Commutative filters for LES on unstructured meshes

By Alison L. Marsden AND Oleg V. Vasilyev[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 condition 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 (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 and shape at each point in space and obtain a discrete filter which satisfies this requirement regardless of the choice of the computational mesh.

Accomplishments

1. Commutation error 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

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

is done in the computational domain. Here, we extend the theory of commutative filters developed in Vasilyev (1998) to the physical domain. We begin by discussing the filtering in one-dimensional space and then extend it to three spatial dimensions.

1.1 Commutation error in one spatial dimension

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

$$\left[\frac{d\phi}{dx}\right] = \overline{\frac{d\phi}{dx}} - \frac{d\overline{\phi}}{dx}. \tag{1}$$

where overbar denotes the filtered quantity. The filtering operation is defined by

$$\overline{\phi}(x) = \frac{1}{\Delta(x)} \int_a^b G\left(\frac{x-y}{\Delta(x)}, x\right) \phi(y) dy, \tag{2}$$

where $\Delta(x)$ is the filter width and $G(\eta, x)$ is the location dependent filter function. With the change of variables $\eta = \frac{x-y}{\Delta(x)}$, (2) can be written as

$$\overline{\phi}(x) = \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} G(\eta, x)\, \phi(x - \Delta(x)\eta)\, d\eta. \tag{3}$$

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

$$\phi(x - \Delta(x)\eta) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) \eta^l \mathcal{D}_x^l \phi(x), \tag{4}$$

where $\mathcal{D}_x = d/dx$ is the derivative operator. This series was proven to be convergent in Vasilyev (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 (4) into (3) and changing the order of summation and integration, we have

$$\overline{\phi}(x) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) \phi^l(x) \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} \eta^l G(\eta, x) d\eta. \tag{5}$$

Defining the filter moment as

$$M^l(x) = \int_{\frac{x-b}{\Delta(x)}}^{\frac{x-a}{\Delta(x)}} \eta^l G(\eta, x) d\eta \tag{6}$$

and substituting (6) into (5) we obtain

$$\overline{\phi}(x) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) D_x^l \phi(x). \tag{7}$$

In the same manner as in Vasilyev (1998) we let

$$M^l(x) = \begin{cases} 1, & l = 0 \\ 0, & l = 1, \ldots, n-1. \end{cases} \tag{8}$$

With this definition we have

$$\overline{\phi} = \phi(x) + \sum_{l=n}^{\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) \mathcal{D}_x^l \phi(x). \tag{9}$$

The filtered quantity of the derivative is

$$\overline{\frac{d\phi}{dx}}(x) = \frac{d\phi}{dx} + \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \Delta^l(x) M^l(x) \mathcal{D}_x^{l+1} \phi(x). \tag{10}$$

The derivative of the filtered quantity is

$$\frac{d\overline{\phi}}{dx}(x) = \frac{d\phi}{dx} + \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \frac{d}{dx} \left(\Delta^l(x) M^l(x) \mathcal{D}_x^l \phi(x) \right). \tag{11}$$

Applying the chain rule to (11) and subtracting (11) from (10), we obtain an expression for the commutation error:

$$\left[\frac{d\phi}{dx} \right] = \sum_{l=n}^{+\infty} \frac{(-1)^l}{l!} \left\{ \frac{d}{dx} \left(\Delta^l(x) M^l(x) \right) \right\} \mathcal{D}_x^l \phi(x). \tag{12}$$

Using the properties in (8) it follows that

$$\frac{dM^l}{dx}(x) = 0 \quad \text{for } \mathrm{l} = 1, \ldots, \mathrm{n} - 1. \tag{13}$$

As a result, the local commutation error is

$$\left[\frac{d\phi}{dx} \right] = O\left(\Delta^n(x) \right), \tag{14}$$

provided that $d\Delta/dx = O(\Delta)$, which is true for most 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 $O\left(\Delta^{n+\gamma-1}\right)$.

1.2 Extension to three dimensions

The extension to three dimensions is quite straightforward. Let us consider a three-dimensional field $\phi(\mathbf{x})$, $(\mathbf{x} \equiv (x_1, x_2, x_3)^{\mathrm{T}})$ defined in a three-dimensional domain Ω. The filtering operation in three-dimensional space is defined by

$$\overline{\phi}(\mathbf{x}) = \frac{1}{\Delta_1(\mathbf{x})\Delta_2(\mathbf{x})\Delta_3(\mathbf{x})} \oint_\Omega G\left(\frac{x_1 - y_1}{\Delta_1(\mathbf{x})}, \frac{x_2 - y_2}{\Delta_2(\mathbf{x})}, \frac{x_3 - y_3}{\Delta_3(\mathbf{x})}, \mathbf{x} \right) \phi(\mathbf{y}) d^3\mathbf{y}. \tag{15}$$

The transformation $\eta_i = (x_i - y_i)/\Delta_i(\mathbf{x})$ maps the domain Ω to domain Ψ. With this change of variable, (15) can be rewritten as

$$\overline{\phi}(\mathbf{x}) = -\oint_{\Psi} G(\boldsymbol{\eta}, \mathbf{x})\, \phi(x_i - \Delta_i(\mathbf{x})\eta_i)\, d^3\boldsymbol{\eta}. \tag{16}$$

Taking the Taylor series expansion of ϕ as in the one-dimensional case, we have

$$\phi(x_1 - \Delta_1(\mathbf{x})\eta_1, x_2 - \Delta_2(\mathbf{x})\eta_2,\ x_3 - \Delta_3(\mathbf{x})\eta_3) = \sum_{l=0}^{+\infty} \frac{(-1)^l}{l!} \left(\sum_{m=1}^{3} \Delta_m(\mathbf{x})\eta_m \mathcal{D}_{x_m} \right)^l \phi(\mathbf{x}), \tag{17}$$

which can alternatively be written as

$$\phi(x_i - \Delta_i(\mathbf{x})\,\eta_i) = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) \eta_1^i \eta_2^j \eta_3^k \mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}), \tag{18}$$

where α_{ijk}^l are coefficients of the polynomial expansion

$$(a + b + c)^l = \sum_{i+j+k=l} \alpha_{ijk}^l a^i b^j c^k.$$

Substituting (18) into (16) and changing the order of summation and integration, we obtain

$$\overline{\phi}(\mathbf{x}) = -\sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) \left[\mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \right] \times \oint_{\Psi} \eta_1^i \eta_2^j \eta_3^k G(\boldsymbol{\eta}, \mathbf{x})\, d^3\boldsymbol{\eta}. \tag{19}$$

Defining the filter moment as before, we have

$$M^{ijk} = -\oint_{\Psi} \eta_1^i \eta_2^j \eta_3^k G(\boldsymbol{\eta}, \mathbf{x})\, d^3\boldsymbol{\eta}. \tag{20}$$

Then, substituting (20) into (19) into gives

$$\overline{\phi}(\mathbf{x}) = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}). \tag{21}$$

As in the one-dimensional case, we let

$$M^{ijk}(x) = \begin{cases} 1, & i, j, k = 0 \\ 0, & 0 < i + j + k < n. \end{cases} \tag{22}$$

Then from (21) we obtain

$$\overline{\phi}(\mathbf{x}) = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}). \quad (23)$$

Without loss of generality let us consider the commutation error between differentiation in the x_1 direction and filtering, $\left[\frac{\partial \phi}{\partial x_1}\right]$. The filtered value of the derivative is

$$\overline{\frac{\partial \phi}{\partial x_1}} = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \mathcal{D}_{x_1}^{i+1} \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \quad (24)$$

and the derivative of the filtered function is

$$\frac{\partial \overline{\phi}}{\partial x_1} = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \left\{ \frac{\partial}{\partial x_1} \left(\Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \right) \mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \right.$$
$$\left. + \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \mathcal{D}_{x_1}^{i+1} \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \right\}. \quad (25)$$

We now have an expression for the commutation error in three dimensions with a variable filter width

$$\left[\frac{\partial \phi}{\partial x_1}\right] = \sum_{l=0}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^{l-1}}{l!} \alpha_{ijk}^l \left[\frac{\partial}{\partial x_1} \left(\Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \right) \right] \times$$
$$\mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}). \quad (26)$$

Using the properties given in (22), it is easily shown that for smoothly varying meshes the local commutation error in three dimensions is given by

$$\left[\frac{\partial \phi}{\partial x_1}\right] = O\left(\Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) \right), \qquad i+j+k = n \quad (27)$$

so that commutation is satisfied to a desired order.

2. 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 (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 (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 (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 the 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 and 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

$$M^l = \sum_{k=1}^{N} w_k (x_k - x_0)^l \tag{28}$$

automatically satisfy the conditions (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 want,

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 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 it will later be extended to three dimensions in Section 3.3. Details on choosing which points to use in the filter are discussed in Sections 3.2 and 3.3.

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

$$P(x, y) = a_{00} + a_{10}(x - x_0) + a_{01}(y - y_0) \tag{29}$$

be a first order polynomial interpolant. Requiring that interpolant (29) goes through the data points f_i $(i = 1, \ldots, 3)$, we obtain the following set of linear equations

$$\begin{aligned} f_1 &= a_{00} + a_{10}(x_1 - x_0) + a_{01}(y_1 - y_0), \\ f_2 &= a_{00} + a_{10}(x_2 - x_0) + a_{01}(y_2 - y_0), \\ f_3 &= a_{00} + a_{10}(x_3 - x_0) + a_{01}(y_3 - y_0). \end{aligned} \tag{30}$$

Note that interpolant (29) 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

$$P(x_0, y_0) = w_1 f_1 + w_2 f_2 + w_3 f_3, \tag{31}$$

where w_i are the filter weights.

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

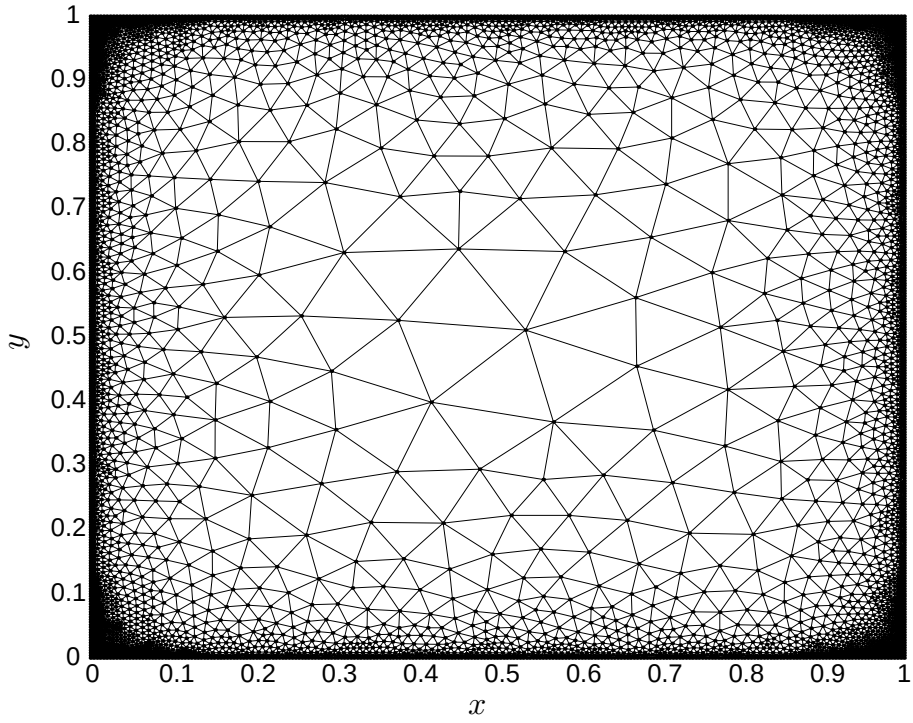

FIGURE 1. Example of mesh used for filter development.

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 - x_0 & y_1 - y_0 \\ 1 & x_2 - x_0 & y_2 - y_0 \\ 1 & x_3 - x_0 & y_3 - y_0 \end{pmatrix} \tag{32}$$

and

$$\mathbf{b} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \tag{33}$$

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

$$\mathbf{w} = \mathbf{b} \cdot \mathbf{A}^{-1}. \tag{34}$$

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 Section 3.3.

3. Implementation of commutative filters

3.1 Two-dimensional filters

In Section 2 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

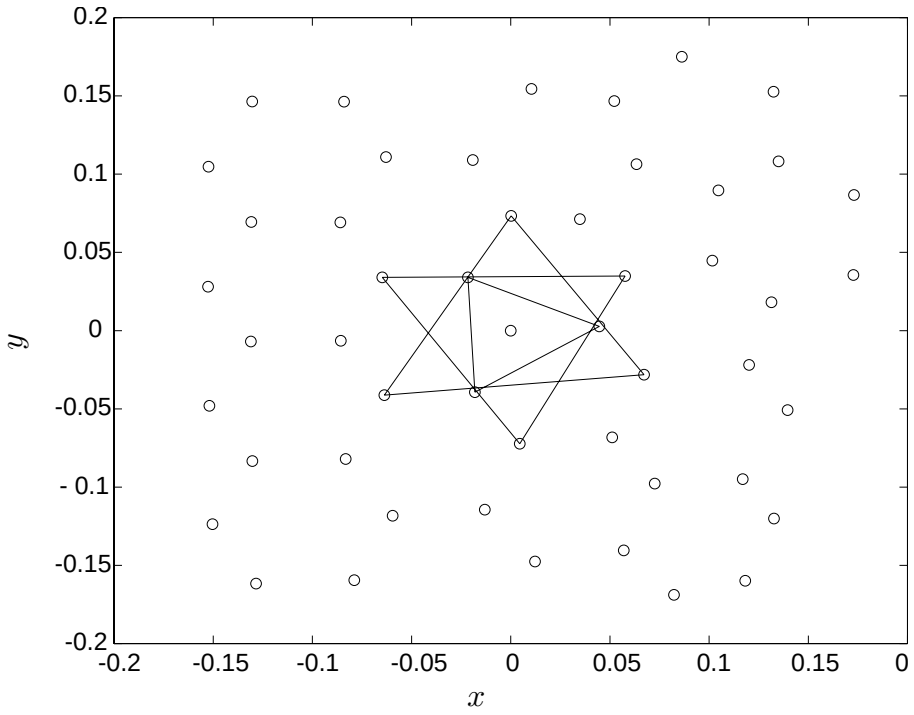

FIGURE 2. Example of filter constructed with triangles on an unstructured grid.

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 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 2 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. 3. 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 2 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.

3.2 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 3.1. The

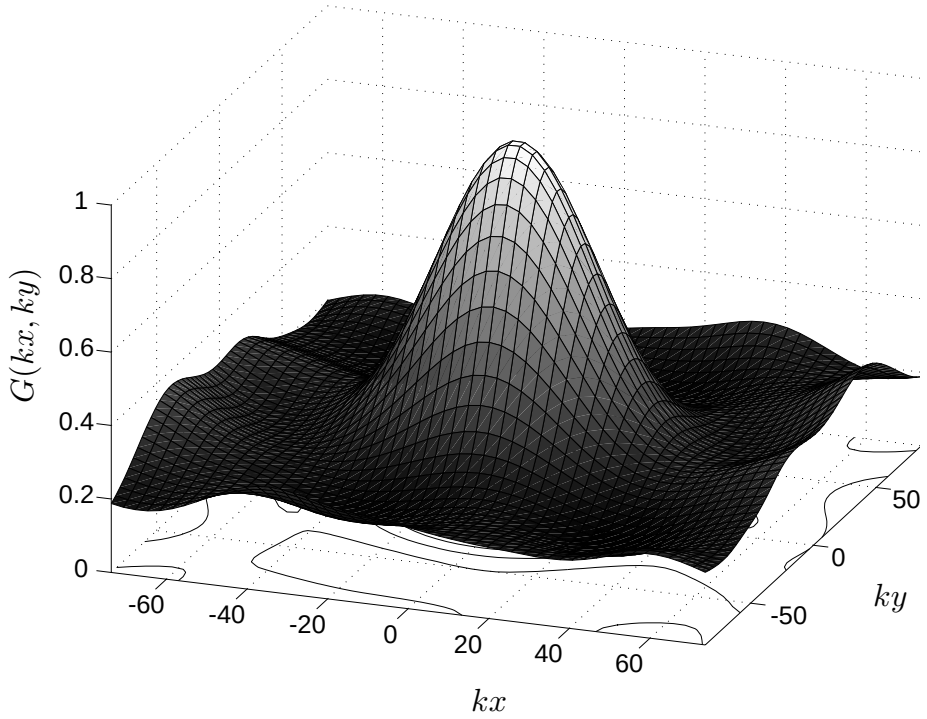

FIGURE 3. Typical transfer function shape.

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 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.

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. In the present formulation this involves defining a box around the central point and sorting mesh points to pick out those which lie in the box. However, for cost reasons the final algorithm will take advantage of the mesh connectivity for this purpose. Figure 1 shows an example mesh which was used in testing and developing the filtering scheme.

Having sorted the surrounding mesh points into a "box", the next step is to calculate the distance to each point in the box 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

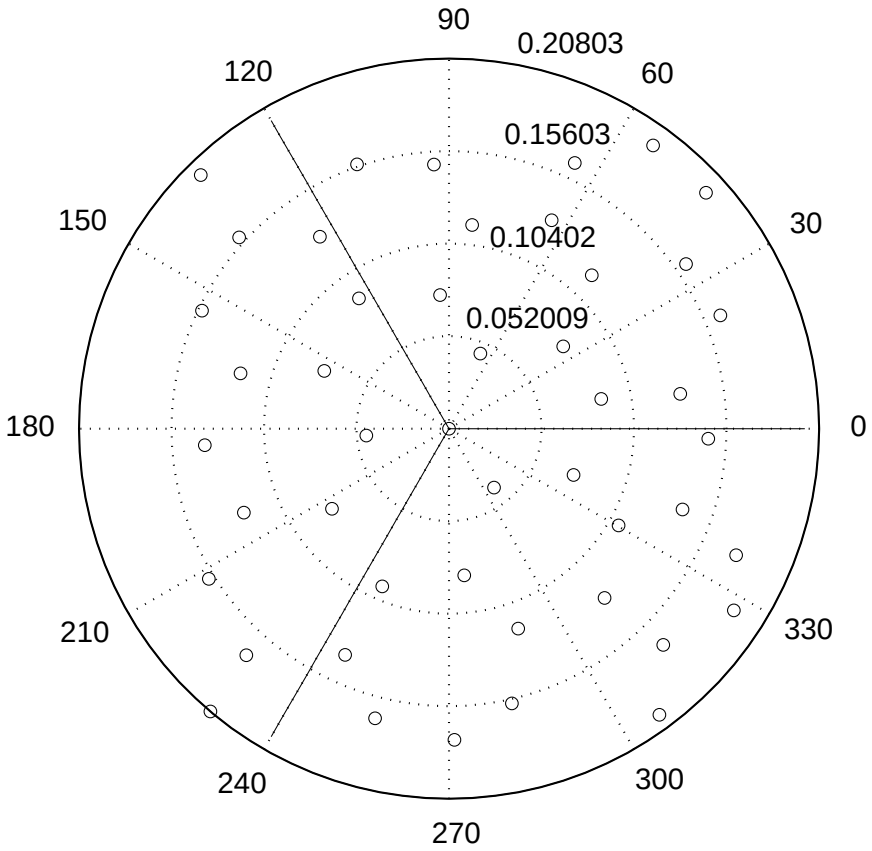

FIGURE 4. Mesh nodes sorted by angle.

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 try another set of three points, continuing until the desired number of triangles has been found.

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

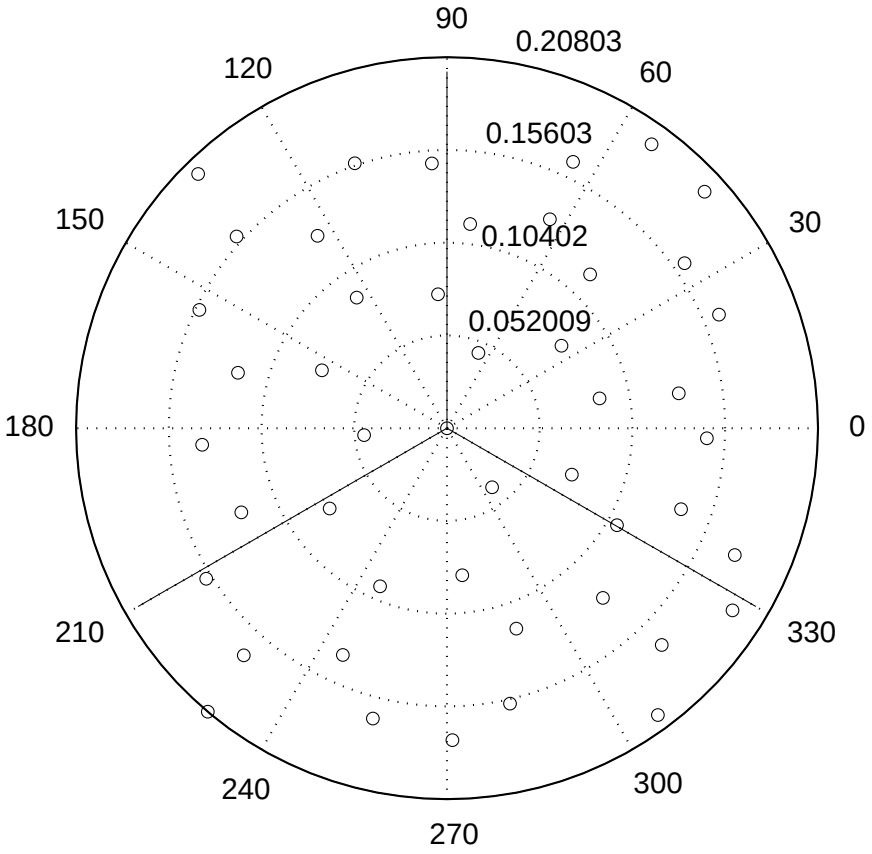

FIGURE 5. Mesh nodes sorted by angle, rotated from Fig. 4.

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 3.1. 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.

3.3 Three-dimensional filters: An extension

The extension of the filtering procedure outlined in Section 3.2 to three dimensions is quite straightforward. While three points are required in two dimensions for commutation, four points are required in three dimensions as shown by the following. For reconstruction of a first order polynomial we have four coefficients.

$$\mathbf{f} = a_{000} + a_{100}\mathbf{x} + a_{010}\mathbf{y} + a_{001}\mathbf{z} \tag{35}$$

The base filter shape now becomes a tetrahedron instead of a triangle, but filter

construction algorithm is completely analogous to the two-dimensional method presented in Section 3.2 Four zones are created in three dimensions, and points in each zone are ordered by distance from the central point. Tetrahedrons are systematically formed by taking a point from each zone starting with the closest point in each. To determine whether a given tetrahedron meets the criterion for use in the filter, we need to know if the central point is inside the tetrahedron and if it lies far enough away from the sides. Analogous to the two-dimensional case, a line is drawn from the central point to each vertex to create four smaller tetrahedrons. If the point is inside, the volume of these tetrahedrons will equal the volume of the larger tetrahedron. Once this condition is met, the next check is that none of the smaller tetrahedrons have a volume which is too great a percentage of the larger tetrahedron.

Once the desired number of tetrahedrons has been reached, a rotation is performed, and a new set is found as in the two-dimensional case. The set of tetrahedrons whose collective weights are closest to one fourth is then chosen to make up the final filter.

4. Conclusions and future plans

A method of constructing commutative filters for unstructured LES has been developed and demonstration of the commutative properties is currently underway. 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.

Acknowledgments

The authors are grateful to Krishnan Mahesh for many helpful discussions and comments.

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. 1999 Mathematical and physical constraints on large-eddy simulation of turbulence. *AIAA J.* **37**(4)**,** 425-433.

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

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.

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

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

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

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.