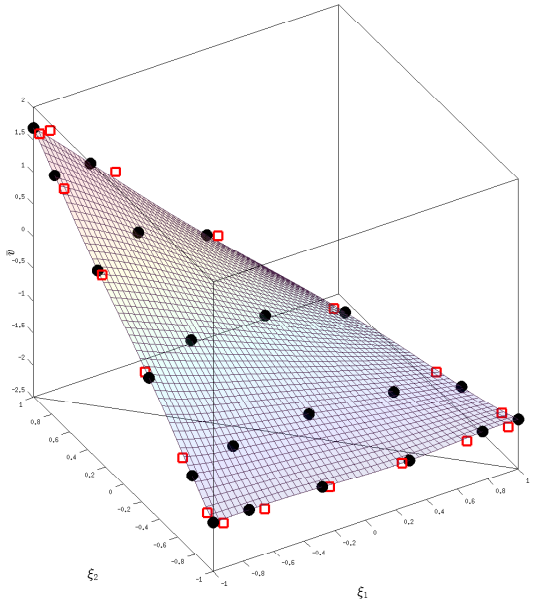


Figure 5. Solution values after filtering with boundary component exclusively, $\vec{v} = \mathcal{B}\vec{v^*}$, where $\vec{v^*} = a\vec{\xi_1^*} + b\vec{\xi_2^*} + c$ for some constants a, b, c. Red squares show the values of the solution at the boundaries, filled black circles show the filtered values of the solution at the interior points, and the meshed surface shows the polynomial representation of the solution.

Figure 6 illustrates how the boundary filtering component would behave in the case the boundary values are not coplanar. It is intersting to note that close to the corner $(\xi_1, \xi_2) = (-1, 1)$, two boundary points with different values are close to an internal point. The value of the filtered solution close to those boundary points assumes a value that is close to the average of the two.

The polynomial interpolation of the filtered interior values shows that the closer an interior point is to a boundary point, the more it will be influenced by such boundary point. This causes the filtered values close to the boundaries to get closer to the boundary values. This suggests that the boundary filtering component does in fact help in bringing the solution closer to satisfying the boundary conditions while diminishing oscillations within the element.

C. Filtered Solutions

No filtering component is used solely by itself. The factor α in Equation 23 determines how much weight to give to each component. Figure 7 illustrates the effect of the boundary values on a fully filtered solution, when $\alpha = 0.8$.

The internal filtering component reduces oscillations, while the boundary filtering component brings the interior values closer to the boundaries. By placing the boundary values on different planes, this effect becomes more apparent.

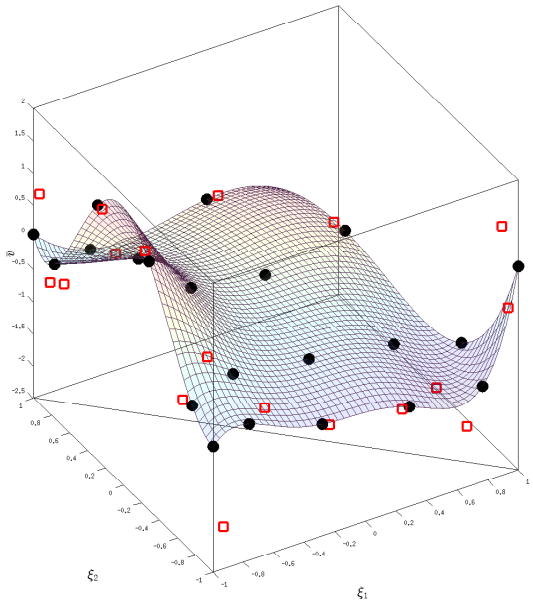
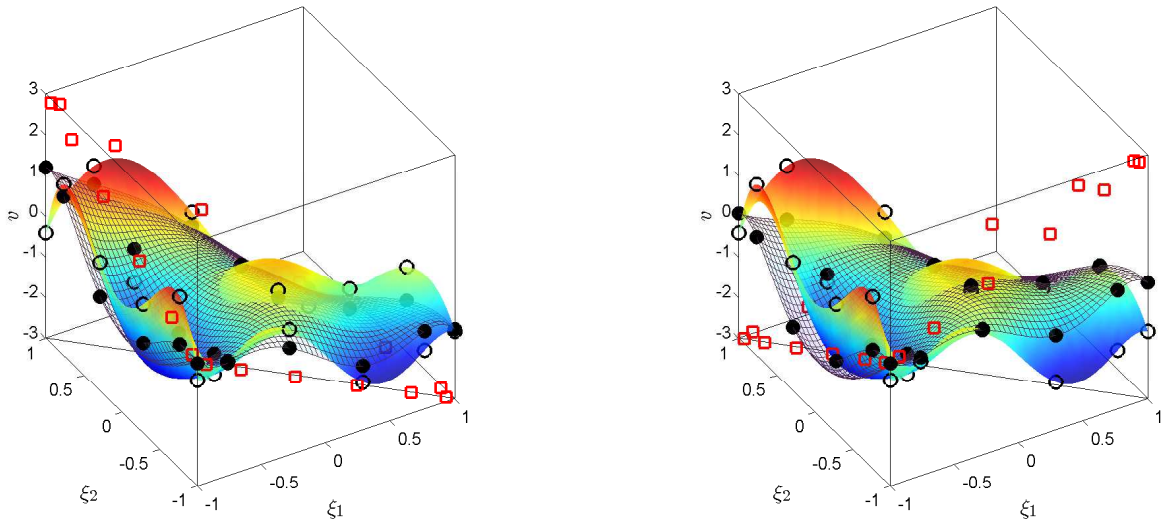


Figure 6. Solution values after filtering with boundary component exclusively, $\vec{v} = \mathcal{B}\vec{v^*}$, where $v_i^* = \sin(k\xi_{i1}^*) + \cos(k\xi_{i2}^*)$, $k = 500$. Red squares show the values of the solution at the boundaries, filled black circles show the filtered values of the solution at the interior points, and the meshed surface shows the polynomial representation of the solution.



(a) $\beta = -0.75$ (b) $\beta = 0.75$

Figure 7. Solution values after filtering with both internal and boundary components, $\vec{v} = \alpha\mathcal{T}\vec{v} + (1-\alpha)\mathcal{B}\vec{v^*}$, where $\alpha = 0.8$, $v_i = \sin(k\xi_{i1}) + \cos(k\xi_{i2})$, $k = 500$, $\vec{v^*} = \beta(-\vec{\xi_1^*} + \vec{\xi_2^*})$. Hollow black circles show the unfiltered solution at the interior points, transparent colored surface is the polynomial representation of the unfiltered solution, filled black circles show the filtered values of the solution at the interior points, hollow red squares show the value of the solution at the boundary points, and the meshed surface shows the polynomial representation of the filtered solution.

V. Results

To test the filters' ability to increase general robustness of a high-order solver, we have implemented their formulation in HiFiLES for triangular elements and performed simulations where the solver would become unstable otherwise.

We wanted to test the increase of robustness in extreme cases of grid coarseness, high Reynolds number, very low Mach number, high Mach number, and even unstable time-steps. It is important to keep in mind that in these cases we are not seeking very accurate results, but rather robustness under all conditions. In order to popularize high-order methods, we need to make them as robust as their low-order counterparts while retaining their benefit of higher accuracy with less computational and setup effort.

The goal is to have a cheap stabilization strategy that preserves boundary conditions for cases in which the mesh is not necessarily perfectly appropriate for resolving the flow physics over the entire domain. This scenario arises frequently in industrial applications, where the mesh would be properly refined at regions of interest and coarse in regions that the engineer/scientist has decided are not as important for the problem at hand.

All the simulations that follow were performed using HiFiLES[25]. 2-D Navier-Stokes equations are being solved, with varying values of Mach number (Ma), Reynolds number (Re), time-step (Δt), and filtering frequency. The common parameters are:

1. Four-stage, five-step, low-storage Runge-Kutta time-stepping method (RK45)[26] was used in the GPUs, and forward Euler was used when running a simulation in CPUs
2. Polynomial solution representation (p) of order 4. Rusanov Flux as a Riemann solver, and a Local Discontinuous Galerkin (LDG)[27] viscous flux.
3. Filters with width $h = 10$ and weghting parameter $\alpha = 0.8$.
4. Starting from uniform flow
5. Characteristic boundary conditions at the inflow and outflow. No-slip, isothermal wall boundary conditions at the cylinder's surface.
6. All quantities non-dimensionalized with free-stream temperature and cylinder wall temperature of 300, referece length of 1.
7. Flow properties: $\gamma = 1.4$, Prandtl number $\mathrm{Pr} = 0.72$, gas constant $R = 286.9\frac{J}{KgK}$, viscosity determined by Sutehrland's law with reference temperature of $291.15K$ and reference viscosity of $\mu = 1.827\mathrm{e}-5$

Results of interest are shown in Table 2. Accuracy of the results is not expected. Nevertheless, as a reference, for the flow around a cylinder at $\mathrm{Re} = 1e6$, $\bar{C_D} \approx 0.6$ in[28], $\bar{C_D} \approx 0.4$ in[29] and $\mathrm{St} \approx 0.4$ in[29].

Because of the results obtained, it is good to keep in mind that for flow around a cylinder at $\mathrm{Re} \approx 2e5$, $\bar{C_D} \approx 0.9$ in[28], $\bar{C_D} \approx 1.18$ in[29] and $\mathrm{St} \approx 0.2$ in[29].

Case	Ma	Δt	n_F	$\bar{C_D}$	St	Flow time (s)	Time steps	Wall time (hours)	Computing Resources
1	0.2	$5e-5$	1000	0.9256	0.1600	1.2069	$1,675,700$	12.65	1 4-core i7 CPU
2	0.077	$5e-5$	1000	0.9314	0.1627	15.44	$8,252,500$	11.78	1 GPU
3	0.87	$5e-5$	100	1.8383	0^*	1.3833	$8,355,256$	12.63	1 GPU
4	0.0077	$1.25e-5$	1000	1.18	0.20	53.44	$11,428,000$	59.51	2 GPUs

Table 2. Summary of simulation results. All cases were run at $\mathrm{Re} = 1e6$ with polynomial representation of order 4. Cases 1-3 were run using the mesh shown in Figure 8. Case 4 was run using the mesh shown in Figure 13. Cases with 0^* Strouhal number reached an artificial steady-state.

A. Stabilization Strategy

In the simulations presented here, the solution inside every element in the entire domain is being filtered using Equation 23 every n_F time-steps, where n_F is an integer to be determined. No sensor is being used to detect problems in the flow.

The frequency of filter application is being chosen in the following heuristic way:

1. Start the simulation with a specific time step and no filtering. Record at what time-step the simulation exits (or, more colloquially, NaNs) and note the value of the residuals at the last valid time-step. This step usually takes no more than 1 minute.

2. To ensure the simulation is ending prematurely because of grid resolution problems or presence of sharp gradients, and not because of an unstable time step, halve the time step and run the simulation again.

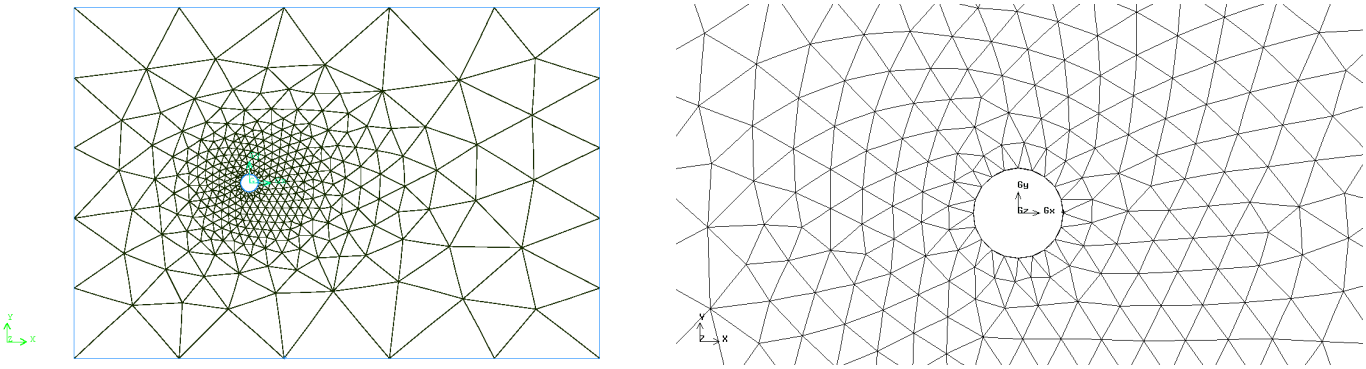
3. Wait for the simulation to exit prematurely. If the residual at this last exit is close in value to the previous exiting residual, the time step in Step 2 was stable. Set the new time step to the time step in 2. Otherwise, record the exiting residual and go back to Step 2.

4. Now that a stable time step has been found, apply the filter to the simulation every n_F time steps, where n_F is about 90% of the number of imte-steps it took the simulation to become unstable when unfiltered.

It would certainly be desirable to filter the solution at elements where a problem is detected. Nevertheless, this heuristic approach has so far enabled the stabilization of every case tried and de-couples the effectiveness of the filters from possible shortcomings of aliasing/shock sensors.

B. Coarse mesh used in simulations

In these tests, we have used the very coarse triangular mesh with 714 elements shown in Figure 8. The boundary layer is purposefully not resolved properly, as we would like to induce aliasing errors in the unfiltered calculation.



(a) Full mesh view (b) Close-up view

Figure 8. Unstructured, coarse mesh of a circular cylinder with 714 triangular elements. Elements adjacent to the cylinder have quadratic edges.

C. Flow Around a Circular Cylinder, $\mathbf{Re} = \mathbf{10^6}, \mathbf{Ma} = \mathbf{0.2}$

This was the first simulation performed after implementing the filters in HiFiLES, so the GPU implementation was not available then. The time-stepping scheme used here is simply forward Euler.

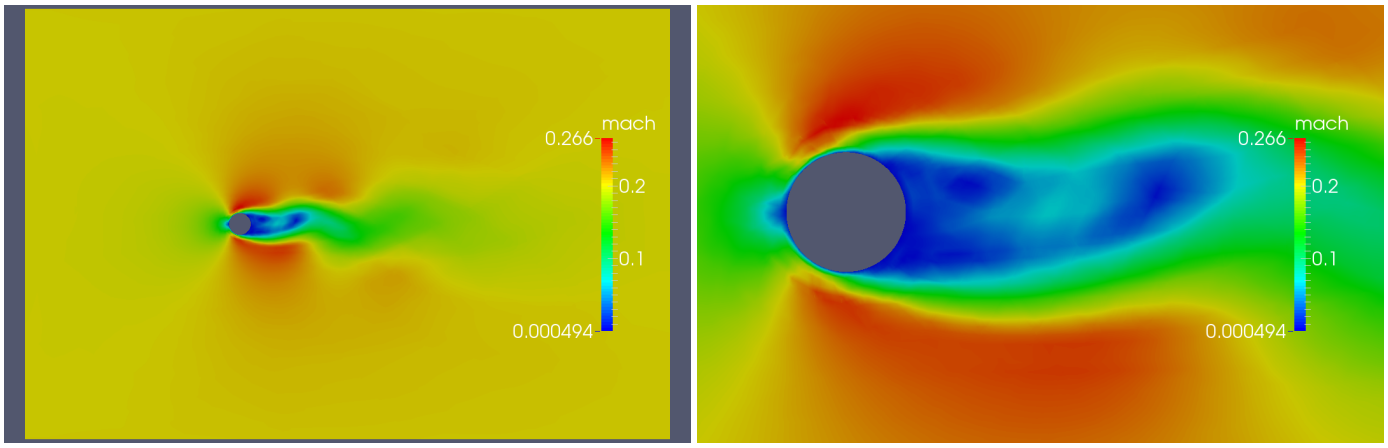
Figure 9 shows "pretty pictures" resulting from the simulation. A video of this simulation is linked here.

It is interesting to note that there is a very dissipative form of vortex shedding occurring. The wake region is long, as in lower Re-number cases.

The simulation was stable throughout and no intervention was performed while it was occurring, from the start in uniform flow to the moment it was stopped.

From this experiment, it is inclear what portion of the numerical dissipation arises from the coarse discretization and what portion is due to the filtering operation.

This case demonstrates that the stabilization strategy can work well in cases where the mesh is improperly refined: they stabilize the solution and preserve the boundary conditions.



(a) Full view (b) Close-up view

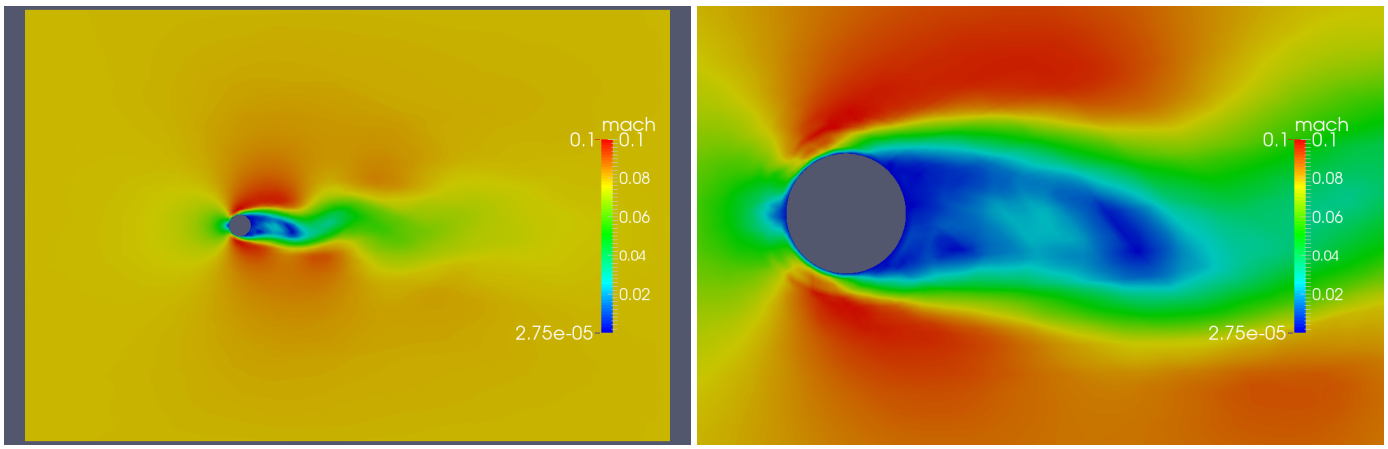
Figure 9. Flow past a cylinder. $\mathrm{Re} = 1e6, \mathrm{Ma} = 0.2, p = 4$

D. High Reynolds Number, Flow Around a Circular Cylinder, $\mathbf{Re = 10^6, Ma = 0.077}$

This was the first simulation performed using GPUs. The lower Ma case was of interest, as HiFiLES had not been able to run full simulations of flows with $\mathrm{Ma} < 0.2$.

Figure 10 shows the colorful results for this case. Once again, the boundary conditions are satisfied and the simulation is stabilized without further intervention. The same time-step size was used as in the previous case in order to leave as many parameters as possible unchanged.

A video of this simulation is linked here in real-time, and here at $0.1\times$. A feature of these simulations that can only be appreciated by watching the linked videos is that the filters have a visible effect on the regions where aliasing and instabilities are expected: the rear part of the cylinder and the boundary layer. However, even though the filters are being applied everywhere, smooth, well-resolved regions of the flow look unchanged. To what extent the smooth regions remain unchanged has not been quantified.



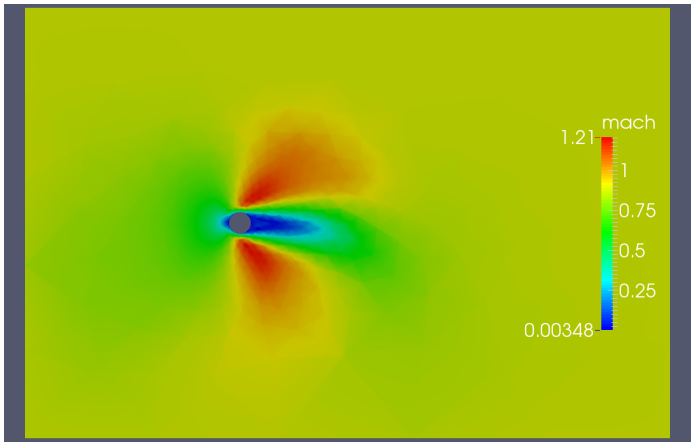
(a) Full view (b) Close-up view

Figure 10. Flow past a cylinder. $\mathrm{Re} = 1e6, \mathrm{Ma} = 0.077, p = 4$

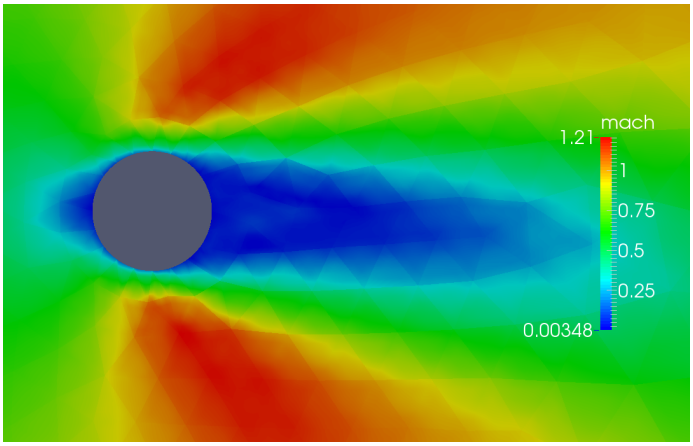
E. High Reynolds Number, Flow Around a Circular Cylinder, $\mathbf{Re = 10^6, Ma = 0.87}$

This case encompasses all potential sources of instabilities in a high-order solver: poor resolution, aliasing, and sharp gradients. Figure 11 shows plots of the solution. The simulation shows a clear un-physical asymmetry due to the coarseness of the mesh. Nevertheless, the no-slip boundary conditions are being satisfied and the shock is present.

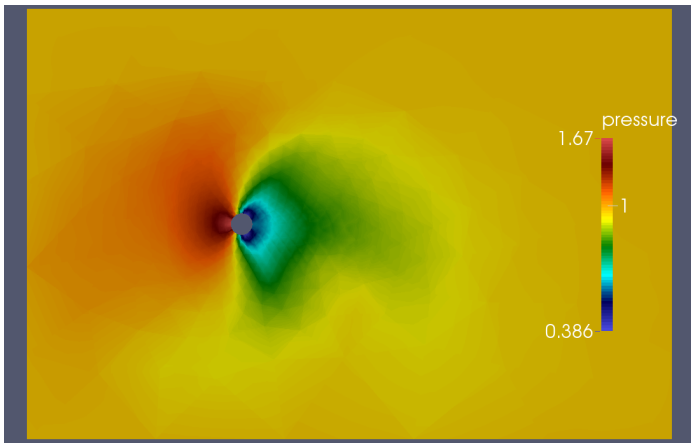
Because of the coarseness of the mesh and the high-gradients present in the solution, quite a lot of filtering had to occur. This forced the flow to a "steady state" and shown in the Residual and C_D plots in Figure 12.



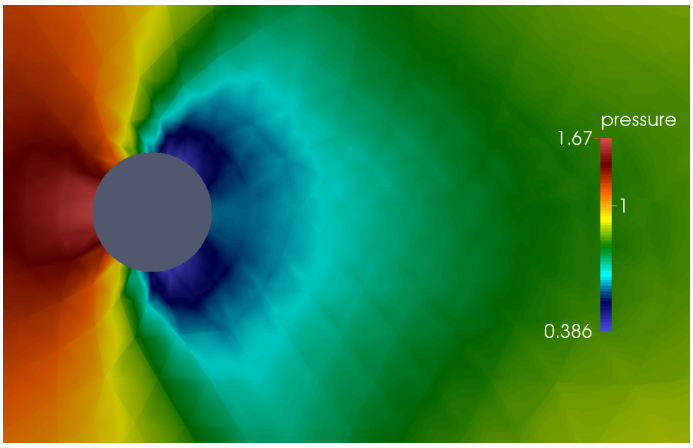
(a) Full view



(b) Close-up view



(c) Full view



(d) Close-up view

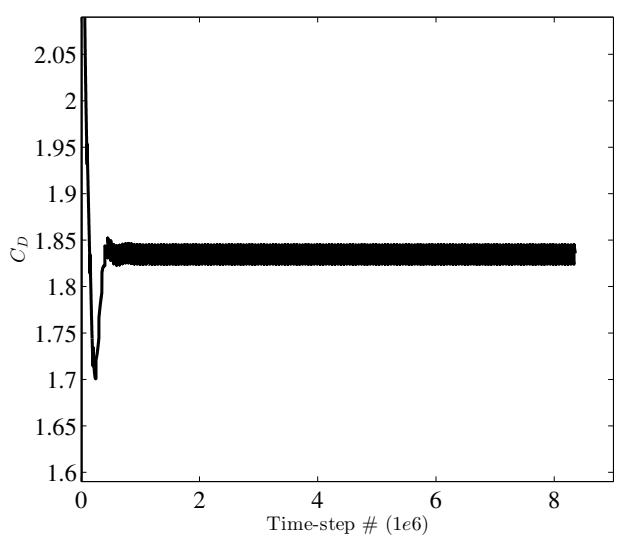
Figure 11. Flow past a cylinder. $\mathrm{Re} = 1e6, \mathrm{Ma} = 0.87, p = 4$

Values of C_D and residual history are shown in Figure 12. The value of drag "converges" after time-step $1.846e6$. The residual in the energy conservation equation also "converges" to a zig-zag pattern after this iteration. Figure 12b shows the energy residual in the last few thousand time-steps. The sharp decrease in residual magnitude reveals the time-steps at which the filter is being applied.

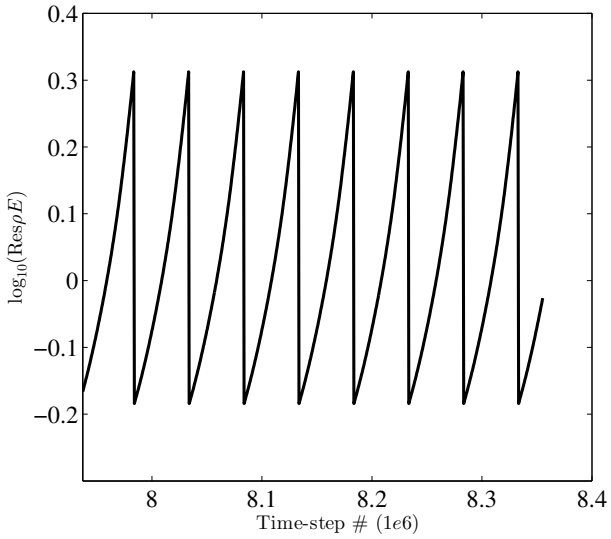
F. High Reynolds Number, Flow Around a Circular Cylinder, $\mathbf{Re = 10^6, Ma = 0.0077}$, less-coarse mesh

This case was run with the more refined mesh seen in Figure 13. This mesh is still coarse for standard turbulent computations. It is interesting to note that the predicted $C_D = 1.18$ and $\mathrm{St} = 0.20$ match experimental results for $\mathrm{Re} = 2e5$. This means physically consistent dissipation was added either via the filtering strategy or the coarseness of the grid.

A real-time video of this simulation is linked here for Mach contours, and here for vorticity strength contours. The filters stabilized this almost-incompressible simulation without a problem.

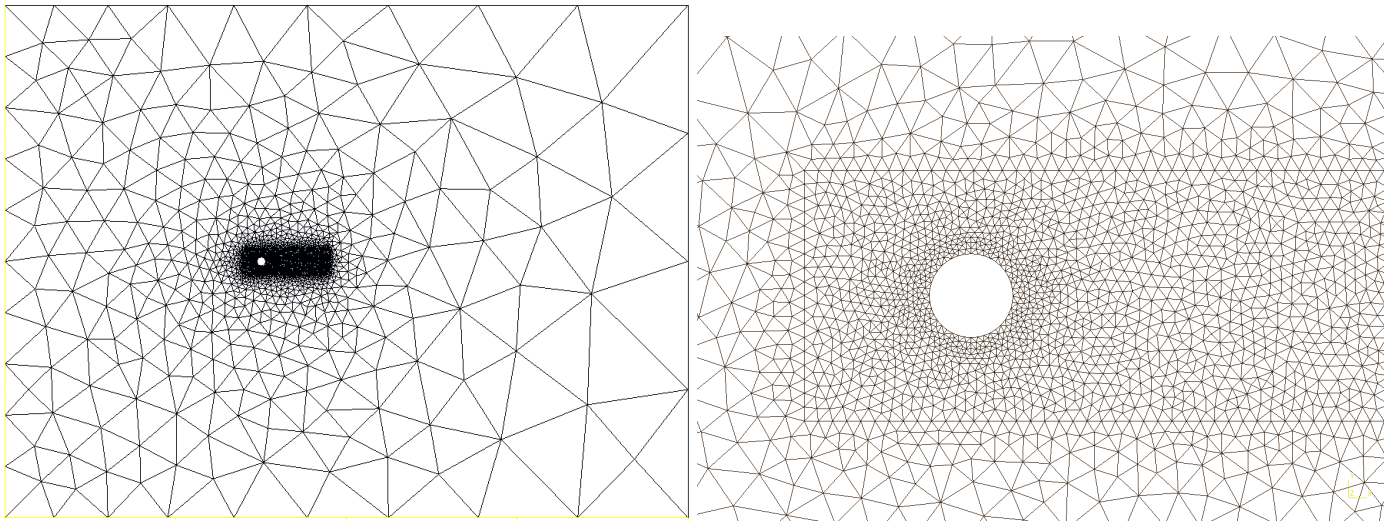


(a) Drag coefficient history over entire simulation run. "Steady state" is reached at time-step $1.846e6$.



(b) Energy residual history over the last few thousand time-steps

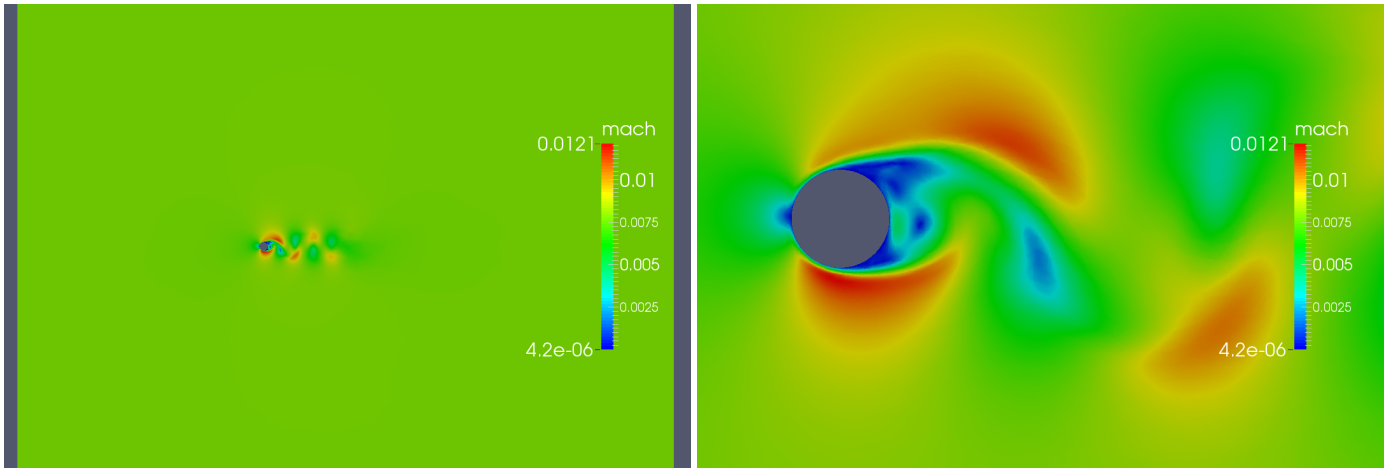
Figure 12. History of C_D and energy residual of simulation in Figure 11



(a) Full mesh view

(b) Close-up view

Figure 13. Unstructured, coarse mesh of a circular cylinder with 5,616 triangular elements. Elements adjacent to the cylinder have quadratic edges.



(a) Full view (b) Close-up view

Figure 14. Flow past a cylinder. $\mathrm{Re} = 1e6, \mathrm{Ma} = 0.0077, p = 4$

VI. Conclusion

We have suggested a formulation of Local Fourier Spectral (LFS) filters for the stabilization of Navier-Stokes solvers for unstructured grids that use a Finite Element Method-based approach to achieve high order spatial discretizations. This includes Discontinous Galerkin, Spectral Difference, Spectral Element, and Flux Reconstruction methods. The filtering operation can be performed at individual elements and maintains a local stencil by using the element's solution and boundary values. This makes their implementation highly parallelizable.

The filters have been developed with the desired properties shown in Section A.. In essence, the filters have a spectral interpretation and satisfy boundary conditions weakly. The computational cost of applying a filtering operation to a single element is two small matrix multiplications. This low cost plus the compact stencil makes the LFS filters a good alternative to using artificial dissipation.

We have shown by implementing the LFS filters in HiFiLES that little to no tuning is necessary to achieve stability in cases where instability is expected: coarse grids, high-Re flows, high-Ma flows, and low-Ma flows. In all cases, the filters preserved the boundary conditions, did not introduce visible flow anomalies, and allowed the flow to develop its natural features. The summary of results can be seen in Table 2.

Because the filter has a physical interpretation, Sub-Grid Scale (SGS) modeling could be done with the classical physical arguments. A similar type of filter has been used by Lodato[6] in the Spectral Difference scheme to do SGS modeling rather than to stabilize the solution.

The unexpected finding that the filters could bring simulations in very coarse meshes to a pseudo-steady state opens up the possibility of using the LFS filters as part of a pre-conditioning strategy or to start flow simulations from conditions more developed than uniform flow.

All algorithms are publicly available in the HiFiLES repository under the branch "LFS-filters". The filters have been fully implemented for GPU/CPU computations for triangular grids only.

VII. Future Work

Implementation in 3D elements is straightforward and shall be the immediate course of action. 3D high-Re simulations had not been possible in HiFiLES and it is expected stabilization with LFS filters will enable them.

So far in all simulations the filters are acting on all elements in the domain with a pre-determined frequency. A more surgical approach to filtering is needed. The implementation of an aliasing/shock sensor is in order. Because the filters appear to have little effect on regions where the solution is well resolved, it is acceptable if the sensor is too conservative. The sensor proposed by Persson et al.[4] for general elements and by Sheshadri et al.[30] for tensor-product elements are prime candidates.

It is clear the filters are modifying all conservation variables within an element, and very likely key flow quantities like entropy and pressure are being disturbed unnphysically. The magnitude of this disturbance needs to be quantified.

Acknowledgments

The authors would like to acknowledge the support for this work provided by the Air Force Office of Scientific Research under grant # FA9550-14-1-0186 monitored by Dr. Fariba Fahroo.

References

[1]Asthana, K., López-Morales, M., and Jameson, A., "Non-linear stabilization of high-order Flux Reconstruction schemes via Fourier-spectral filtering," *in review at Journal of Computational Physics*, 2014.

[2]Kolmogorov, A. N., "A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number," *Journal of Fluid Mechanics*, Vol. 13, No. 01, 1962, pp. 82–85.

[3]Vincent, P. and Jameson, A., "Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists," *Mathematical Modelling of Natural Phenomena*, Vol. 6, No. 03, 2011, pp. 97–140.

[4]Persson, P.-O. and Peraire, J., "Sub-cell shock capturing for discontinuous Galerkin methods," *AIAA paper*, Vol. 112, 2006, pp. 2006.

[5]Nguyen, N. C., Persson, P.-O., and Peraire, J., "RANS solutions using high order discontinuous Galerkin methods," *AIAA Paper*, Vol. 914, 2007, pp. 2007.

[6]Lodato, G., Castonguay, P., and Jameson, A., "Structural Wall-modeled LES Using a High-order Spectral Difference Scheme for Unstructured Meshes," *Flow, Turbulence and Combustion*, Vol. 92, No. 1-2, 2014, pp. 579–606.

[7]Guba, O., Taylor, M., and St-Cyr, A., "Optimization-based limiters for the spectral element method," *Journal of Computational Physics*, Vol. 267, 2014, pp. 176–195.

[8]Kuzmin, D. and Turek, S., "High-resolution FEM-TVD schemes based on a fully multidimensional flux limiter," *Journal of Computational Physics*, Vol. 198, No. 1, 2004, pp. 131–158.

[9]Castonguay, P., Williams, D., Vincent, P., López-Morales, M. R., and Jameson, A., "On the Development of a High-Order, Multi-GPU Enabled, Compressible Viscous Flow Solver for Mixed Unstructured Grids," *AIAA Computational Fluid Dynamics Conference, AIAA-2011-3229*, 2011.

[10]Hesthaven, J. and Warburton, T., *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Vol. 54, Springer, 2007.

[11]Castonguay, P., Williams, D., Vincent, P., and Jameson, A., "Energy stable flux reconstruction schemes for advection–diffusion problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 267, 2013, pp. 400–417.

[12]Huynh, H., "A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods," *AIAA paper*, Vol. 4079, 2007, pp. 2007.

[13]Vincent, P. E., Castonguay, P., and Jameson, A., "A new class of high-order energy stable flux reconstruction schemes," *Journal of Scientific Computing*, Vol. 47, No. 1, 2011, pp. 50–72.

[14]Huynh, H., "A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods," *AIAA paper*, Vol. 4079, 2007, pp. 2007.

[15]Castonguay, P., Vincent, P. E., and Jameson, A., "A new class of high-order energy stable flux reconstruction schemes for triangular elements," *Journal of Scientific Computing*, Vol. 51, No. 1, 2012, pp. 224–256.

[16]Williams, D., Castonguay, P., Vincent, P., and Jameson, A., "Energy Stable Flux Reconstruction Schemes for Advection-Diffusion Problems on Triangles," *Journal of Computational Physics*, 2013.

[17]Williams, D. and Jameson, A., "Energy Stable Flux Reconstruction Schemes for Advection–Diffusion Problems on Tetrahedra," *Journal of Scientific Computing*, 2013, pp. 1–39.

[18]Jameson, A., Schmidt, W., Turkel, E., et al., "Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes," *AIAA paper*, Vol. 1259, 1981, pp. 1981.

[19]Mirzaee, H., Ryan, J. K., and Kirby, R. M., "Efficient implementation of smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions," *Journal of Scientific Computing*, Vol. 52, No. 1, 2012, pp. 85–112.

[20]Steffen, M., Curtis, S., Kirby, R. M., and Ryan, J. K., "Investigation of smoothness-increasing accuracy-conserving filters for improving streamline integration through discontinuous fields," *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 14, No. 3, 2008, pp. 680–692.

[21]Walfisch, D., Ryan, J. K., Kirby, R. M., and Haimes, R., "One-sided smoothness-increasing accuracy-conserving filtering for enhanced streamline integration through discontinuous fields," *Journal of Scientific Computing*, Vol. 38, No. 2, 2009, pp. 164–184.

[22]Forsythe, G. E., Moler, C. B., and Malcolm, M. A., "Computer methods for mathematical computations," 1977.

[23]Nelder, J. A. and Mead, R., "A simplex method for function minimization," *The computer journal*, Vol. 7, No. 4, 1965, pp. 308–313.

[24]Jia, B., "Simplex Optimization Algorithm and Implemetation in C++ Programming," http://www.codeguru.com/cpp/article.php/c17505/Simplex-Optimization-Algorithm-and-Implemetation-in-C-Programming.htm, Accessed: 2015-05-11.

[25]López-Morales, M. R., Bull, J., Crabill, J., Economon, T. D., Manosalvas, D., Romero, J., Sheshadri, A., Watkins II, J. E., Williams, D., Palacios, F., et al., "Verification and Validation of HiFiLES: a High-Order LES unstructured solver on multi-GPU platforms," 2014.

[26]Carpenter, M. H. and Kennedy, C. A., "Fourth-order 2N-storage Runge-Kutta schemes," 1994.

[27]Cockburn, B. and Shu, C.-W., "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[28]Achenbach, E., "Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to Re= 5$\times$ 106," *Journal of Fluid Mechanics*, Vol. 34, No. 04, 1968, pp. 625–639.

[29]Roshko, A., "Experiments on the flow past a circular cylinder at very high Reynolds number," *Journal of Fluid Mechanics*, Vol. 10, No. 03, 1961, pp. 345–356.

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