Construction of commutative filters for LES on unstructured meshes

By Alison L. Marsden, Oleg V. Vasilyev† AND Parviz Moin

1. Motivation and objectives

Application of large eddy simulation (LES) to flows with increasingly complex geometry necessitates the extension of LES to unstructured meshes. A desirable feature for LES on unstructured meshes is that the filtering operation used to remove small scale motions from the flow commutes with the differentiation operator. If this commutation requirement is satisfied, the LES equations have the same structure as the unfiltered Navier Stokes equations. Commutation is generally satisfied if the filter has a constant width. However, in inhomogeneous turbulent flows, the minimum size of eddies that need to be resolved varies throughout the flow. Thus, the filter width should also vary accordingly. Given these challenges, the objective of this work is to develop a general theory for constructing discrete variable width commutative filters for LES on unstructured meshes.

Variable width filters and their commuting properties have been the focus of several recent works. Van der Ven (1995) constructed a family of continuous filters which commute with differentiation up to arbitrary order in the filter width. However, this set of filters applies only to an infinite domain without addressing the practical issue of boundary conditions in a finite domain. More recently, a class of discrete commutative filters was developed by Vasilyev *et al.* (1998) for use on nonuniform structured meshes. Their formulation uses a mapping function to perform the filtering in the computational domain. Although this type of mapping is impossible for the unstructured case, the theory developed in Vasilyev *et al.* (1998) was used as a starting point for the present work.

In this paper we present a theory for constructing discrete commutative filters for unstructured meshes in two and three dimensions. In addition to commutation, other issues such as control of filter width and shape in wavenumber space are also considered. In particular, we wish to specify a desired filter width at each point in space and obtain a discrete filter which satisfies this requirement regardless of the choice of the computational mesh.

2. Commutation error of filtering and differentiation operations in physical space

Recently Vasilyev *et al.* (1998) developed a general theory of discrete filtering in arbitrarily complex geometries. With the use of a mapping function, the filtering is done in the computational domain. Here, we extend the theory of commutative filters developed in Vasilyev *et al.* (1998) to the physical domain. We begin by discussing filtering in one-dimensional space and then extend it to three spatial dimensions.

† Dept. of Mechanical & Aerospace Engr., University of Missouri, Columbia, MO 65211

2.1. *Commutation error in one spatial dimension*

Following Vasilyev *et al.* (1998), an operator to measure commutation error is defined as follows. Given a function $\phi(x)$, the commutation error is

$$\left[\frac{d\phi}{dx}\right] = \overline{\frac{d\phi}{dx}} - \frac{d\overline{\phi}}{dx}. \tag{2.1}$$

where over-bar denotes the filtered quantity. The continuous filtering operation is defined by

$$\overline{\phi}(x) = \frac{1}{\Delta(x)} \int_a^b G\left(\frac{x-y}{\Delta(x)}, x\right) \phi(y) dy, \tag{2.2}$$

where $\Delta(x)$ is the filter width and $G(\eta, x)$ is the location dependent filter function. With the change of variables $\eta = \frac{x-y}{\Delta(x)}$, Eq. (2.2) can be written as

$$\overline{\phi}(x) = \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} G(\eta, x)\, \phi(x - \Delta(x)\eta)\, d\eta. \tag{2.3}$$

Taking the Taylor series expansion of $\phi(x - \Delta(x)\eta)$ in powers of Δ gives

$$\phi(x - \Delta(x)\eta) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) \eta^l \mathcal{D}_x^l \phi(x), \tag{2.4}$$

where $\mathcal{D}_x = d/dx$ is the derivative operator. This series was proven to be convergent in Vasilyev *et al.* (1998) for the case of uniform Δ by assuming that the spectrum did not include wavenumbers higher than some finite cutoff wavenumber k_{max}. The proof is analogous for the case of varying Δ and with the same assumptions the radius of convergence in this case is considered to be infinite. Substituting (2.4) into (2.3) and changing the order of summation and integration, we have

$$\overline{\phi}(x) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) \phi^l(x) \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} \eta^l G(\eta, x) d\eta. \tag{2.5}$$

Defining the filter moment as

$$M^l(x) = \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} \eta^l G(\eta, x) d\eta \tag{2.6}$$

and substituting (2.6) into (2.5), we obtain

$$\overline{\phi}(x) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) \mathcal{D}_x^l \phi(x). \tag{2.7}$$

In the same manner as in Vasilyev *et al.* (1998) we let

$$M^l(x) = \begin{cases} 1 & l = 0 \\ 0 & l = 1, \ldots, n-1. \end{cases} \tag{2.8}$$

With this definition we have

$$\overline{\phi} = \phi(x) + \sum_{l=n}^{\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) \mathcal{D}_x^l \phi(x). \tag{2.9}$$

The filter of the derivative of the function is

$$\overline{\frac{d\phi}{dx}}(x) = \frac{d\phi}{dx} + \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) \mathcal{D}_x^{l+1} \phi(x). \tag{2.10}$$

The derivative of the filtered quantity is

$$\frac{d\overline{\phi}}{dx}(x) = \frac{d\phi}{dx} + \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \frac{d}{dx} \left(\Delta^l(x) M^l(x) \mathcal{D}_x^l \phi(x) \right). \tag{2.11}$$

Applying the chain rule to (2.11) and subtracting (2.11) from (2.10), we obtain an expression for the commutation error:

$$\left[\frac{d\phi}{dx} \right] = \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \left\{ \frac{d}{dx} \left(\Delta^l(x) M^l(x) \right) \right\} \mathcal{D}_x^l \phi(x). \tag{2.12}$$

Using the properties in (2.8) it follows that

$$\frac{dM^l}{dx}(x) = 0 \quad \text{for } l = 1, \ldots, n-1. \tag{2.13}$$

As a result, the local commutation error is

$$\left[\frac{d\phi}{dx} \right] = O\left(\Delta^n(x) \right), \tag{2.14}$$

provided that $d\Delta/dx = O(\Delta)$, which is true for most of the smoothly varying grids. For highly stretched grids $d\Delta/dx$ is $O(\Delta^\gamma)$, $\gamma < 1$, which results in lowering the order of the commutation error to $O\left(\Delta^{n+\gamma-1}\right)$. The extension to three dimensions is quite straightforward, and has been presented in Marsden (1999).

3. Construction of discrete commutative filters

The filters developed by Vasilyev *et al.* (1998) were constructed by applying the necessary number of constraints to the filter weights to achieve both commutation and an acceptable filter shape. The following constraints were imposed in finding the filter weights. The zeroth moment should be one, a specified number (order of commutation error) of higher moments should be zero, and other constraints were added for defining the filter shape.

These ideas were used as a starting point for developing filters for the unstructured case. However, in the unstructured mesh formulation it is impossible to use the same discrete filter at all points on the mesh as was possible in Vasilyev *et al.* (1998). Instead, filter weights must be computed at each mesh point and stored in a table. This restriction means that the algorithm must have a way to assess the filter shape at each point since the user cannot adjust the filter constraints by hand at each mesh point.

An initial formulation for filter construction on an unstructured mesh used the ideas presented in Vasilyev *et al.* (1998) generalized to physical space. Given a mesh point to filter about, a set of neighboring points was chosen to make up the filter. Then, constraints were applied directly on the filter moments and shape in order to determine the filter weights. This procedure followed directly from Vasilyev *et al.* (1998). Two problems arose in implementing this method. First, it was found that in the case of a non-uniform point distribution such as an unstructured mesh, the shapes of the resulting filters were highly

unpredictable. In order to overcome this problem, the filter construction algorithm would have to choose the most appropriate constraints to apply based on some filter shape criterion. Second, the nature of unstructured meshes is such that a point may have any number of neighboring points. The algorithm would, therefore, have to decide which points to include and possibly apply different constraints at each mesh point, leading to inconsistencies in the filters from one part of the mesh to another.

Greater predictability and ease of implementation can be gained by using interpolation based filters to achieve commutation rather than directly implementing constraints as discussed above. The construction of discrete filters on unstructured meshes is motivated by work on interpolating wavelets Donoho (1992) and the theory of second generation wavelets Sweldens (1996), Sweldens (1997), Daubechies & Sweldens (1998). To illustrate the idea of construction of discrete filters based on polynomial interpolation, let us consider a one-dimensional example. Suppose we have a set of N unevenly spaced grid points x_i and the values of the function f_i are known at these points. We can uniquely define the $N-1$ order polynomial $P_{N_1}(x)$ that passes through the data. Polynomial coefficients are uniquely determined by locations x_i and values f_i. Evaluating this polynomial at the point x_0 and substituting the values of the polynomial coefficients expressed in terms of the values f_i, we easily find that $P_{N-1}(x_0) = \sum_{k=1}^{N} w_i f_i$. If we treat these weights as the weights of the corresponding discrete filter, then this filter will have the unique property that when it is applied to the polynomial of degree less than $N-1$ it does not change this polynomial. Then the discrete filter moments defined by

$$M^l = \sum_{k=1}^{N} w_k (x_k - x_0)^l \tag{3.1}$$

automatically satisfy the conditions (2.8) since $(x - x_0)^l$ is exactly zero at $x = x_0$ for $l = 1, \ldots, N-1$ and 1 for $l = 0$. Consequently, the discrete filters based on polynomial construction automatically guarantee an N^{th} order commutation error. To control the shape and other properties of the discrete filters, we can construct a filter as a linear combination of as many polynomial based filters as we like while preserving the commutation properties of the filter. The same idea can be easily extended to n dimensions using an n-dimensional polynomial. This simple idea gives us all the flexibility we need to construct filters with the desired shape and properties in any dimension, yet it is very straightforward to implement.

In general, with an N^{th} order numerical scheme, the filtering operation must commute to order N. Reducing error further has no significant impact on overall accuracy because the discretization error is also of order N. As in the case for a structured mesh, the filters developed here must have $N-1$ zero moments to commute to order N. In developing filters for an unstructured mesh, we will begin by assuming a second order finite difference scheme. However, as discussed above, the extension to a higher order method is straightforward. With this second order scheme in mind, we proceed with the goal of developing filters which ensure a second order commutation error. A two-dimensional discrete filter based on first order polynomial interpolation can be constructed using a triangle where (x_0, y_0) is the point where we want the filtered value. A triangle is chosen because in two dimensions three points are needed for exact reconstruction of a first order polynomial. Weights are calculated by fitting a polynomial to the vertices of the triangle, and they are then used to find a weighted average at the central point (x_0, y_0). The shape of the resulting filter in wavenumber space is very well defined, and the number of points used can be the same at each mesh point because any three close points can be chosen to

make up a triangle. The method for finding the filter weights using a triangle in two dimensions is presented here, but is extended to three dimensions in Marsden (1999). Details on choosing which points to use in the filter are discussed in Sections 4.1 and Marsden (1999).

The vector of interpolating weights, $\mathbf{w}$ is calculated as follows. Let (x_1, y_1), (x_2, y_2), and (x_3, y_3) be coordinates of the points where the function is given and (x_0, y_0) be the coordinates of the point to interpolate to. Let

$$P(x, y) = a_{00} + a_{10}(x - x_0) + a_{01}(y - y_0) \tag{3.2}$$

be a first order polynomial interpolant. Requiring that interpolant (3.2) goes through the data points f_i $(i = 1, \ldots, 3)$, we obtain the following set of linear equations

$$\begin{aligned} f_1 &= a_{00} + a_{10}(x_1 - x_0) + a_{01}(y_1 - y_0), \\ f_2 &= a_{00} + a_{10}(x_2 - x_0) + a_{01}(y_2 - y_0), \\ f_3 &= a_{00} + a_{10}(x_3 - x_0) + a_{01}(y_3 - y_0). \end{aligned} \tag{3.3}$$

Note that interpolant (3.2) is chosen such that a_{00} is the value of interpolant at point (x_0, y_0). This value is also the weighted sum of the functional values given by

$$P(x_0, y_0) = w_1 f_1 + w_2 f_2 + w_3 f_3, \tag{3.4}$$

where, w_i are the filter weights.

Some manipulation shows that the weights can be simply calculated with a single matrix inversion. If

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 - x_0 & y_1 - y_0 \\ 1 & x_2 - x_0 & y_2 - y_0 \\ 1 & x_3 - x_0 & y_3 - y_0 \end{pmatrix} \tag{3.5}$$

and

$$\mathbf{b} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \tag{3.6}$$

the vector of weights, $\mathbf{w}$, is given by

$$\mathbf{w} = \mathbf{b} \cdot \mathbf{A}^{-1}. \tag{3.7}$$

We now have weights which make up a two-dimensional discrete filter which satisfies commutation to second order. The three-dimensional equivalent is straightforward and requires four points instead of three to satisfy commutation. The extension to three dimensions is discussed in Marsden (1999).

4. Implementation of commutative filters

4.1. *Two-dimensional filters*

In Section 3 we demonstrated construction of discrete two-dimensional filters with second order commutation error using polynomial interpolation. The result was a set of discrete triangular filters with weights assigned to each vertex and the central point. Using these triangular filters as a basis, the topic of this section is the construction of commutative filters which combine multiple triangles into one filter and allow for a variable filter width.

Although one triangular filter satisfies the properties of commutation, it is undesirable because it offers no flexibility in filter width or shape. Also, an ideal filter would

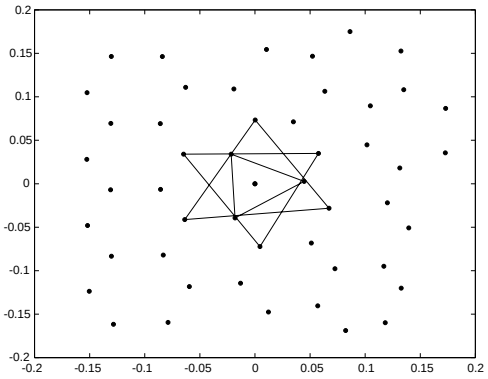

FIGURE 1. Example of filter constructed with triangles on an unstructured grid.

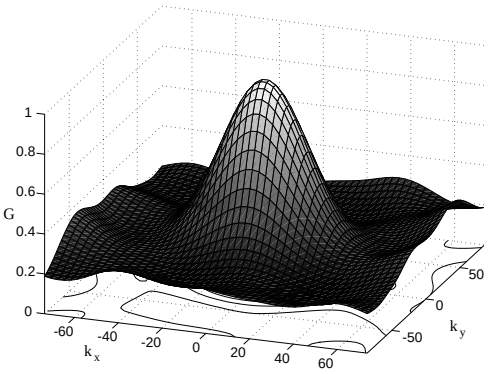

FIGURE 2. Transfer function corresponding to filter in Fig. 1.

include more neighboring points to obtain a nice distribution. To take advantage of the predictability of simple triangle based filters while adding flexibility in filter width, it is possible to use a linear combination of multiple triangular filters. This method offers the advantage of a predictable and well defined transfer function shape while ensuring that the resulting filter will satisfy commutation to the same order as the basis triangles.

Figure 1 shows an example of a 2-D filter constructed from three triangles. The corresponding transfer function, which has very nice characteristics, is shown in Fig. 2. To achieve flexibility in filter width, each triangle as well as the central point is assigned a weight which applies equally to all vertices of the triangle. We will refer to the weights on triangles as β_i whereas the filter weights on individual vertices calculated in section 3 are w_i. The value of β_i can be varied from 0 to 1 as long as the sum total is 1. The optimum value of β for the central point is $1/2$ because this results in the most desirable filter shape.

4.2. *Implementation in two dimensions*

We are now ready to discuss details of filter construction in two dimensions. The first task for the filter construction algorithm is to choose the set of points to include in the filter. Each included point is part of a triangle which will later be linearly combined with other triangles to form the total filter as discussed in section 4.1. The number of triangles included in each filter may be specified by the user; however, the filters presented in this paper consist of three triangles because this provides a nice distribution of points on a

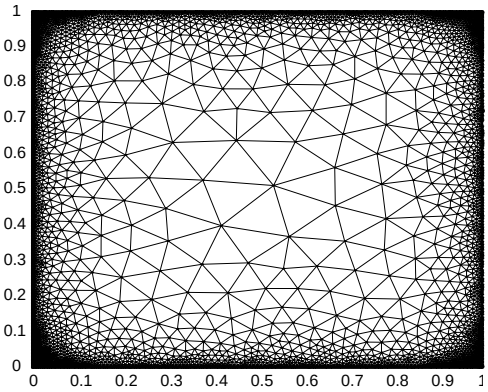

FIGURE 3. Example of mesh used for filter development.

regular unstructured mesh in two dimensions. After the set of points is chosen, the next step is to calculate the weights associated with each mesh point included in the filter. From these, we calculate the transfer function G and apply the filter to the discrete data. Figure 3 shows an example mesh which was used in testing the algorithm which chooses filter points.

Points to include in the filter are selected based on chosen criterion for the triangles which make up the filter. Given a point to filter about, surrounding points are searched in groups of three until a set of triangles to use in the filter is settled on. It is obviously undesirable to search points on the entire mesh because of computational cost. Because of this, the first step in the algorithm is to come up with a set of neighboring points to include in the search. This is done by using the tree structure of the mesh connectivity to obtain a set of surrounding points. In two dimensions, three levels of the tree are sufficient for obtaining a large enough group of points.

Having found a group of surrounding mesh points, the next step is to calculate the distance to each point in the group as well as the angle from the x axis. The points are then sorted according to angle into three zones of 120^o each. The zones are created to ensure that the chosen points have a nice distribution of angles about the central point. Figure 4 shows these three zones. Within each zone, the points are sorted by distance from the central point.

Triangles are systematically formed by taking a point from each zone, starting with the closest point in each, and then seeing if the chosen triangle meets the criterion for being included in the filter. For each triangle formed, we must determine whether to use it in the filter or continue the search by trying the next combination of three points. When three triangles have been found, the choice of points for the filter is complete.

Triangles must satisfy two criterion to be selected for use in the filter. First, the central point must be inside the triangle and second, the central point must be as close to the centroid of the triangle as possible. Both criterion involve drawing lines from the central point to each vertex of the triangles to form three sub-triangles. If the summed area of the sub-triangles exceeds that of the larger triangle, the central point is outside. If the area of any of the sub triangles is a large percentage of the total area of the triangle, the central point is too close to the side of the triangle. The allowable percentage is a user specified parameter. If one of these checks is true, the triangle is rejected and we advance to the next row of the table, continuing until the desired number of triangles has been found.

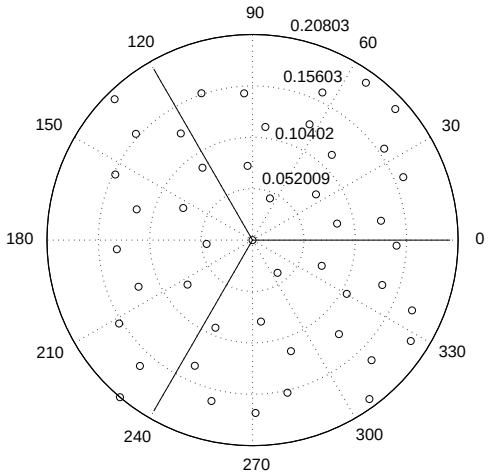

FIGURE 4. Mesh nodes sorted by angle.

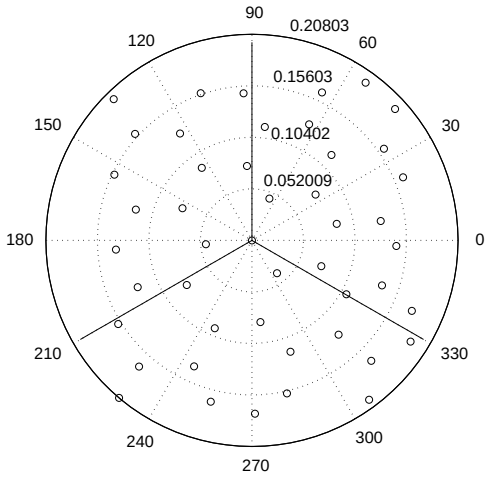

FIGURE 5. Mesh nodes sorted by angle, rotated from Fig. 4.

This procedure has one drawback. When the best choice of triangle has two points in the same region, usually very close to the region boundaries, it is never tried as a possibility for the filter. As a solution to this problem, the next step in the algorithm is to rotate the zone boundaries as shown in Fig. 5, and the procedure of choosing triangles is performed again, returning a new set of triangles. The set of triangles whose collective weights are closest to one third is then chosen to make up the final filter.

We now have a set of three triangles to make up the filter which can be linearly combined to create the complete filter as described in Section 4.1. Flexibility is gained by applying the same filter over again with different triangle weights, β_i, to capture low and high wavenumbers and achieve a nice transfer function shape. In addition, by applying the same filter more than once, it is possible to increase the filter width until the desired value is reached. With this method it also becomes possible to exactly specify the filter ratio for use in the dynamic model.

5. Prescribing the filter width

The main advantage of the filtering method presented is that the filter width can be prescribed by the user *a priori*. For example, in a boundary layer it is typical to use an exponential form of the filter width function since the scale of eddies increases with distance away from the wall. The filter width can be defined as the radius Δ of an equivalent circular top hat filter. The corresponding second moment, M_2, can be defined by the following integral.

$$M_2 = \int_0^{2\pi} \int_0^{\Delta} r^2 \cos^2\theta \frac{1}{\pi\Delta^2} r dr d\theta. \tag{5.1}$$

Evaluating this integral we have a relation between the second moment and the filter width.

$$\Delta = \sqrt{4M_2} \tag{5.2}$$

Given a target value for the filter width, which is specified by the user, the above relation can be used to prescribe target values for the second moment. This second moment target value can be used to find the values of β which will result in the closest value to the desired value of Δ. The values of β are chosen by solving the following set of equations using least squares method with constraints on the values of β as follows, where β_0 is the value assigned to the central point and $\beta_1, \beta_2, \beta_3$ are assigned to the three triangles in the filter.

$$\beta_0 = 1/2 \tag{5.3}$$

$$\beta_1 + \beta_2 + \beta_3 = 1 \tag{5.4}$$

The target second moment values in x, y, and xy are M^{02}, M^{20}, and M^{11} respectively and m_i^{02}, m_i^{20}, and m_i^{11} are the moments of the individual triangles.

$$\begin{aligned} M^{02} &= m_1^{02}\beta_1 + m_2^{02}\beta_2 + m_3^{02}\beta_3 \\ M^{20} &= m_1^{20}\beta_1 + m_2^{20}\beta_2 + m_3^{20}\beta_3 \\ M^{11} &= m_1^{11}\beta_1 + m_2^{11}\beta_2 + m_3^{11}\beta_3 \end{aligned} \tag{5.5}$$

6. Demonstrating commutation

In order to validate that the filters developed commute to the desired order, a series of numerical tests were performed. Measuring the commutation properties of the directional derivatives was found not to be a good test, since the accuracy of derivative calculations strongly depends on the orientation of the mesh element, and as a consequence the the resulting truncation error is very nonuniform. As it was demonstrated in Perot (2000), a more natural operation to perform on an unstructured mesh is to calculate the curl at each mesh point. It is relatively straightforward to demonstrate that commutative properties of filtering and the curl operators are the same as discussed in Section 2.

Using the curl operator and the notation of Section 2.1, the commutation error is defined by

$$[\nabla \times f] = \nabla \times \overline{f} - \overline{\nabla \times f} \tag{6.1}$$

where f is the value of the prescribed function at each mesh point. The trivial case of the linear function $f = ax + by + c$ was used to verify that the resulting error was within machine zero range. With this check complete, the commutation and truncation errors were calculated on a series of consecutively finer meshes, using the equations describing

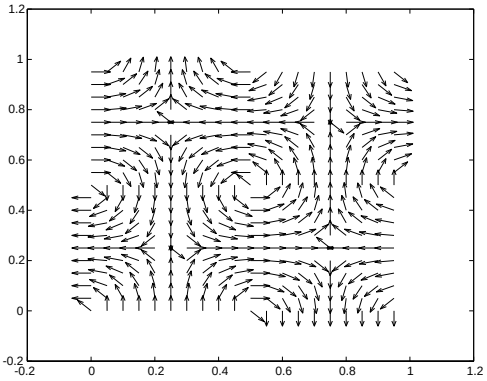

FIGURE 6. Streamlines of vortices in a box.

a vortex in a box.

$$u = -\cos(\beta_1 x + \phi_1)\sin(\beta_2 y + \phi_2) \tag{6.2}$$

$$v = \sin(\beta_1 x + \phi_1)\cos(\beta_2 y + \phi_2) \tag{6.3}$$

A plot of streamlines of these equations for the case $\beta_1 = \beta_2 = 2\pi$, $\phi_1 = \phi_2 = 0$ is shown in Fig. 6. The truncation error for the above function is the difference between the exact value of the curl

$$C_{exact} = (\beta_1 + \beta_2)[\cos(\beta_1 x + \phi_1)\cos(\beta_2 y + \phi_2)] \tag{6.4}$$

and the numerically calculated value.

The procedure for calculating the commutation error at each mesh element center using the curl operator is as follows. First, the functional values are calculated at the cell centers. These values are filtered discretely at the cell centers using the method described in Section 4.2, and then the curl of the filtered value is found numerically using surrounding filtered values. This gives the first term in Eq. 6.1. The second term in Eq. 6.1 is found by first taking the curl of the functional values at all cell centers and then filtering these values using the values of the curl which were found at adjacent mesh elements. The commutation error can then be compared to the truncation error at each mesh element center.

Both high and low order integration schemes were used to calculate the curl. For simplicity, only filters with with one triangle, or three filter points, were used. Using the low order integration method, the circulation is computed by interpolating values from the cell centers to the midpoints of the edges to find the velocity tangent to the side of the element and then integrating over the three triangle edges. The curl is then obtained by dividing by the cell area. Overall this is a first order operation. Using this method, it was found that the resulting commutation and truncation errors had first order convergence. Thus, the second order accuracy of the filtering method had been reduced when integration was performed. A plot of the commutation and truncation errors using the curl operation is shown in Fig. 7. All error plots are the L_∞ error vs. square root of the number of mesh points, and all use the same series of increasingly finer meshes to demonstrate convergence. Plotted in this way, the slope of the error is equal to the order of accuracy. All errors have been normalized with the maximum value of the curl.

The high order integration method was developed for a uniform symmetric unstruc-

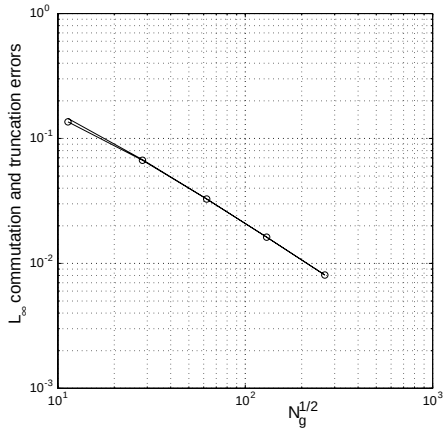

FIGURE 7. Normalized L_∞ commutation and truncation errors of curl, low order method. ——— ,truncation error; −○-, commutation error.

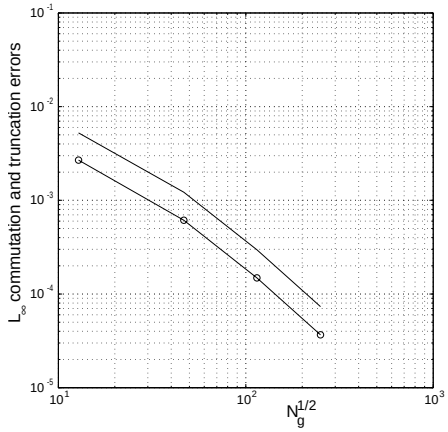

FIGURE 8. Normalized L_∞ commutation and truncation errors of curl, high order method. ——— ,truncation error; −○-, commutation error.

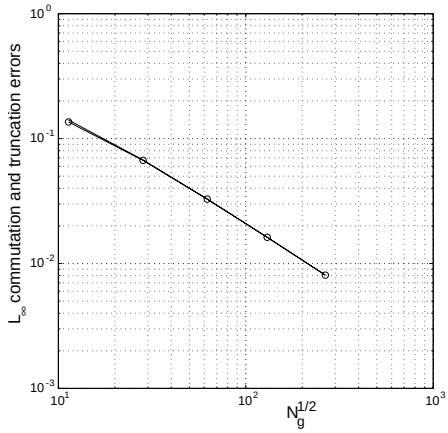

FIGURE 9. Normalized L_∞ commutation and truncation errors of circulation, low order method. ——— ,truncation error; −○-, commutation error.

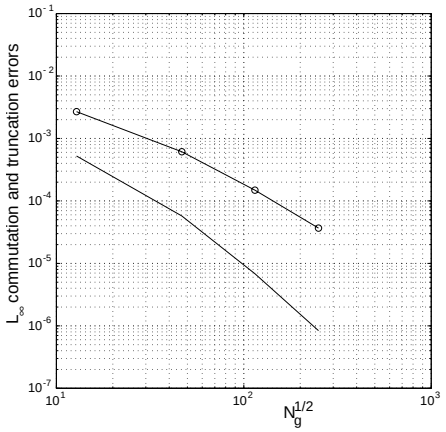

FIGURE 10. Normalized L_∞ commutation and truncation errors of circulation, high order method. ——— ,truncation error; −○-, commutation error.

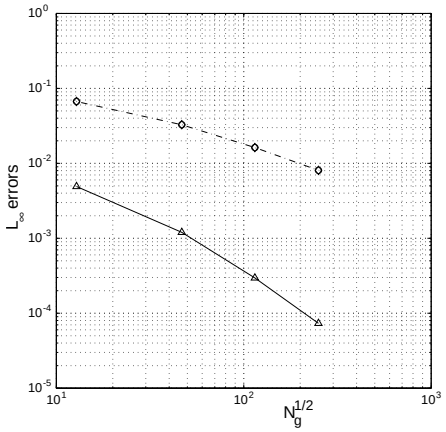

FIGURE 11. Comparison of filtered and unfiltered normalized L_∞ truncation errors. ○: $\Gamma_{low}(V) - \Gamma_{high}(V)$; ◇: $\Gamma_{low}(\overline{V}) - \Gamma_{high}(\overline{V})$; –△–: $\overline{\Gamma}_{low}(V) - \overline{\Gamma}_{high}(V)$; —·— : $\Gamma_{low}\overline{V} - \overline{\Gamma}_{low}(V)$.

tured mesh. While use of a "structured" unstructured mesh is ludicrous for any practical application, it serves the purpose of demonstrating commutation of the filtering method. As discussed earlier, the filtering method presented here can be applied to any unstructured mesh. The integration method first requires interpolating to the vertices from twelve surrounding cell centers, using weights 2/9 for the nearest six cells and $-1/18$ for neighboring cells. The routine then interpolates to the edges using the two vertices and two cell centers which lie on the line connecting the edge midpoint and the vertices of the neighboring points using the weights $(-1/16,\ 9/16,\ 9/16,\ -1/16)$. Finally, integration can be carried out along each edge of the cell using the value at the edge midpoint and the values at the two vertices using the weights $(1/6,\ 2/3,\ 1/6)$. A plot of the high order method using the curl operator is shown in Fig. 8. Due to the division by area, the curl operator is second order regardless of integration scheme. So, we see second order convergence of both commutation and truncation error as expected. However, it is important to

note that the magnitude of the commutation error is consistently less that the truncation error.

The results using the curl operator show that the truncation error is associated with division by area even for the higher order scheme. In order to eliminate this difficulty, we can introduce the discrete values of circulation that correspond to each mesh element. We can therefore compute the commutation error using values of circulation. In this way, we can clearly demonstrate that the truncation error will be higher order, and the commutation properties of the filter will become apparent. The exact value of circulation must be found for each mesh element to compute the truncation error. This is done by integrating the function exactly along the sides of each mesh element.

The commutation error using the circulation was analyzed using the low order method, and results are shown in Fig. 9. These plots also show a first order convergence in which the commutation error was contaminated due to the truncation error.

We are able to show that with this low order integration scheme, using both the curl and circulation, the commutation error is contaminated by the truncation error due to high frequency components. Figure 11 shows the truncation error and the filtered truncation error and proves that, without these high frequency components, the truncation error is a full order of magnitude decreased. Therefore, the commutation error is contaminated and the values of the commutation error and truncation error are nearly identical.

The high order method applied with the circulation shows a third order convergence in the truncation error and a second order convergence in the commutation error. This time, the order of integration was high enough that the commutation error was not affected, and we can confirm that it has second order convergence. The results for this case are shown in Fig. 10. The convergence of the truncation error for the cases using circulation, compared with the cases using the curl, confirms that dividing by the area reduced the order of magnitude by two. These tests confirm that the filtering method has a second order commutation error, which was predicted in section 3. Although filters consisting of one triangle were used, it is fully expected that filters with multiple triangles will have similar convergence properties, thus giving the user control over the filter width. In addition, even though it was necessary to use a uniform mesh for the numerical tests, it is fully expected that the filtering method used will have similar convergence properties for any unstructured mesh.

7. Conclusions

A method of constructing commutative filters for unstructured LES has been developed and validated. The method is intended for use in the ASCI unstructured solver at the Center for Turbulence Research for use in a wide variety of applications. The convergence tests performed confirm that the filtering method leads to a second order commutation error and can therefore be used in conjunction with a second order accurate numerical scheme.

One important feature of the method of filter construction presented here is that it has no requirements on the type of mesh used. Because the filter can be constructed simply from a set of points in two- or three-dimensional space, there are no constraints on the shape of mesh elements or the connectivity. It is possible to use connectivity to improve the efficiency of the algorithm, but the method remains general to any mesh. In addition, the filters presented have a consistent filter shape and flexible filter width. This allows the filter width ratio to be exactly specified for use in the dynamic model.

It would be relatively straightforward to extend the filter construction procedure developed here to higher order accuracy. For example, if one desired to use a third order finite difference scheme, the polynomial interpolant would have to be second order, requiring six neighboring points in two dimensions.

REFERENCES

Carati, D. & Eijnden, E. V. 1997 On the self-similarity assumption in dynamic models for large eddy simulations. *Phys. Fluids.* **9**(7), 2165-2167.

Daubechies, I. & Sweldens, W. 1998 Factoring Wavelet Transforms into Lifting Steps. *J. Fourier Anal. Appl.* **4**(3), 245-267.

Donoho, D. L. 1992 Interpolating Wavelet Transforms. *Technical Report 408,* Department of Statistics, Stanford University.

Germano, M., Piomelli, U., Moin, P. & Cabot, W. H. 1991 A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A.* **3**(7), 1760-1765.

Ghosal, S. & Moin, P. 1995 The basic equations of the large eddy simulation of turbulent flows in complex geometry. *J. Comp. Phys.***118**, 24-37.

Ghosal, S. 1993 On the large eddy simulation of turbulent flows in complex geometry. *Annual Research Briefs,* Center for Turbulence Research, NASA Ames/Stanford Univ. 111-128.

Ghosal, S. 1999 Mathematical and Physical Constraints on Large-Eddy Simulation of Turbulence. *AIAA J.* **37**(4), 425-433.

Lund, T. S. 1997 On the use of discrete filters for large eddy simulation. *Annual Research Briefs,* Center for Turbulence Research, NASA Ames/Stanford Univ. 83-95.

Marsden, A. L., Vasilyev, O. V. 1999 Commutative filters for LES on unstructured meshes. *Annual Research Briefs.* Center for Turbulence Research, NASA Ames/Stanford Univ. 389-402.

Perot, P. 2000 Conservation Properties of Unstructured Staggered Mesh Schemes. *J. Comp. Phys.***159,** 58-89.

Sweldens, W. 1996 The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.* **3**(2), 186-200.

Sweldens, W. 1997 The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.* **29**(2), 511-546.

Van der Ven, H. 1995 A family of large eddy simulation (LES) filters with nonuniform filter widths. *Phys. Fluids.* **7**(5), 1171-1172.

Vasilyev, O. V., Lund, T. S. & Moin, P. 1998 A General Class of Commutative Filters for LES in complex Geometries. *J. Comp. Phys.* **146,** 82-104.