Journal of Computational Physics **175,** 584–603 (2002)
doi:10.1006/jcph.2001.6958, available online at http://www.idealibrary.com on IDEAL®

Construction of Commutative Filters for LES on Unstructured Meshes

Alison L. Marsden,* Oleg V. Vasilyev,† and Parviz Moin*

*Center for Turbulence Research, Stanford University, Stanford, California 94305; and †Department of Mechanical and Aerospace Engineering, University of Missouri, Columbia, Missouri 65211
E-mail: *amarsden@stanford.edu, †vasilyevo@missouri.edu

Received January 15, 2001; revised July 2, 2001

A method of constructing discrete filters for large eddy simulation of turbulent flows on unstructured meshes is presented. The commutation error between differentiation and filtering can be made arbitrarily small with these filters. The filtering method is applied to various test cases to demonstrate commutation. An extension to three dimensions and implementation into an unstructured solver for LES are discussed. © 2002 Elsevier Science (USA)

1. INTRODUCTION

The application of large eddy simulation (LES) to flows with increasingly complex geometry necessitates the extension of the LES technique 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 form 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. A general discussion of filtering and commutation error as applied to LES is presented by Geurts and Leonard [10]. In that work, the authors stress that the commutation error in LES should be the subject of further study in order to apply LES to complex geometries. In addition, Van der Ven [5] 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. A class of discrete commutative filters was developed by

0021-9991/02 $35.00

Vasilyev *et al.* [19] for use on nonuniform structured meshes. Their formulation uses a mapping function to perform the filtering in the computational domain. While this type of mapping may not be possible for the unstructured case, the theory developed in [19] 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 filter profile 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.* [19] developed a general theory of discrete filtering in arbitrarily complex geometries. With the use of a mapping function, the filtering was done in the computational domain. Here, we extend the theory of commutative filters developed in [19] to the physical domain. We begin by discussing filtering in one-dimensional space and then extend it to three spatial dimensions.

2.1. *Commutation Error in One Spatial Dimension*

In this section, we follow the development in [19]; however, we begin with the definition of the filtering operator in physical space, and we do not transform to the mapped space. An operator to measure commutation error is defined as follows. Given a function $\phi(x)$, the commutation error is

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

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

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

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

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

Taking the Taylor series expansion of $\phi(x - \Delta(x)\eta)$ in powers of $\Delta(x)$ 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 [19] for the case of uniform Δ by assuming that the Fourier spectrum did not include

wavenumbers higher than some finite cutoff wavenumber $k_{\max}$. The proof is analogous to the case of varying $\Delta(x)$ and, using the same assumptions, the radius of convergence is considered to be infinite. Substituting (4) into (3) and changing the order of summation and integration, we have

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

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

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

In the same manner as in [19] 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

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

The filtered derivative of the function is

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

The derivative of the filtered quantity is

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

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

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

Using the properties in (8) it follows that

$$\frac{d}{dx} (\Delta^l(x) M^l(x)) = 0 \qquad \text{for } l = 0, \ldots, n-1. \tag{13}$$

As a result, the local commutation error is

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

provided that $d\Delta/dx = O(\Delta)$, which is true if the filter width varies smoothly. For rapidly changing filter width, $d\Delta/dx$ is $O(\Delta^\gamma)$, $\gamma < \mathbf{1}$, which results in lowering the order of the commutation error to $O(\Delta^{n+\gamma-1})$.

2.2. *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

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

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

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

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

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

Defining the filter moment as before we have

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

Then, substituting (20) into (19) gives

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

As in the one-dimensional case, we let

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

Using the properties given in (22), Eq. (21) becomes

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

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

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

and the derivative of the filtered function is

$$\begin{aligned} \frac{\partial \bar{\phi}}{\partial x_1} = \frac{\partial \phi}{\partial x_1} + \sum_{l=n}^{+\infty} \sum_{i+j+k=l} \frac{(-1)^l}{l!} \alpha_{ijk}^l \Bigg\{ & \frac{\partial}{\partial x_1} \left(\Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \right) \mathcal{D}_{x_1}^i \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \\ & + \Delta_1^i(\mathbf{x}) \Delta_2^j(\mathbf{x}) \Delta_3^k(\mathbf{x}) M^{ijk}(\mathbf{x}) \mathcal{D}_{x_1}^{i+1} \mathcal{D}_{x_2}^j \mathcal{D}_{x_3}^k \phi(\mathbf{x}) \Bigg\}. \end{aligned} \tag{25}$$

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

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

from which it easily follows that for a smoothly varying filter width, the local commutation error in three dimensions is given by

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

so that commutation is achieved to a desired order.

3. CONSTRUCTION OF DISCRETE COMMUTATIVE FILTERS

The filters developed by Vasilyev *et al.* [19] were constructed by applying the necessary number of constraints to the filter weights to achieve both commutation and an acceptable

filter shape in wavenumber space. The constraints imposed on the filter weights were as follows. The zeroth moment should be 1, a specified number (order of commutation error) of higher moments should be 0, 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 done in [19]. 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 manually adjust the filter constraints at each mesh point.

An initial formulation for filter construction on an unstructured mesh used the ideas presented in [19] generalized to physical space. Given a point where a filtered value was needed, a set of neighboring points was chosen to construct the filter. Then, constraints were applied directly on the filter moments and shape to determine the filter weights. This procedure followed directly from [19]. Two problems arose in implementing this method. First, it was found that in the case of a nonuniform point distribution such as an unstructured mesh, the shapes of the resulting filters were highly unpredictable. 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 [6] and the theory of second generation wavelets [2, 17, 18]. To illustrate the idea of construction of discrete filters based on polynomial interpolation, we consider a one-dimensional example. Suppose we have a set of N unevenly spaced grid points x_i $(i = 1, 2, \ldots, N)$ and the values of the function f_i are known at these points. Suppose we desire a filtered value at an arbitrary point x_0. 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 use these weights, w_i, as the weights of the corresponding discrete filter, then this filter will have the unique property that when applied to a polynomial of degree less than $N-1$ it will not change the 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 0 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 Nth-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 Nth-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 of 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 of three points surrounding the point (x_0, y_0) 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). With this method, the same number of points are used in the filter for any point on the mesh.

Because every filter using this method is a triangle, the shape of the resulting filter in wavenumber space is very well defined. The transfer function of an equilateral triangle filter has a symmetric, low-pass-filter shape with a well-defined peak. Since any triangle can be obtained from an equilateral triangle by a linear transformation, we are guaranteed that any triangular filter will retain these desirable characteristics.

The method for finding the filter weights using a triangle in two dimensions is presented here but it will be extended to three dimensions in Section 4.3. Details on choice of filter points are discussed in Sections 4.1 and 4.3.

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 let (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 the interpolant (29) goes through the data points $f_i (i = 1, 2, 3)$ leads to a set of linear equations for the coefficients a_{00}, a_{10}, a_{01}. 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

$$P(x_0, y_0) = w_1 f_1 + w_2 f_2 + w_3 f_3, \tag{30}$$

where w_i are the filter weights.

The weights can be simply calculated from the equation $\mathbf{Aw} = \mathbf{b}$, where

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 - x_0 & y_1 - y_0 \\ 1 & x_2 - x_0 & y_2 - y_0 \\ 1 & x_3 - x_0 & y_3 - y_0 \end{pmatrix} \tag{31}$$

and

$$\mathbf{b} = (1\ 0\ 0). \tag{32}$$

From the above, we have weights making 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 4.3.

For this linear case, the weights can be found analytically by inverting the matrix **A**. Realistically, we have no need to extend this method to higher order before further development of higher order nummerical schemes for unstructured meshes. However, for completeness we stress that for higher order Vandermonde matrices, it is well known that the condition number can grow exponentially with the order of the matrix [8]. There are two numerical packages available which deal with multivariate interpolation and could be adapted for use with higher order filters. A nice overview of polynomial interpolation is given in [7] with some discussion of these routines. Further details can be found in [4] and [3].

4. IMPLEMENTATION OF COMMUTATIVE FILTERS

4.1. *Two-Dimensional Filters*

In Section 3 we constructed 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. Using these triangular filters as a basis, we will construct commutative filters that combine multiple triangles into one filter and allow for a variable filter width.

Although a single triangular filter satisfies the commutation property, it is undesirable because it offers no flexibility in filter width or shape. In Section 3 we showed that triangular filters have the desired low-pass-filter shape because they are simply linear transformations of an equilateral triangle. To take advantage of this property 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 desirable transfer function shape while ensuring that the resulting filter will satisfy commutation to the same order as the basis triangles. Because of the predictable transfer function shape, we are also guaranteed that the filter properties will be smoothly varying in space.

Figure 1 shows an example of a 2-D filter constructed from three triangles. The corresponding transfer function,

$$\hat{G}(k_x, k_y) = \sum_{l=1}^{N} w_l e^{i[k_x(x_l - x_0) + k_y(y_l - y_0)]}, \tag{33}$$

is shown in Fig. 2 and has the desired low pass characteristics. 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 a transfer function with a well-defined peak and low-pass-filter shape.

4.2. *Implementation in Two Dimensions*

We now 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

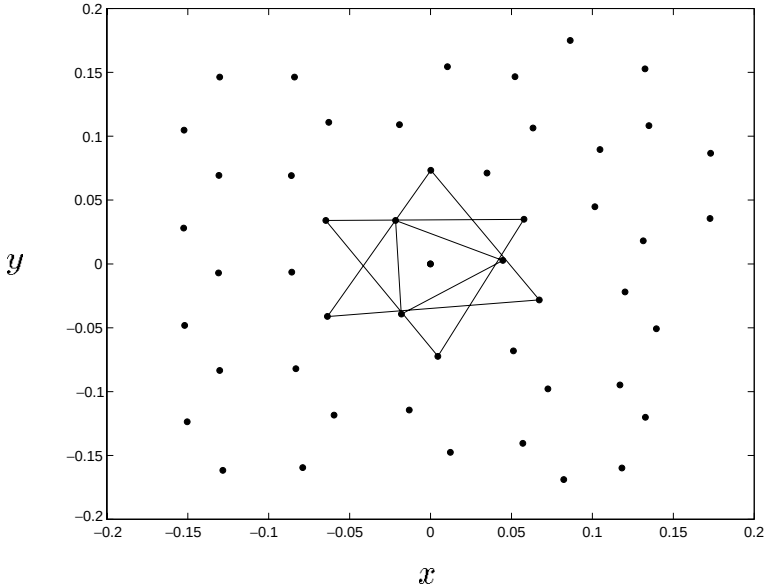

FIG. 1. Example of filter constructed with triangles on an unstructured grid.

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 minimum number of triangles needed for a symmetric low-pass-filter shape on a regular 2-D unstructured mesh is three. Because of this property, choosing filters with three triangles also ensures that the filter width will vary smoothly. We will therefore use three triangles for the present work.

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

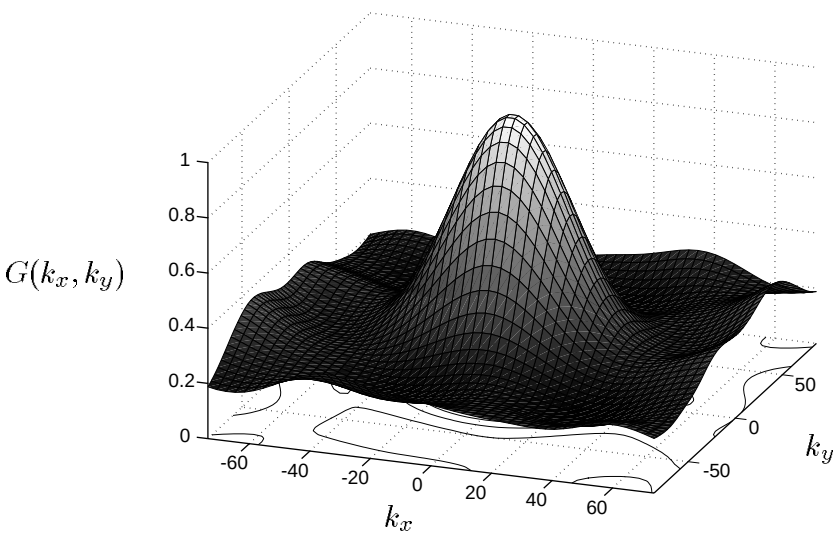

FIG. 2. Transfer function corresponding to filter in Fig. 1.

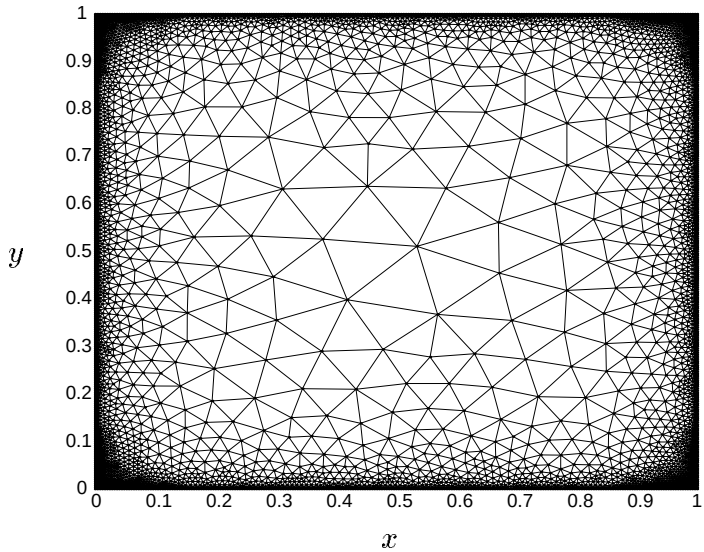

FIG. 3. Example of mesh used for filter development.

apply the filter to the discrete data. Figure 3 shows an example of an unstructured mesh used in testing the algorithm that chooses filter points.

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 arrived at. 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.

Having found a group of surrounding mesh points, we 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° each. The zones are created to ensure that the chosen points have a near symmetric distribution of angles about the central point. Figure 4 shows these three zones. Within each zone, the points are sorted by their distances from the central point.

Triangles are systematically formed by taking a point from each zone, starting with the closest point in each, and then testing if the chosen triangle meets the criteria 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 criteria 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 criteria involve drawing lines from the central point to each vertex of the triangle to form three subtriangles. If the summed area of the subtriangles exceeds that of the larger triangle, the central point is outside. If the area of any of the subtriangles 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.

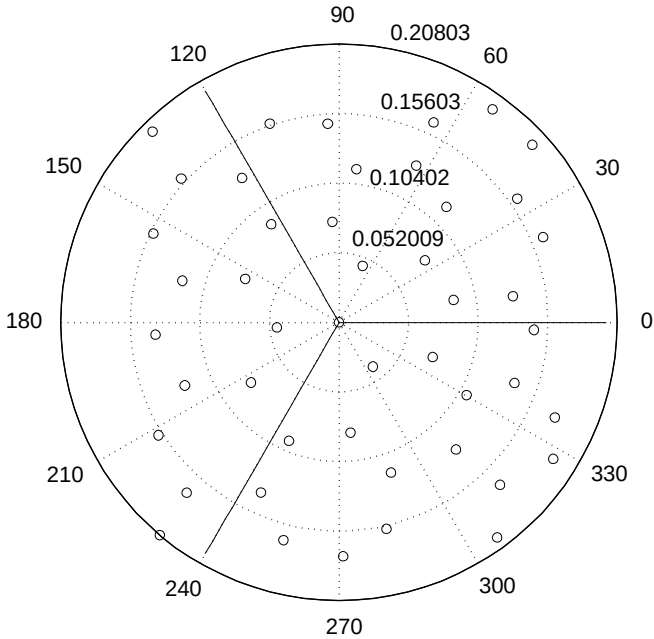

FIG. 4. Mesh nodes sorted by angle.

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 considered 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 weight is 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 a complete filter as described in Section 4.1. Flexibility is gained by applying the same filter again with different triangle weights, β_i, to achieve a desirable 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 subgrid scale model.

4.3. *Three-Dimensional Filters: An Extension*

The extension of the filtering procedure outlined in Section 4.2 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:

$$\mathbf{f} = a_{000} + a_{100}\mathbf{x} + a_{010}\mathbf{y} + a_{001}\mathbf{z}. \tag{34}$$

The base filter shape now becomes a tetrahedron instead of a triangle, but the filter construction algorithm is completely analogous to the two-dimensional method presented

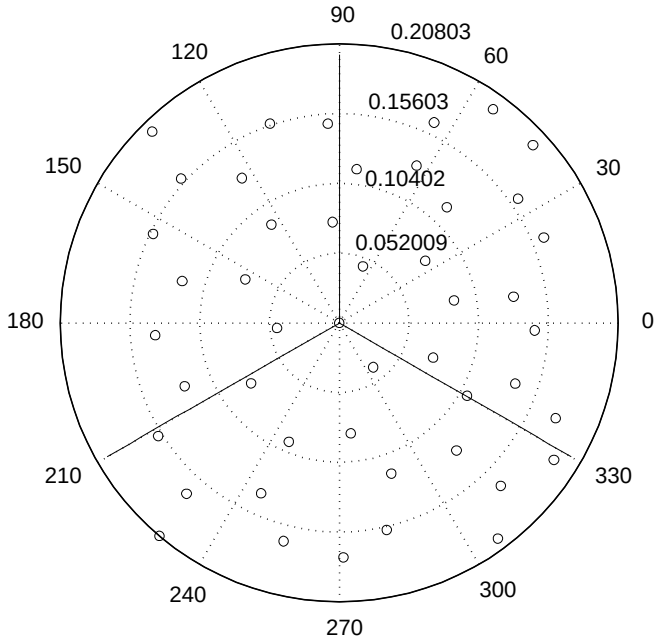

FIG. 5. Mesh nodes sorted by angle, rotated from Fig. 4.

in Section 4.2. Four zones are created in three dimensions, and points in each zone are ordered by their distances 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 determine 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 weight is closest to one-fourth is then chosen to make up the final filter.

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 desirable to use an exponentially increasing filter width since the scale of eddies increases with distance away from the wall. In two dimensions, 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 integral

$$M_2 = \int_0^{2\pi} \int_0^{\Delta} r^2 \cos^2\theta \frac{1}{\pi \Delta^2} r\, dr\, d\theta, \tag{35}$$

where r and θ are polar coordinates centered at the filter point. Evaluating this integral we have a relation between the second moment and the filter width:

$$\Delta = \sqrt{4M_2}. \tag{36}$$

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 in turn be used to find the values of β which will approximately result in the desired value of Δ. The values of β are chosen by solving the following set of equations using the least squares method with constraints on the values of β such that

$$\beta_0 = 1/2 \tag{37}$$

$$\beta_1 + \beta_2 + \beta_3 = 1, \tag{38}$$

where β_0 is the value assigned to the central point and β_1, β_2, β_3 are assigned to the three triangles in the filter.

The target second moment values in x, y, and xy are M^{02}, M^{20}, and M^{11} respectively,

$$\begin{aligned} M^{02} &= m_1^{02}\beta_1 + m_2^{02}\beta_2 + m_3^{02}\beta_3 \\ M^{20} &= m_1^{20}\beta_1 + m_2^{20}\beta_2 + m_3^{20}\beta_3 \\ M^{11} &= m_1^{11}\beta_1 + m_2^{11}\beta_2 + m_3^{11}\beta_3, \end{aligned} \tag{39}$$

where m_i^{02}, m_i^{20}, and m_i^{11} are the moments of the individual triangles.

For higher order filters, the definition of the filter width cannot be based on the second moment, since it is zero by definition. However, we can use the more general definition given by Lund in [14] where the filter width is based on the second moment of the transfer function.

6. DEMONSTRATING COMMUTATION

To validate that the filters developed commute to the desired order, a series of numerical tests were performed. Since each filter can be constructed as a linear combination of several triangle filters, it is sufficient to demonstrate commutation for single triangle filters. 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 resulting truncation error is very nonuniform. A more directionally symmetric operation to perform on an unstructured mesh is to calculate the curl at each mesh point. In recently developed conservative schemes for incompressible flows on unstructured meshes, the nonlinear terms are written in the rotational form, involving the curl of the velocity vector [15]. It is relatively straightforward to demonstrate that the commutative properties of the filtering and curl operators are the same as those discussed in Section 2.

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

$$[\nabla \times \mathbf{u}] = \nabla \times \overline{\mathbf{u}} - \overline{\nabla \times \mathbf{u}}, \tag{40}$$

where **u** is the prescribed vector field. The trivial case of the linear function $f = ax + by + c$ was used to verify that the filtering operation does not change this function within machine zero. With this check complete, the commutation and truncation errors were calculated on a series of consecutively finer meshes, using the velocity field describing a vortex in a box,

$$u = -\cos(\beta_1 x + \phi_1)\sin(\beta_2 y + \phi_2) \tag{41}$$

$$v = \sin(\beta_1 x + \phi_1)\cos(\beta_2 y + \phi_2). \tag{42}$$

A plot of velocity vectors 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,

$$C_{\text{exact}} = (\beta_1 + \beta_2)[\cos(\beta_1 x + \phi_1)\cos(\beta_2 y + \phi_2)], \tag{43}$$

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. (40). The second term in (40) 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.

The curl is found by dividing the circulation by the cell area. Both high- and low-order integration schemes were used to calculate the circulation. Filters with one triangle are sufficient to demonstrate commutation.

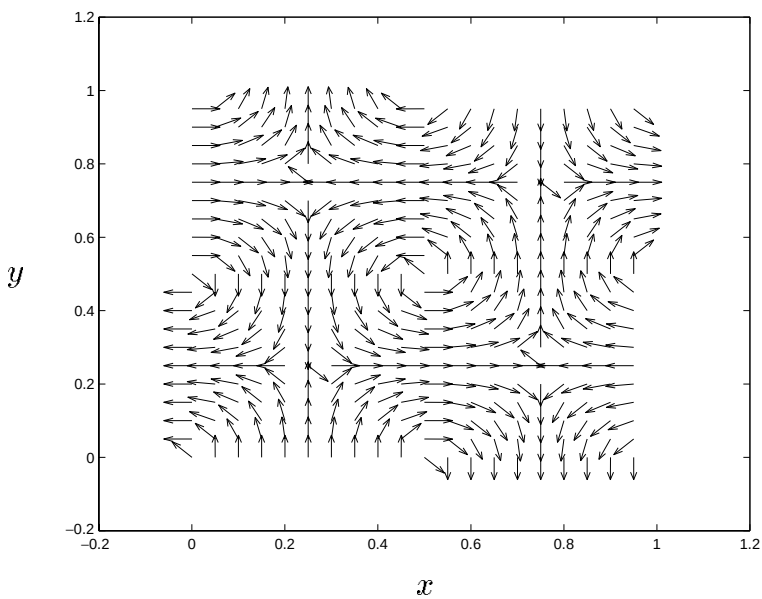

FIG. 6. Velocity vectors for vortices in a box.

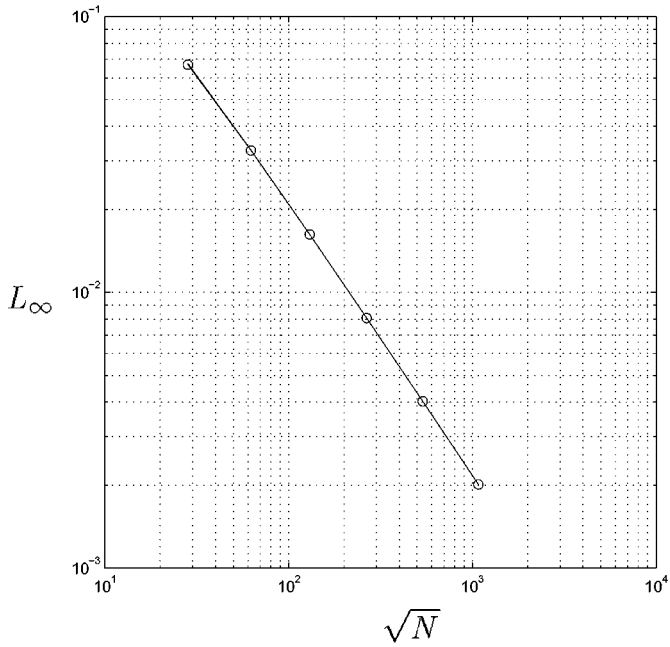

FIG. 7. Normalized L_∞ commutation and truncation errors of curl, using the low-order integration method. —, truncation error, ○, commutation error.

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 the error in the entire operation is first-order. 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 curve is equal to the order of accuracy. All errors have been normalized with the maximum value of the curl. A high-order integration method was developed for a uniform symmetric unstructured mesh. The integration method first requires interpolating to the vertices from 12 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 $(1/6, 2/3, 1/6)$. The truncation and commutation errors using the high-order integration method are shown in Fig. 8. Second-order convergence of both commutation and truncation errors are obtained. In addition, the magnitude of the commutation error is consistently less than the truncation error. However, the convergence of the truncation error is second order, not third order. This confirms that the curl operator is second order regardless of integration scheme due to the division by area.

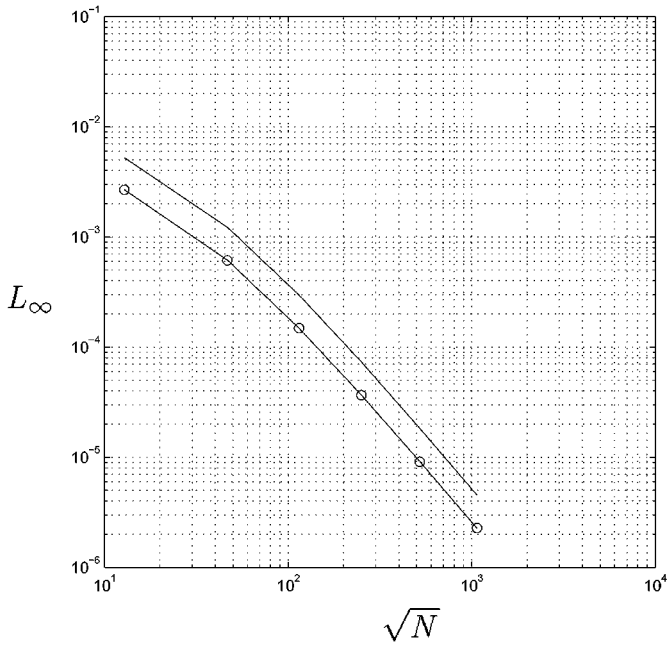

FIG. 8. Normalized L_∞ commutation and truncation errors of curl, using the high-order integration method. —, truncation error, ○, commutation error.

To confirm that the commutation and truncation errors are in fact independent, we 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.

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. In this case, the order of integration is high enough that the commutation error is not affected, and we can confirm that it has second-order convergence. The results for this case are shown in Fig. 10 and they confirm that the filtering method has the desired convergence properties. Comparing the cases with the high-order method using both curl and circulation, we confirm that the division by area in the curl operation reduces the truncation error convergence to second-order. Using the low-order integration scheme, with 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 decreased a full order of magnitude. In the figure, Γ_{low} refers to circulation found with the low-order integration method, and Γ_{high} refers to circulation found with the high-order

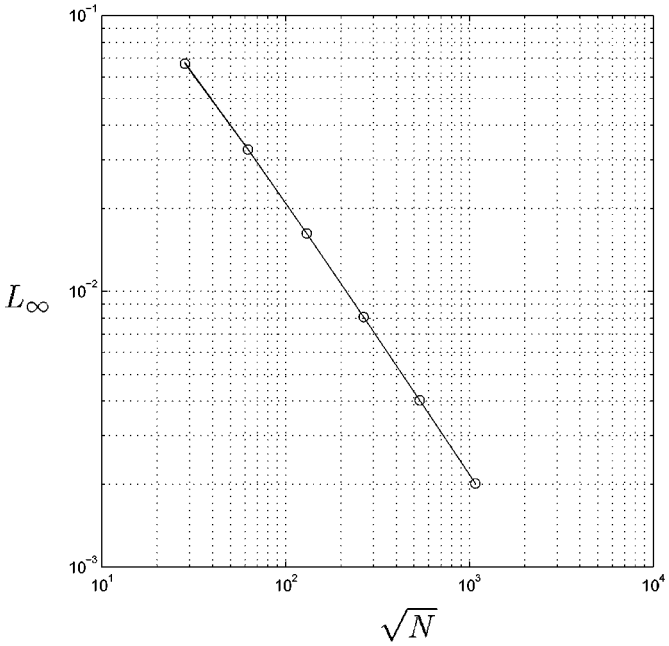

FIG. 9. Normalized L_∞ commutation and truncation errors of circulation, using the low-order integration method. —, truncation error, ○, commutation error.

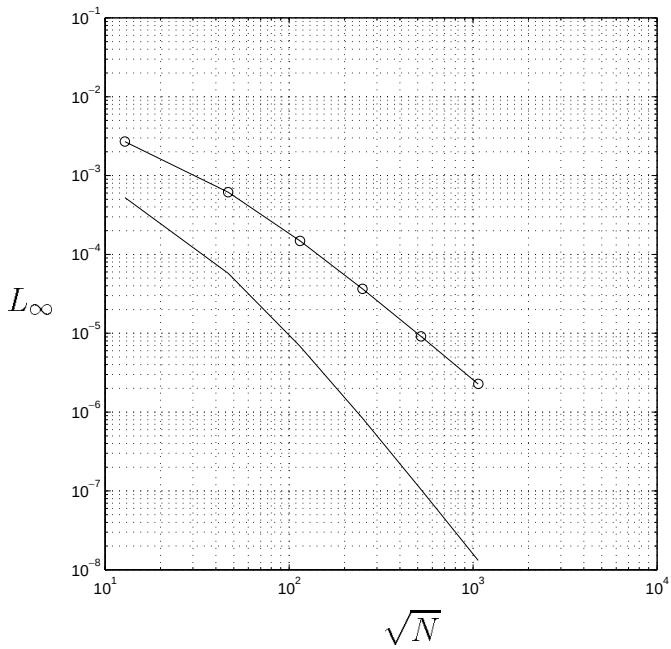

FIG. 10. Normalized L_∞ commutation and truncation errors of circulation, using the high-order integration method. —, truncation error, ○, commutation error.

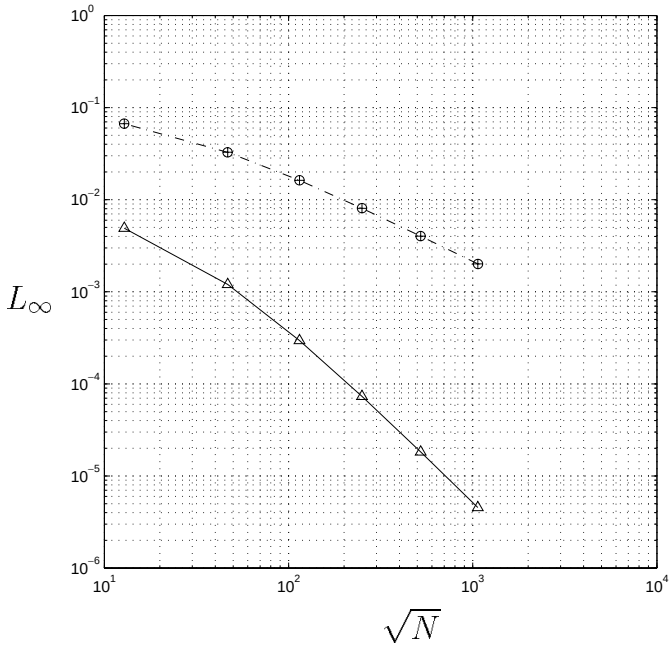

FIG. 11. Comparison of filtered and unfiltered normalized L_∞ truncation errors. ○, $\Gamma_{\text{low}}(V) - \Gamma_{\text{high}}(V)$; +, $\Gamma_{\text{low}}(\bar{V}) - \Gamma_{\text{high}}(\bar{V})$; Δ, $\bar{\Gamma}_{\text{low}}(V) - \bar{\Gamma}_{\text{high}}(V)$; ––, $\Gamma_{\text{low}}(\bar{V}) - \bar{\Gamma}_{\text{low}}(V)$.

integration method. The overbar indicates the filtering operation. These results explain why the commutation error is contaminated and the values of the commutation error and truncation error using the low-order scheme are nearly identical, as seen in Fig. 7 and 9. These tests confirm that the filtering method has a second-order commutation error, as predicted in Section 3. Multiple triangle filters are guaranteed to share this convergence property because they are simply a linear combination of single triangle filters. Using multiple triangle filters has the advantage of giving the user control over the filter width. In addition, even though it was necessary to use a uniform unstructured mesh for the numerical tests, we fully expect that the filtering method used will have similar convergence properties for any unstructured mesh since the filter varies from point to point by definition.

7. CONCLUSIONS

A method of constructing commutative filters for unstructured LES has been developed and validated. The method is intended for use in unstructured mesh flow solvers using the large eddy simulation technique. The convergence tests performed confirm that the filtering method leads to a second-order commutation error, and it 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 places 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 for any mesh. In addition, the filters presented have a low-pass-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 for use in conjunction with higher order schemes. For example, if a third-order finite difference scheme is to be used, the polynomial interpolant would have to be second-order, requiring six neighboring points in two dimensions. We should stress that the filtering mehtod can be extended to arbitarily high, but finite, order. However, there is nothing to be gained by extending the order beyond that of the numerical scheme.

ACKNOWLEDGMENTS

The authors are grateful to Krishnan Mahesh for many helpful discussions and comments. This work was funded in part by the Department of Energy's ASCI Program.

REFERENCES

1. D. Carati and E. V. Eijnden, On the self-similarity assumption in dynamic models for large eddy simulations, *Phys. Fluids* **9**(7), 2165 (1997).
2. I. Daubechies and W. Sweldens, Factoring wavelet transforms into lifting steps, *J. Fourier Annal. Appl.* **4**(3), 245 (1998).
3. C. de Boor, Computational aspects of multivariate polynomial interpolation: Indexing the coefficients, *Adv. Comput. Math.* **12**(4), 289 (2000).
4. C. de Boor and A. Ron, Computational aspects of polynomial interpolation in several variables, *Math. Comput.* **58**, 705 (1992).
5. H. Van, der Ven, A family of large eddy simulation (LES) filters with nonuniform filter widths, *Phys. Fluids* **7**(5), 1171 (1995).
6. D. L. Donoho, *Interpolating Wavelet Transforms*, Technical Report 408 (Department of Statistics, Stanford University, 1992).
7. M. Gasca and T. Sauer, Polynomial interpolation in several variables, *Adv. Comput. Math.* **12**(4), 377 (2000).
8. W. Gautschi, How (un)stable are vandermonde systems, in *Asymptotic and Computational Analysis*, edited by R. Wong, Lecture Notes in Pure and Applied Mathematics (Marcel Dekker, New York, 1990), Vol. 124, pp. 193–210.
9. M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, A dynamic subgrid-scale eddy viscosity model, *Phys. Fluids A* **3**(7), 1760 (1991).
10. B. J. Geurts and A. Leonard, *Is* LES ready for complex flows?, in *Closure Strategies for Turbulent and Transitional Flows*, edited by N. Sandham and B. Launder (Cambridge Univ. Press, Cambridge, UK, to appear), Chap. 25.
11. S. Ghosal, *On the Large Eddy Simulation of Turbulent Flows in Complex Geometry*, Annual Research Briefs (Center for Turbulence Research, City? 1993), pp. 111–128.
12. S. Ghosal, Mathematical and physical constraints on large-eddy simulation of turbulence, *AIAA J.* **37**(4), 425 (1999).
13. S. Ghosal and P. Moin, The basic equations of the large eddy simulation of turbulent flows in complex geometry, *J. Comput. Phys.* **118**, 24 (1995).
14. T. S. Lund, *On the Use of Discrete Filters for Large Eddy Simulation*, Annual Research Briefs (Center for Turbulence Research, City? 1997), pp. 83–95.

15. Blair Perot, Conservation properties of unstructured staggered mesh schemes, *J. Comput. Phys.* **159**, 58 (2000).
16. Heinz Rutishauser, *Lectures on Numerical Mathematics* (Birkhäuser, Boston, 1990).
17. W. Sweldens, The lifting scheme: A custom-design construction of biorthogonal wavelets, *Appl. Comput. Harmon. Anal.* **3**(2), 186 (1996).
18. W. Sweldens, The lifting scheme: A construction of second generation wavelets, *SIAM J. Math. Anal.* **29**(2), 511 (1997).
19. O. V. Vasilyev, T. S. Lund, and P. Moin, A general class of commutative filters for LES in complex geometries, *J. Comput. Phys.* **146**, 82 (1998).