INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2001; **35**: 93–116

A stabilized finite element method for the incompressible Navier–Stokes equations using a hierarchical basis

Christian H. Whiting[a] and Kenneth E. Jansen[b,*]

[a] *Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, U.S.A.*
[b] *Department of Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy, NY, U.S.A.*

SUMMARY

Stabilized finite element methods have been shown to yield robust, accurate numerical solutions to both the compressible and incompressible Navier–Stokes equations for laminar and turbulent flows. The present work focuses on the application of higher-order, hierarchical basis functions to the incompressible Navier–Stokes equations using a stabilized finite element method. It is shown on a variety of problems that the most cost-effective simulations (in terms of CPU time, memory, and disk storage) can be obtained using higher-order basis functions when compared with the traditional linear basis. In addition, algorithms will be presented for the efficient implementation of these methods within the traditional finite element data structures. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: finite element method; hierarchical basis functions; Navier–Stokes equations

1. INTRODUCTION

Over the last two decades, stabilized finite element methods have grown in popularity, especially for fluid dynamics applications. Starting with the streamline upwind/Petrov–Galerkin (SUPG) method of Brooks and Hughes [1] through the work of Hughes *et al.* [2] on the Galerkin/least squares (GLS) method and up to recent work on multiscale methods of Hughes [3] and related work on residual-free bubbles by Rússo [4] and Brezzi *et al.* [5], a number of stabilized formulations have been proposed. A key feature of stabilized methods is that they have been proven (for relevant model problems) to be stable and to attain optimal convergence rates with respect to the interpolation error (see Hughes *et al.* [2] and Franca *et al.* [6]). This implies that as the polynomial order of the underlying finite element space is increased, the error in the numerical solution decreases at the same rate as the interpolation error. The present work extends the work of Whiting *et al.* [7] to the incompressible Navier–Stokes equations.

* Correspondence to: Department of Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180-3590, U.S.A.

Received 16 June 1999
Revised 24 February 2000

The goal of the present work is the use of hierarchical basis functions as a means to attain more accurate and cost-effective finite element simulations of complex turbulent flows. Previous experience suggests that stabilized methods can be successfully used to compute turbulence (see Jansen *et al.* [8] for an application to Reynolds-averaged Navier–Stokes (RANS) simulations and Jansen [9] for an application to direct numerical simulations (DNS) and large eddy simulations (LES)). These methods have been demonstrated by Jansen [9] to be minimally dissipative by studying the growth of a small disturbance in channel flow. The real pacing item in the simulations of Jansen [9] was the cost of the finite element simulation rather than the ability of a stabilized method to compute turbulence. This has led to a search for a more cost effective method, while remaining within the general stabilized finite element framework.

It is hoped that this new approach will enable simulations of fluid dynamics problems that are not currently feasible due to current cost restrictions. This work presents a step toward this more ambitious goal by applying the hierarchical basis to simpler, well-understood problems, where their cost effectiveness may be more readily quantified. We have chosen a stabilized finite element formulation based on the formulation of Taylor *et al.* [10] for incompressible flows and have extended it to accommodate higher-order basis functions. This formulation has been proven to be robust and accurate for turbulent simulations using linear basis functions. This, combined with the higher-order accuracy that stabilized methods have been shown to attain, has led us to select this formulation.

The basis functions are defined using the rich mesh data structure of Beall and Shephard [11], where basis functions are associated with the individual topological entities of the mesh (see also Whiting *et al.* [7] for an application to the compressible Navier–Stokes equations). Mesh entity based hierarchical basis functions support non-uniform k-refinement of meshes of arbitrary element type, e.g., tetrahedral, hexahedral, and pyramid (k refers to the polynomial order of the finite element basis) while maintaining C_0 continuity.

This paper presents a unique approach to the use of k-order finite element computations. Here, the rich data structure typically used for these computations (see Beall and Shephard [11]) is used only for pre- and post-processing, where the full topological hierarchy (region's relationship to faces, faces' relationship to edges, edges' relationship to vertices) is required. The analysis code is then able to use traditional finite element data structures (where, for each element, local element degrees of freedom are linked to the global degree of freedom) such as those found in Hughes [12]. This enables us to maintain the structure of an already optimized three-dimensional finite element solver, while gaining the efficiency and accuracy of the hierarchical basis. With relatively few modifications, an existing finite element solver can be converted to use the hierarchical basis.

Numerical simulations will be presented that demonstrate a clear advantage of higher-order methods over the traditional, linear basis methods for the incompressible Navier–Stokes equations. It will be shown that the higher-order methods can provide the most cost-effective solutions in terms of both storage and computer time. These simulations have led us to begin the application of these hierarchical basis methods to turbulent flows using the Reynolds-averaged equations as well as LES. While it is expected that non-uniform k-refinement (in addition to h-refinement) may be necessary to get the best results, the simulations presented here were all performed using uniform k meshes.

2. MESH ENTITY BASED HIERARCHICAL BASIS

The hierarchical basis functions employed herein are based on the constructions of Shephard *et al.* [13] for specifying variable k-order meshes based on the individual entities (vertices, edges, faces, and regions) that define the finite element mesh. This is often referred to as a topological hierarchy of mesh entities in the k-version finite element community. The motivation for using this abstract representation is that data structures consisting only of nodal co-ordinates and element connectivity are not sufficient for adaptive, variable k-order finite element meshes, which must rely on richer structures that allow the independent assignment of polynomial order over the elements as noted by Demkowicz *et al.* [14]. These considerations are taken into account in the present work wherever possible. The set of basis functions used here has also been shown to yield better conditioned matrices than other hierarchical bases for tetrahedral elements (see Carnevali *et al.* [15]). Following is a brief discussion of the basis functions; a more complete description may be found in Whiting *et al.* [7]. It should be emphasized that these rich data structures are used only for pre- and post-processing so that the efficiency of the flow solver is maintained for large-scale parallel computations.

2.1. Description of basis

The definition of the piecewise polynomial basis used in the present work requires us to introduce the concept of a topological hierarchy of mesh entities (i.e., the collection of vertices, edges, faces, and regions, which comprise the finite element mesh). Additional details about the mesh data structures may be found in the work of Beall and Shephard [11]. The individual basis functions are defined in terms of parametric co-ordinate systems, ξ_i, which are local to each mesh entity. This is in contrast to the Lagrange basis functions, which are defined solely in terms of element parametric co-ordinates. As mentioned above, efficiency on large-scale problems is maintained by using this rich mesh data structure only in the pre- and post-processing phases of the analysis. During pre-processing, the traditional finite element data structures, generalized to include the higher-order degrees of freedom, are generated and written to disk and subsequently read by the analysis code. This compact data structure stores the degree of freedom connectivity information and nodal co-ordinates as well as information indicating the sign of each basis function (which differs from $+1.0$ only for certain cubic shape functions, see Whiting [16]).

The topological description of the mesh, denoted T_{M}, along with its adjacency relationships, enables a precise definition of a finite element.

Definition 1
The closure of a finite element, denoted $\bar{\Omega}_e$, of dimension d_e, is defined as

$$\bar{\Omega}_e = \{M_e^{d_e}, M_e^{d_e}\{M_j^{d_e-1}\}, \ldots, M_e^{d_e}\{M_j^0\}\} \tag{1}$$

where $M_e^{d_e}$ represents the mesh entity of dimension d_e, which bounds mesh region e.

We have introduced the notation for the mesh entity adjacencies as

$$M_i^{d_i}\{M^{d_j}\} \tag{2}$$

which represents the mesh entities of dimension d_j bounding mesh entity d_i. In other words, the finite element is the mesh region along with its lower-order bounding mesh entities. For example, each tetrahedral element has four bounding vertices, six bounding edges, four bounding faces, and one region.

To construct the finite element approximation to the solution, we expand the continuous quantities appearing in the weak form in terms of a piecewise polynomial basis defined on each element (as described below).

Definition 2
Let $P_k(\bar{\Omega}_e)$ be the piecewise polynomial space, complete to order k, defined on the finite element $\bar{\Omega}_e$.

The basis for $P_k(\bar{\Omega}_e)$ consists of functions $N_a(\xi_i)$, $a = 1, \ldots, n_{\text{es}}$, contributed by the mesh entities in $\bar{\Omega}_e$; the polynomial order assigned to each entity dictates how many basis functions it will contribute. Here, n_{es} is the number of basis functions contributing to a given element's basis and equals the sum of the number of functions contributing from the region and each bounding entity. Also note that ξ_i is a local co-ordinate for the mesh entity in question, and is mapped to the standard co-ordinate system of the element in question (e.g., barycentric co-ordinates for a tetrahedral region). Although the polynomial order may be assigned independently to each mesh entity, it should be noted that the order of complete polynomial representable by a given element's basis will be constrained by the minimum complete order assigned to any of the entities in $\bar{\Omega}_e$. More details of the individual basis functions may be found in Shephard *et al.* [13] or Whiting [16].

To construct element matrices and residual vectors, the discrete solution is expanded in terms of these basis functions as

$$\phi^e(\xi_i, t) = \sum_{a=1}^{n_{\text{es}}} \phi_a(t) N_a(\xi_i) \tag{3}$$

where $\phi^e(\xi_i, t)$ is the finite element approximation of any variable on element e, and $\phi_a(t)$ are the desired coefficients with respect to the basis functions. Note that ξ_i is mapped to the global co-ordinates x_j using a linear mapping involving the vertex shape functions, which, due to the hierarchical nature of the basis, are linear regardless of the polynomial order of the element (note that for non-straight-sided elements such a linear mapping is not sufficient, and a higher-order mapping must be constructed). As mentioned above, the number of basis functions contributed by each mesh entity depends on the polynomial order assigned to that entity. When only C^0 continuity is desired, vertices will each contribute one basis function (equivalent to the standard linear basis functions). For a basis complete to order k, Table I provides the number of basis functions contributed by each type of mesh entity. From this table, we can compute the number of shape functions contributed by a tetrahedral element complete to order k as

Table I. Number of contributed shape functions.

Vertex:	$n_v = 1$	$(k \geq 1)$
Edge:	$n_e = (k-1)$	$(k \geq 2)$
Face:	$n_f = \frac{1}{2}(k-1)(k-2)$	$(k \geq 3)$
Region:	$n_r = \frac{1}{6}(k-1)(k-2)(k-3)$	$(k \geq 4)$

$$n_{es} = 4n_v + 6n_e + 4n_f + n_r = \frac{1}{6}(k+1)(k+2)(k+3) \tag{4}$$

As Table I indicates, edges, faces, and regions only contribute basis functions if the polynomial order is greater than that indicated in the right-hand column. The key difference between hierarchical and Lagrange basis functions is that the hierarchical basis of order $k-1$ is a subset of the basis of order k. This important property provides a natural scale separation that is crucial for the design of new multiscale methods for computing turbulence (see Hughes *et al.* [17]). These LES techniques exploit this capacity by applying common models (e.g., Smagorinsky) only to the higher polynomial order portion of the basis, leaving the low-order scales free of modeling. Hierarchical and Lagrange basis functions also differ in that the Lagrange basis only contributes polynomials of a single order, whereas hierarchical basis functions will be of different order. The polynomial order of each of the basis functions for each entity type is discussed in detail in Shephard *et al.* [13]. To get the total (global) number of basis functions, n_s, for a given mesh we sum over the number of shape functions contributed by each mesh entity for all entities in the mesh. (Note that for a Lagrange basis, n_s is equal to the number of 'nodes' in the mesh.)

As mentioned above, the higher-order computations can be made more efficient (for fixed k) by simply extending the existing finite element data structures to accommodate the additional degrees of freedom emanating from higher-order basis functions. This extension is only possible (to our knowledge) for $k < 4$ due to the nature of the higher-order face functions. For $k \geq 4$, a minimum of an additional face-type data structure must be added.

2.2. Post-processing hierarchical solutions

Post-processing of higher-order solutions presents some difficulty, since current visualization packages typically require linear basis functions represented by element nodal connectivity, with data associated with nodes. Since the solution coefficients of the hierarchical basis are not simply the solution values at specific nodal points (as with the Lagrange basis), additional work is needed to effectively visualize the hierarchical solution. The most straightforward approach is to create a refined mesh, evaluate the hierarchical solution at each of the new nodes, and generate a new element connectivity before using a standard, linear visualization package. This is the method that has been employed herein, where the number of refinement levels was chosen such that the plots did not visibly change as additional levels were added. It is hoped that future visualization packages may incorporate the hierarchical basis function coefficients directly. Research is currently underway in this area. Line plots of solution values are obtained by evaluating the (higher-order) finite element solution at a series of locations in the global co-ordinate system. This operation involves a search through the elements to determine what element a given global point lies in before the solution is evaluated. While this approach can be quite costly for low polynomial-order solutions, where a large number of elements are required, the burden is reduced with high-order solutions owing to the dramatic reduction in the number of elements necessary to attain solutions of similar quality. More details of the post-processing techniques for hierarchical basis simulations may be found in Whiting [16].

2.3. Application of boundary and initial conditions

We would like to conclude this section with a discussion of boundary and initial conditions. Higher-order simulations using Lagrange basis functions enforce essential boundary conditions in a relatively straightforward manner, as basis coefficients correspond to solution values at nodes, *viz*. the Lagrange interpolation equation $N_a(\xi_b) = \delta_{ab}$. Hierarchical basis functions, however, involve the solution of a system of linear equations to compute the basis coefficients of the known boundary condition function. In addition, since the basis coefficients are not associated with particular spatial locations (except vertex functions), a unique set of interpolation points must be chosen.

Suppose we wish to specify that $\phi(x_i) = g(x_i)$ over some portion of the boundary (the entire domain for an initial condition), where $\phi(x_i)$ could be any of the solution variables. The finite element solution procedure uses the values of the basis coefficients that correspond to $g(x_i)$. We can find the coefficients of an approximation to $g(x_i)$ as

$$g(x_i) \approx \hat{g}(x_i) = \sum_{a=1}^{n_{\text{ip}}} g_a N_a \tag{5}$$

where N_a are the element basis functions, g_a are the unknown coefficients, and n_{ip} is the number of interpolation points, which must equal n_{es}, the number of element shape functions. To find these coefficients, we require the approximation to interpolate the given function, i.e.

$$\boldsymbol{M}\boldsymbol{g} = \boldsymbol{R} \tag{6}$$

$$\boldsymbol{M} = [M_{ab}] = N_a(\xi_b^{\text{int}}) \quad \text{and} \quad \boldsymbol{R} = [\boldsymbol{R}_b] = g(x_i(\xi_b^{\text{int}})) \tag{7}$$

where ξ_b^{int} is the bth interpolation point (in element co-ordinates). This system of linear equations is solved for the basis coefficients, g_b, for each element, which are used when needed by the analysis code to evaluate $\phi(x_i)$ (which is expanded in the same basis as $\hat{g}(x_i)$). It should be noted that this boundary condition data is also pre-computed and subsequently read by the analysis code along with the element data structures. The interpolation points used in the present work are taken from the work of Chen and Babuska [18], where they derived an optimal set of interpolation points for a tetrahedral region.

3. INCOMPRESSIBLE NAVIER–STOKES EQUATIONS

Stabilized finite element methods have been proven to be stable and higher-order-accurate for a linear advective–diffusive system (a model problem for the Navier–Stokes equations) in Hughes *et al.* [2] and for the linearized incompressible Navier–Stokes equations in Franca and Frey [6]. These types of formulations have also been effectively used for computing complex compressible and incompressible turbulent flows (see Jansen [9]). The higher-order accuracy properties as well as the robustness on complex flows have motivated our choice of finite element formulation. We first provide the strong form of the incompressible Navier–Stokes equations, followed by a description of the finite element method used to discretize the associated weak form, and finally a discussion of the local reconstruction technique used to obtain the diffusive flux terms (for $k \geq 2$).

3.1. Strong form

Consider the application of the mesh entity based hierarchical basis functions to the time-dependent, incompressible Navier–Stokes equations. First, consider the strong form of the continuity and momentum equations written in the so-called advective form (see Gresho [19])

$$\begin{aligned} &u_{i,i} = 0 \\ &\dot{u}_i + u_i u_{i,j} = -p_{,i} + \tau_{ij,j} + f_i \end{aligned} \tag{8}$$

where u_i is the ith component of velocity, p is the pressure divided by the density ρ (assumed constant), f_i is the prescribed body force (also divided by ρ), and τ_{ij} is the viscous stress tensor given by

$$\tau_{ij} = v(u_{i,j} + u_{j,i}) \tag{9}$$

where $v = \mu/\rho$ is the kinematic viscosity, and the summation convention is used throughout (sum on repeated indices).

3.2. Weak form—finite element discretization

To proceed with the finite element discretization of the weak form of the Navier–Stokes equations (8), we first introduce the discrete weight and solution function spaces that are used. Recall that $\bar{\Omega} \subset \boldsymbol{R}^N$ represents the closure of the physical spatial domain, $\Omega \cup \Gamma$, in N dimensions; only $N = 3$ is considered. The boundary is decomposed into portions with natural boundary conditions, Γ_h, and essential boundary conditions, Γ_g, i.e., $\Gamma = \Gamma_g \cup \Gamma_h$. In addition, $H^1(\Omega)$ represents the usual Sobolev space of functions with square-integrable values and derivatives on Ω (see Hughes [12]).

Subsequently, Ω is discretized into n_{el} finite elements, $\bar{\Omega}_e$. With this, we can define the discrete trial solution and weight spaces for the semi-discrete formulation as

$$\mathscr{S}_h^k = \{\boldsymbol{v} | \boldsymbol{v}(\cdot, t) \in H^1(\Omega)^N, t \in [0, T], \boldsymbol{v}|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^N, \boldsymbol{v}(\cdot, t) = \boldsymbol{g} \text{ on } \Gamma_g\} \tag{10}$$

$$\mathscr{W}_h^k = \{\boldsymbol{w} | \boldsymbol{w}(\cdot, t) \in H^1(\Omega)^N, t \in [0, T], \boldsymbol{w}|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^N, \boldsymbol{w}(\cdot, t) = \boldsymbol{0} \text{ on } \Gamma_g\} \tag{11}$$

$$\mathscr{P}_h^k = \{p \,|\, p(\cdot\,, t) \in H^1(\Omega),\ t \in [0, T],\ p|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)\} \tag{12}$$

where $P_k(\bar{\Omega}_e)$ is defined in Definition 2. Let us emphasize that the local approximation space, $P_k(\bar{\Omega}_e)$, is the same for both the velocity and pressure variables. This is possible due to the stabilized nature of the formulation to be introduced below. These spaces represent discrete sub-spaces of the spaces in which the weak form is defined. Note that the boundary conditions are represented by approximations within the discrete finite element spaces.

The stabilized formulation used in the present work is based on that described by Taylor *et al.* [10] modified to include the higher-order basis functions. Given the spaces defined above, we first present the semi-discrete Galerkin finite element formulation applied to the weak form of Equation (8) as

Find $\boldsymbol{u} \in \mathscr{S}_h^k$ and $p \in \mathscr{P}_h^k$ such that

$$B_{\mathrm{G}}(w_i, q; u_i, p) = 0$$

$$B_{\mathrm{G}}(w_i, q;\ u_i, p) = \int_{\Omega} \{w_i(\dot{u}_i + u_j u_{i,j} - f_i) + w_{i,j}(-p\delta_{ij} + \tau_{ij}) - q_{,i}u_i\}\,\mathrm{d}x$$
$$+ \int_{\Gamma_h} \{w_i(p\delta_{in} - \tau_{in}) + qu_n\}\,\mathrm{d}s \tag{13}$$

for all $\boldsymbol{w} \in \mathscr{W}_h^k$ and $q \in \mathscr{P}$. The boundary integral term arises from the integration by parts and is only carried out over the portion of the domain without essential boundary conditions. Since the Galerkin method is unstable for the equal-order interpolations given above, we add additional stabilization terms, which yields

Find $\boldsymbol{u} \in \mathscr{S}_h^k$ and $p \in \mathscr{P}_h^k$ such that

$$B(w_i, q; u_i, p) = 0$$

$$B(w_i, q;\ u_i, p) = B_{\mathrm{G}}(w_i, q;\ u_i, p) + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\bar{\Omega}_e} \{\tau_{\mathrm{M}}(u_j w_{i,j} - q_{,i})\mathscr{L}_i + \tau_C w_{i,i} u_{j,j}\}\,\mathrm{d}x$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\bar{\Omega}_e} \{w_i \overset{\Delta}{u}_j u_{i,j} + \bar{\tau} \overset{\Delta}{u}_j w_{i,j} \overset{\Delta}{u}_k u_{i,k}\}\mathrm{d}x \tag{14}$$

for all $\boldsymbol{w} \in \mathscr{W}_h^k$ and $q \in \mathscr{P}_h^k$. We have used $\mathscr{L}_i$ to represent the residual of the ith momentum equation

$$\mathscr{L}_i = \dot{u}_i + u_j u_{i,j} + p_{,i} - \tau_{ij,j} - f_i \tag{15}$$

The second term on the right-hand side in the stabilized formulation (14) represents the typical stabilization added to the Galerkin formulation for the incompressible set of equations (see Franca and Frey [6]). The first part of the third term in Equation (14) was introduced by Taylor *et al.* [10] to overcome the lack of momentum conservation introduced as a consequence of the momentum residual appearance in the continuity equation. The second part of this term was introduced to stabilize this new advective term. To see that this formulation conserves momentum, set $\boldsymbol{w} = \{1, 0, 0\}$ and $q = u_1$ in Equation (14), which leaves only boundary terms. This exercise also yields the advective velocity correction to restore conservation, i.e.

$$\overset{\Delta}{u}_i = -\tau_M \mathscr{L}_i \tag{16}$$

The stabilization parameters for continuity and momentum are defined as

$$\tau_M = \frac{C}{\sqrt{c_1/\Delta t^2 + c_2 u_i g_{ij} u_j + c_3 \nu^2 g_{ij} g_{ij}}} \tag{17}$$

$$\tau_C = \frac{1/\tau_M}{\mathrm{tr}(g_{ij})} \tag{18}$$

and the stabilization of the new advective term is defined in direct analogy with τ_M as

$$\bar{\tau} = \frac{C}{\sqrt{c_2 \overset{\Delta}{u}_i g_{ij} \overset{\Delta}{u}_j}} \tag{19}$$

where C, c_1, c_2, and c_3 are defined based on the one-dimensional, linear advection–diffusion equation using a linear finite element basis and $g_{ij} = \xi_{k,i}\xi_{k,j}$ is the covariant metric tensor related to the mapping from global to element co-ordinates. The constant c_3 is modified for higher-order elements to obtain the correct order of convergence in the diffusive limit as required by the use of the inverse estimates in the accuracy analysis of Franca and Frey [6]. More details and precise definitions of these constants may be found in Whiting [16].

3.3. Local reconstruction of diffusive flux

Careful inspection of the weak form (14), and in particular the momentum residual equation (15), reveals that it is necessary to calculate the second derivative of the solution variable when evaluating the residual of the diffusive flux stabilization terms

$$q_i \equiv \tau_{ij,j} = \nu(u_{i,j} + u_{j,i})_{,j} \tag{20}$$

While these terms are often neglected for linear basis calculations (with simple justification), their inclusion is vital to the accuracy of higher-order simulations (examples run without these terms have shown a significant degradation of solution quality). Although it is possible to evaluate these terms directly from the second derivatives of the basis functions, we opt for a more efficient method of using a local reconstruction of the diffusive flux terms based on an L_2-projection followed by a re-interpolation (see Whiting *et al.* [7] or Jansen *et al.* [20]). Briefly

$$\boldsymbol{M}\hat{\tau}_{ij} = \boldsymbol{R}_{ij} \tag{21}$$

where

$$\boldsymbol{M} = [\boldsymbol{M}_{ab}] = \int_{\bar{\Omega}_e} N_a N_b \, \mathrm{d}x, \qquad \boldsymbol{R} = \{\boldsymbol{R}_a\} = \int_{\bar{\Omega}_e} N_a \tau_{ij} \, \mathrm{d}x \tag{22}$$

are solved for the diffusive flux projection coefficients, $\hat{\tau}_{ij}^{a}$, which are then re-interpolated with the gradients of the basis functions to form an approximation to q_i as

$$q_i = \sum_{a=1}^{n_{\mathrm{es}}} N_{a,j} \hat{\tau}_{ij}^{a} \tag{23}$$

Jansen *et al.* [20] also present a technique for global reconstruction of the diffusive flux for linear basis computations which has been shown to improve the accuracy of linear basis computations at a negligible additional cost. Due to these considerations, all linear basis computations shown here use the global reconstruction technique.

3.4. Discrete system of equations

To derive a discrete system of equations, the weight functions w_i and q, the solution variables u_i and p, and their time derivatives are expanded in terms of the finite element basis functions. Since we have a non-linear, time-dependent system of equations, Gauss quadrature of the spatial integrals results in a system of first-order, non-linear ordinary differential equations (ODEs), which can be written as

$$\boldsymbol{R}_A(\boldsymbol{u}_i^{(h,k)}, \dot{\boldsymbol{u}}_i^{(h,k)}, \boldsymbol{p}^{(h,k)}) = 0, \quad A = 1, \dots, n_s \tag{24}$$

where we have assumed the coefficients of the weight functions to be arbitrary and $\boldsymbol{u}_i^{(h,k)}$, $\dot{\boldsymbol{u}}_i^{(h,k)}$ and $\boldsymbol{p}^{(h,k)}$ are the discrete representations of these variables. The superscript (h, k) is omitted for clarity in what follows. The backward Euler method is used to transform this system of ODEs into a non-linear system of algebraic equations, which may be linearized using Newton's method to obtain

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{G} \\ -\boldsymbol{G}^T & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{u} \\ \Delta \boldsymbol{p} \end{pmatrix} = -\begin{pmatrix} \boldsymbol{R}_{\mathrm{m}} \\ \boldsymbol{R}_{\mathrm{c}} \end{pmatrix} \tag{25}$$

where

$$\boldsymbol{K} \approx \frac{\partial \boldsymbol{R}_{\mathrm{m}}}{\partial \boldsymbol{u}}, \qquad \boldsymbol{G} \approx \frac{\partial \boldsymbol{R}_{\mathrm{m}}}{\partial \boldsymbol{p}}, \quad \text{and} \quad \boldsymbol{C} \approx \frac{\partial \boldsymbol{R}_{\mathrm{c}}}{\partial \boldsymbol{p}} \tag{26}$$

and $\boldsymbol{R}_{\mathrm{m}}$ and $\boldsymbol{R}_{\mathrm{c}}$ represent the portions of the residual from the momentum and continuity equations respectively, and $\boldsymbol{K}$, $\boldsymbol{G}$, and $\boldsymbol{C}$ are approximations to the full tangent matrices. This linear system of equations is solved (at each time step) and the solution is updated *viz.*

$\boldsymbol{u}^{i+1} = \boldsymbol{u}^i + \Delta \boldsymbol{u}$ and $\boldsymbol{p}^{i+1} = \boldsymbol{p}^i + \Delta \boldsymbol{p}$ for each of the Newton iterations. For steady problems, only one Newton iteration per time step is performed. For unsteady problems, the generalized α-method time integrator presented in Jansen *et al.* [21] is used. The linear algebra solver of Shakib [22] is used to solve the system of Equations (25).

3.5. Parallel processing

The hierarchical basis simulations rely on message passing implemented using the Message Passing Interface (MPI) library [23] for all parallel communications. To make efficient use of the MPI library requires that the communication structures be pre-processed, at which time all necessary data structures for exchanging information between processors during the computation are set up. The mesh is partitioned such that each element is associated with a unique processor (partitioning software such as METIS [24] can be used to perform this task). Based on this information, a methodology has been developed by which each mesh entity on the inter-processor boundary is assigned a unique master processor, on which equations relating to its shape functions are solved. Information pertaining to the mesh entities lying on the inter-processor boundary is then collected and used to carry out the communications (see Whiting [16]).

4. COMPUTATIONAL EFFORT

The discretization method has been introduced above with little consideration to the computational effort engendered by each of the different polynomial orders. Solution time, memory use, and disk storage are clearly the most important measures when considering the cost of a simulation. However, by looking at time alone, we would fail to assess the scalability of the method to large-scale problems. In this section, we introduce three additional measures that will be used to quantify the benefit of using higher polynomial order. These measures take into account the costs of computing the distinct components of a simulation, i.e., tangent matrix, residual vector, and linear system solution.

The examples we present are two-dimensional, since we wish to discuss the cost for problems with well-understood benchmark results. It is somewhat difficult to get a fair cost comparison on two-dimensional problems when using the three-dimensional code, since the cost of the higher polynomial order simulations is penalized for adding many additional degrees of freedom in the third (inactive) dimension. Still, we would like to make some estimates of the relative simulation cost, so we consider three cost indices, C_1, C_2, and C_3, defined as

$$C_1 = n_{\mathrm{f}} \times n_{\mathrm{shp}}^2 \times n_{\mathrm{int}} \tag{27}$$

$$C_2 = n_{\mathrm{f}} \times n_{\mathrm{shp}} \times n_{\mathrm{int}} \tag{28}$$

$$C_3 = n_{\mathrm{k}} \times (n_{\mathrm{v}} \times n\mathrm{nz}_{\mathrm{v}} + n_{\mathrm{e}} \times n\mathrm{nz}_{\mathrm{e}} + n_{\mathrm{f}} \times n\mathrm{nz}_{\mathrm{f}}) \tag{29}$$

where n_{f} is the number of equivalent two-dimensional triangular face elements, n_{shp} is the number of two-dimensional degrees of freedom, and n_{int} is the number of triangular integration points required to accurately integrate a two-dimensional element. For the linear solver cost, we have used n_k to represent the number of Krylov vectors needed, and $n\mathrm{nz}_v$, $n\mathrm{nz}_e$, and $n\mathrm{nz}_f$ for the non-zero fill pattern associated with vertices, edges, and faces (predictable only for uniform meshes). The first of these measures, C_1, represents the computational cost associated with the formation of the left-hand side or residual tangent matrix. C_2 is associated with the cost of forming the residual vector (right-hand side). C_3 relates to the cost of solving the linear system, which is dominated by the non-zero fill pattern. This cost is relevant to our linear solve since we are using a sparse iterative solver, which, for each Krylov vector, performs a matrix–vector product only with the non-zero matrix entries (see Saad [25]). The impact of each of these cost measures is somewhat problem-dependent, and more details will be discussed with respect to the individual simulations presented in the following section.

5. NUMERICAL EXAMPLES

This section presents numerical simulations using the hierarchical basis methods described above. The accuracy of the method is first demonstrated on a problem with a closed-form analytical solution. It is shown that the method converges at the theoretical rates in both the L_2- and H_1-norms. Additional examples are then provided that serve to demonstrate the ability of higher-order basis methods to attain more accurate simulations for substantially less cost. The results for the higher-order simulations shown in this section all employ the methodology described in Section 2.2 for creating higher-order visualizations.

The simulations described below were all performed with the full three-dimensional code. Thus, to simulate the two-dimensional flows described below, we have used two vertices in the x_3-direction and imposed no x_3 velocity and zero viscous flux through the x_3 planes; i.e., $u_3 = 0$ and $\tau_{i3} = 0$. The depth in the x_3-direction was set equal to the length of an element in the x_1-direction.

5.1. Kovasznay flow

The first example may be identified with the laminar flow behind a grid, and is known as the Kovasznay flow (see Kovasznay [26]). We will use this flow to demonstrate the convergence of the method, since we have a closed-form analytical expression for the exact solution, given by

$$u_1 = 1 - \mathrm{e}^{\lambda x_1}\cos(2\pi x_2) \tag{30}$$

$$u_2 = \frac{\lambda}{2\pi}\mathrm{e}^{\lambda x_1}\sin(2\pi x_2) \tag{31}$$

with

$$\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2} \tag{32}$$

and we have taken $Re = 40$ for the present study. The flow is considered on a rectangular domain of $-1/2 \leq x_1 \leq 1$ and $-1/2 \leq x_2 \leq 3/2$ with the exact solution imposed as an essential boundary condition at the inflow and upper and lower walls, while the pressure was set at the outflow. The qualitative behavior of the solution is depicted in Figure 1, which shows contours of fluid speed from the cubic simulation on the 21×21 mesh.

A convergence study was performed for this flow to determine the accuracy of the method with respect to the L_2- and H_1-norms shown in Figure 2(a) and (b) respectively. Table II summarizes the convergence results and demonstrates that the method is performing at the theoretical convergence rate in all cases, $O(h^{k+1})$ and $O(h^k)$, for the L_2- and H_1-error norms respectively.

It is also clear from Figure 2(a) and (b) that the constant in the error estimate also greatly improves for the higher-order simulations, making the higher polynomial order basis most attractive even on the coarsest meshes. This is a particularly attractive feature of stabilized methods. Figure 3 demonstrates the exponential convergence of the method when Δx_1 is fixed and the polynomial order is increased.

5.2. *Flow over a backward-facing step*

Consider a two-dimensional flow over a backward-facing step at $Re = 800$, based on the step height and the average inflow velocity. The geometry and boundary conditions are similar to those used by Gartling [27]. The problem is specified by a fully developed flow entering a

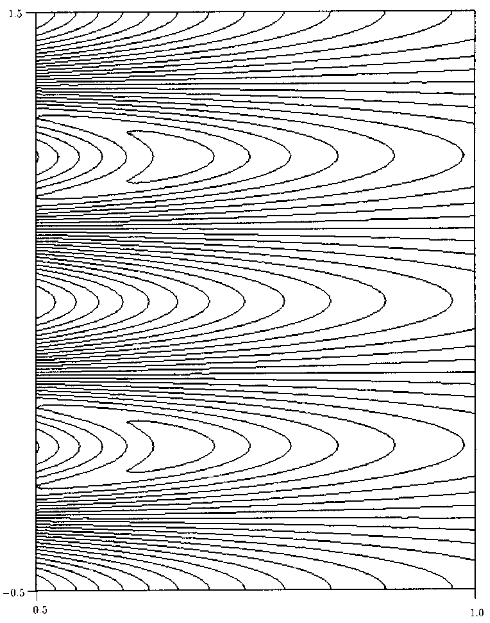
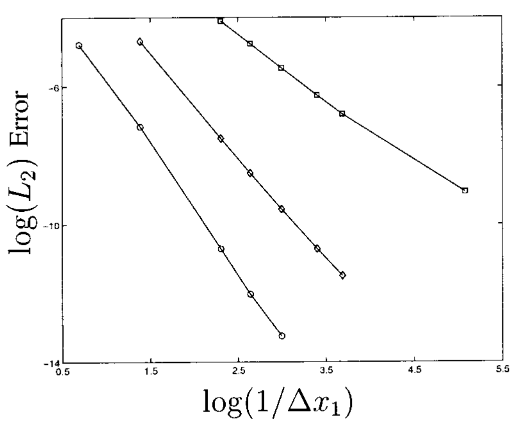
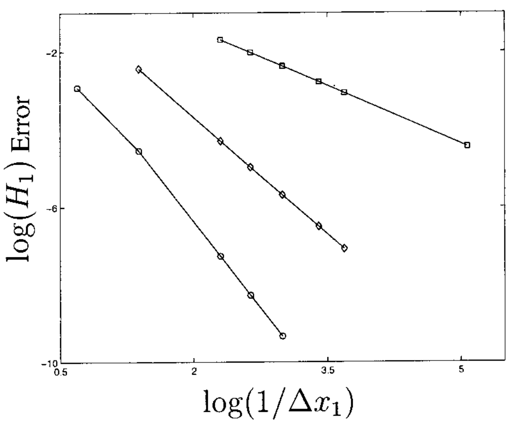


Figure 1. Kovasznay flow. Contours of fluid speed for cubic simulation on 21×21 mesh.



(a) log of L_2 error vs. $\log(1/\Delta x_1)$



(b) log of H_1 error vs. $\log(1/\Delta x_1)$.

Figure 2. Kovasznay flow convergence study: □, $k=1$; ◇, $k=2$; ○, $k=3$.

Table II. Convergence rates for Kovasznay flow.

k	L_2-norm	H_1-norm
1	1.97	0.99
2	3.00	2.01
3	3.93	2.98

confined channel which, at $Re = 800$, has been demonstrated by numerous researchers to be steady and stable (see Gresho *et al.* [19]). The geometry and boundary conditions are shown in Figure 4, and the initial condition consists of a parabolic velocity profile imposed upon the entire channel. This initial condition is marched in time using the backward Euler technique until the steady solution is reached, confirmed in all cases by monitoring the changes in various flow quantities.

Since the objective of this study was the comparison of various polynomial-order bases rather than a complete description of the physics, the standard step flow geometry was simplified by excluding the region upstream of the step as described by Gartling [27]. This also allows for a more accurate comparison with his benchmark results. Numerical solutions were obtained on a variety of uniform tetrahedral meshes for several different polynomial orders. The mesh statistics (for the equivalent two-dimensional problem) are shown in Table III. Here, Δx_1 and Δx_2 represent the element size in the x_1 and x_2 direction, respectively.

The basic character of this flow is well known. At $Re = 800$, there are two separation regions, one starting at the step corner and continuing downstream approximately 12 step heights, and another on the upper wall of the channel occupying a region from approximately 10–20 step heights downstream. These key features are shown in Figure 5(a)–(c), which

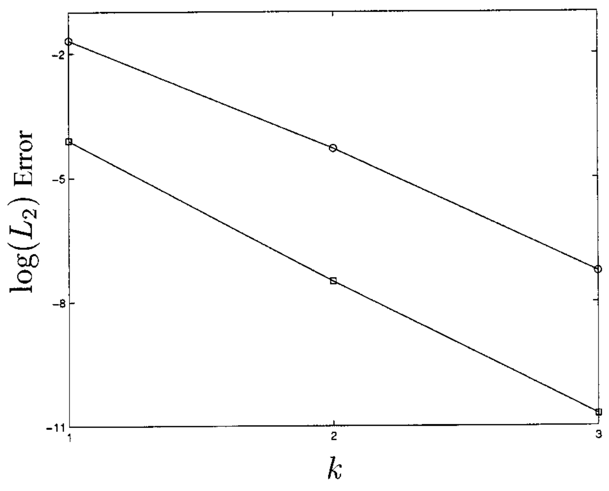


Figure 3. Log of error versus polynomial order for Kovasznay flow: ○, H_1-norm; □, L_2-norm.

represent the fluid speed, pressure, vorticity, and velocity vectors for the cubic simulation on mesh C. These figures are shown in the correct scale, however, only the first ten step heights of the channel are shown. Qualitatively, these figures compare well with those presented in Gartling [27].

Two-dimensional contour plots of various flow quantities look similar for all simulations making it difficult to quantify the benefit of the higher-order methods. We will, therefore, compare line plots of these quantities at different spatial locations. Figure 6 presents a comparison between the cubic simulation on mesh A, the quadratic on mesh C, and the linear on mesh E. The x_1- and x_2-velocities and pressure are shown at two locations along the channel, $x_1 = 7.0$ and $x_1 = 15.0$, the same locations presented in Gartling [27], which we have included on the velocity plots as a benchmark result. The cubic and quadratic are able to exactly reproduce the benchmark simulation, while the linear, even on the most refined grid, is still slightly off in the x_2-velocity and pressure at the $x_1 = 7.0$ location. This is not surprising, since the benchmark result is from a quadratic simulation with 41 vertices across the channel. More refined simulations were run for the quadratic (mesh D) and cubic (mesh C) bases to confirm that these solutions were grid independent.

These three simulations represent qualitatively similar results. Clearly, the only results that visibly differ from the benchmark result are the linear x_2 velocity and pressure at $x_1 = 7.0$, which is the most sensitive quantity in the study. Based on the data in Table III and the linear solver information, the cost measures introduced in Section 4, Equations (27)–(29), were computed for these three simulations. The results are summarized in Table IV. For the purposes of computing C_3 and CPU time, we have converged each of the simulations so that the maximum delta increment in any component of the solution was less than 1×10^{-6}. Note that all the cost estimates have been normalized by the cubic costs, to clarify the presentation.

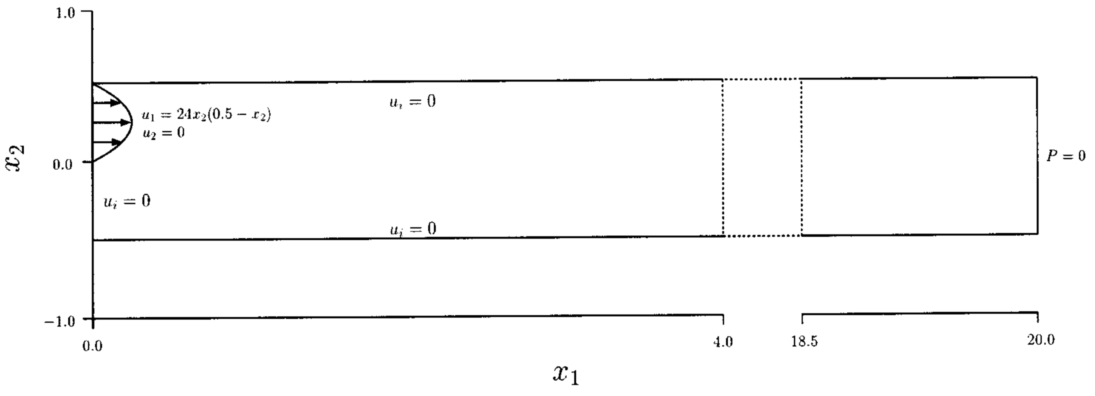


Figure 4. Step flow geometry and problem description.

The CPU time included in the table includes all three-dimensional modes, and therefore is not an entirely accurate measure of the full benefit of higher-order simulations, since many higher-order modes are wasted on the inactive third dimension. Furthermore, the linear mesh probably requires another level of refinement to attain the same quality as the quadratic and cubic solutions and we are perhaps erring on the conservative side with these estimates.

Table IV clearly shows that the cubic simulation is the most cost effective, by any of the measures. It can be seen that C_1, related to the cost of forming the tangent matrix, is the least sensitive to polynomial order (although still over a factor of 2 better than the linear). This is to be expected since this number is proportional to n_{shp}^2, so that despite the reduction in the number of elements, this cost remains relatively constant. However, for steady problems, it has been observed that it may not be necessary to form the tangent matrix every time step, and may be used for ten or more iterations before reforming. The cost of the linear solve, represented by C_3 indicates that the cubic simulation is over 40 times cheaper than the linear. While one expects the equations to get stiffer as k is increased, to get the same level of accuracy, Δx must be decreased so far (for $k = 1$) as to make the linear system much more ill-conditioned than the modest increase in condition number associated with an increase in k.

Table III. Mesh statistics

Mesh	Vertices	Edges	Faces	Δx_1	Δx_2
A	405	1044	640	0.250	0.250
B	847	2286	1440	0.167	0.167
C	2211	6210	4000	0.100	0.100
D	8421	24 420	16 000	0.050	0.050
E	32 841	96 840	64 000	0.025	0.025

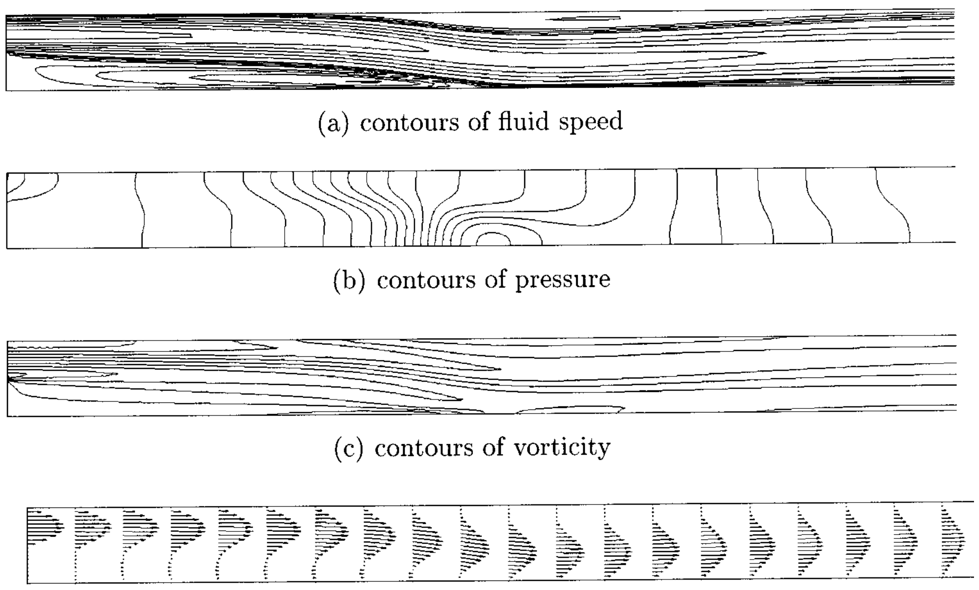


Figure 5. Step flow simulation characteristics: mesh C, $k = 3$.

A further study was carried out to determine the accuracy of the pressure for the linear basis method. Since traditional Galerkin methods must interpolate pressure one order lower than the velocity, the pressure is necessarily one order less accurate. The stabilized method does not suffer from this limitation. This is demonstrated by comparing two linear basis simulations with the most refined cubic simulation at the $x_1 = 7$ location (see Figure 7). The log of the L_∞ error versus Δx_2 shows a slope of 2.1, which is slightly better than optimal for the meshes considered; the optimal L_∞ error for the interpolation being $O(h^2)$ (see Johnson [28]).

5.3. Lid-driven cavity flow

The final problem considered is the steady, two-dimensional flow inside a closed container driven by its lid. The lid slides to the right across the top of the cavity, shearing the fluid and setting up a recirculation region. There is a primary vortex in the center of the cavity and secondary eddies in the lower corner (the number of these secondary eddies depends on the Reynolds number). For the present study, we have chosen to consider $Re = 400$ (based on the lid velocity), for which there exists well-established benchmark results with which to compare (see Ghia *et al.* [29]). Since the velocity is discontinuous at both upper corners, singularities will develop in the pressure field, which must be controlled by the method.

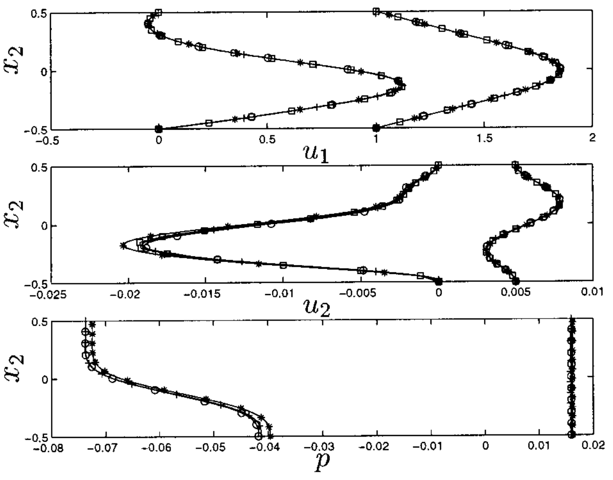


Figure 6. Bachward-facing step: comparison of qualitatively similar solutions. Velocity and pressure are plotted versus x_2 at $x_1 = 7$ and $x_1 = 15$. Velocities at $x_1 = 15$ were shifted for plotting: $*$, $k = 1$; $+$, $k = 2$; ○, $k = 3$; □, Gartling [27].

The geometry and boundary conditions are illustrated in Figure 8. In addition to the velocity constraints, the pressure field is constrained by setting its value at the single vertex in the lower left corner of the cavity. Uniform meshes were used with equal spacing in the x_1- and x_2-directions. To isolate the singularities in the upper corners, nested local mesh refinement was achieved by sub-dividing the original corner elements. The number of new corner elements was chosen such that the first point is 3.90625×10^{-4} units from the corner for each mesh. This distance dictates the extent to which the discontinuity in the velocity field is resolved (i.e., how much fluid is 'leaked' from the cavity). The statistics for these meshes are shown in Table V. These figures do not include the refinement in the upper corners. Using these meshes, linear simulations were run on meshes C–E, quadratic on B–D, and cubic on A–C. These simulations were advanced in time until the normalized changes in the solution variables (u and p) were less than 1×10^{-6}.

The basic solution characteristics of this flow are shown in Figure 9(a)–(c), which displays contours of fluid speed, pressure, and vorticity, as well as velocity vectors, respectively. The

Table IV. Step flow simulation cost comparison.

Mesh	K	C_1	C_2	C_3	CPU time
A	3	1.00	1.00	1.00	1.00
C	2	1.13	1.88	5.26	1.80
E	1	2.25	7.50	43.93	6.55

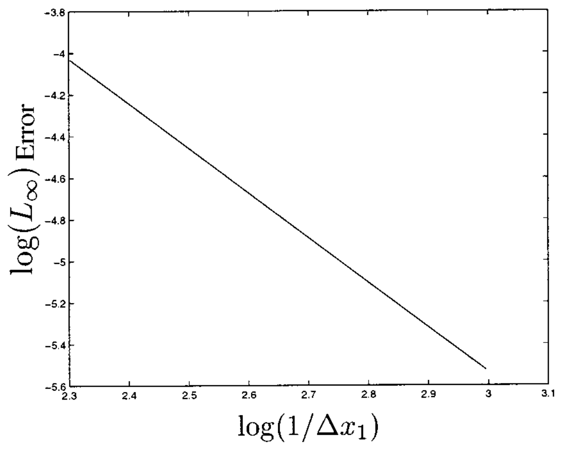


Figure 7. Log of L_∞ error in pressure versus $\log(1/\Delta x_1)$.

plots shown here are the quadratic simulation on mesh C; however, all converged simulations look identical. To more carefully monitor the convergence, we have again made use of two-dimensional line plots of velocity. Figure 10(a) shows profiles of $u_2(x_1, x_2 = 0)$ and $u_1(x_1 = 0, x_2)$ for the most refined mesh for each polynomial order. Note that the u_1 velocity

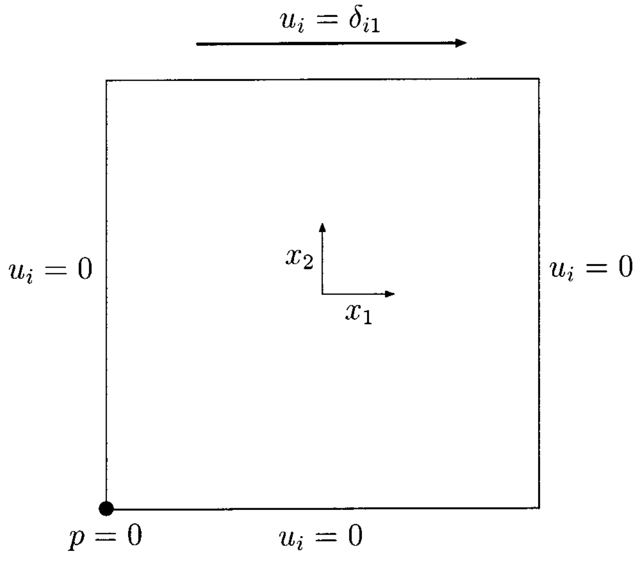


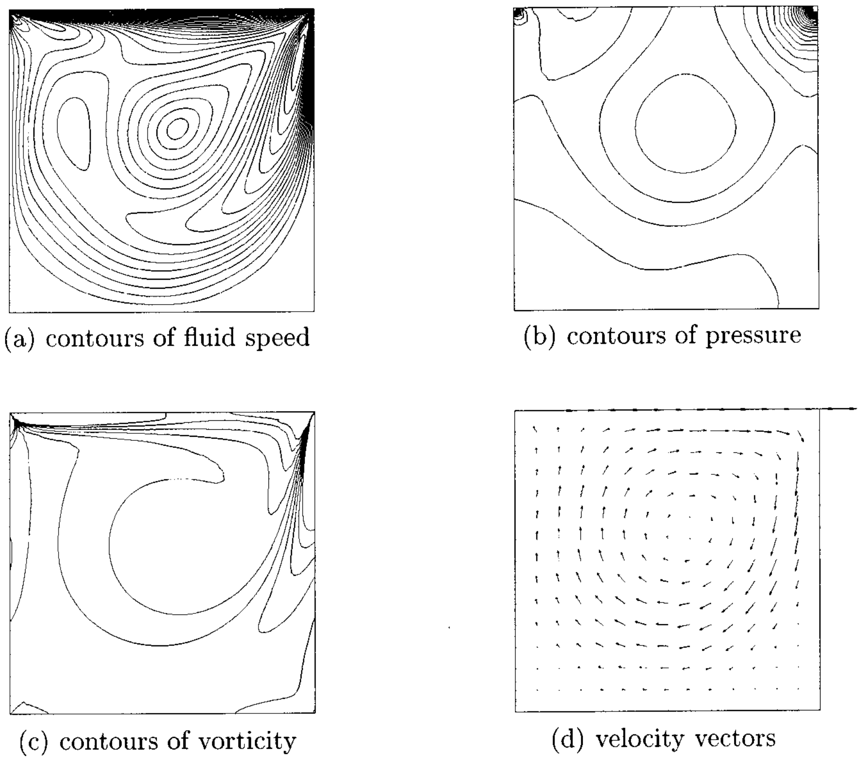
Figure 8. Lid-driven cavity geometry and boundary conditions.

Table V. Lid-driven cavity mesh statistics.

Mesh	Vertices	Δx_1
A	11×11	0.1
B	21×21	0.05
C	41×41	0.025
D	81×81	0.0125
E	161×161	0.00625

was scaled by 0.5 to facilitate plotting. Also shown is the benchmark result of Ghia *et al.* [29]. The three plots are virtually indistinguishable.

A cost comparison study similar to that for the backward-facing step flow was carried out for the lid-driven cavity flow (see Figure 10(b)). The three cost indices and CPU time are summarized in Table VI. The values of the cost indices in this case are even more dramatic than in the case of the backward-facing step flow. For this simulation, we have also provided

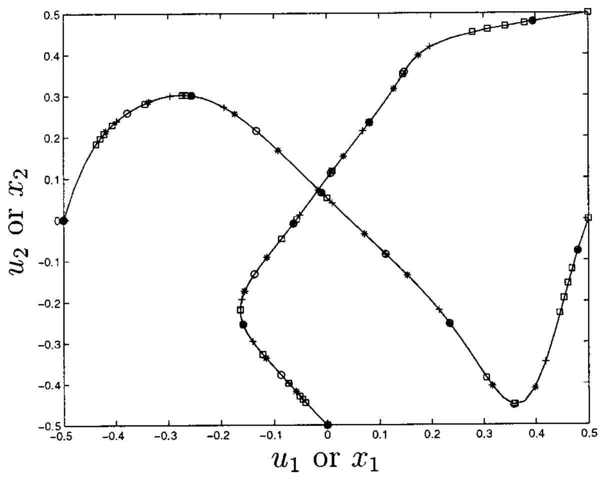

(a) contours of fluid speed

(b) contours of pressure

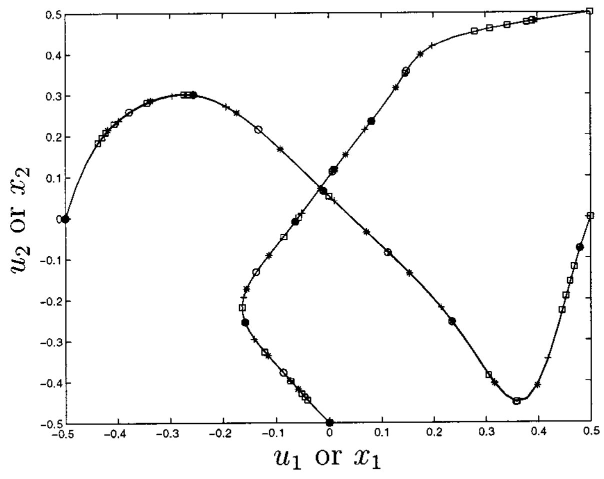
(c) contours of vorticity

(d) velocity vectors

Figure 9. Lid-driven cavity flow characteristics: mesh D, $k = 2$.



(a) Comparison on finest meshes.



(b) Qualitatively similar solutions: linear on mesh E, quadratic on mesh C and cubic on mesh A.

Figure 10. Lid-driven cavity flow. Plots of $u_1(x_1 = 0, x_2)$ and $u_2(x_1, x_2 = 0)$: ○, $k = 1$; $*$, $k = 2$; $+$, $k = 3$; □, Ghia *et al.* [29].

information comparing the memory requirements and disk storage required for the simulations. The 'Matrix storage' column of Table VI indicates the number of non-zero blocks for the sparse storage of the tangent matrix (the dominant memory requirement), indicating that the memory requirements for the cubic simulation are about 15 times less than the linear, while the cubic is about six times better than the quadratic. The 'Mesh size' column compares the size in megabytes of the mesh data file. This size indicates the storage of the compact data structure, used in the analysis code, not the entire rich mesh database file. It should be pointed out that the size of the rich mesh database file is fixed for each mesh, regardless of the polynomial order.

6. CONCLUSIONS

A stabilized finite element method using hierarchical basis functions applied to the incompressible Navier–Stokes equations has been presented. The implementation is general, allowing three-dimensional simulations on arbitrary unstructured meshes. The goal of the present work was to clarify the potential benefit of using higher-order basis functions with stabilized methods. To achieve this goal, which we believe to have been accomplished, only relatively simple geometries and laminar flows have been considered in the present work. Due to the positive outcome of the present study, applications of the hierarchical basis are currently underway for turbulent flows using both the Reynolds averaged equations and LES. It is expected that the higher-order methods will reduce the computational cost of these simulations to a level that will enable better simulations than are currently available using only linear basis methods. In addition, variational multi-scale simulations of turbulence based on the hierarchical basis presented in this work are underway, with preliminary models being presented in Hughes *et al.* [17].

We have presented simulations that compare the cost versus accuracy of higher-order simulations. These cost comparisons have demonstrated that the higher-order simulations can be used to obtain much more cost-effective results when compared with traditional linear basis methods. The higher-order simulations also decrease the amount of core memory required for a simulation. It is expected that the benefit of the hierarchical basis will be even more profound when truly three-dimensional flows are considered. On the two-dimensional simulations, much of the computational effort is wasted on the third quasi-dimension.

Table VI. Lid flow simulation cost comparison.

Mesh	K	C_1	C_2	C_3	CPU time	Matrix storage	Mesh size
A	3	1.00	1.00	1.00	1.00	1.00	1.00
C	2	2.88	4.80	21.60	4.99	6.55	6.02
E	1	32.0	19.2	440.7	6.42	15.65	41.58

ACKNOWLEDGMENTS

The authors would like to thank Anil Kumar Karanam for assistance running some of the simulations and processing the results. They would also like to thank Farzin Shakib of Acusim Software for making his linear algebra software available to us. CHW was supported by a NASA LaRC grant NGT-1-52160 and KEJ was supported by an AFOSR grant F49620-97-1-0043.

REFERENCES

1. Brooks AN, Hughes TJR. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**: 199–259.
2. Hughes TJR, Franca LP, Hulbert GM. A new finite element formulation for fluid dynamics: VIII. The Galerkin/least-squares method for advective–diffusive equations. *Computer Methods in Applied Mechanics and Engineering* 1989; **73**: 173–189.
3. Hughes TJR. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering* 1995; **127**: 387–401.
4. Rússo A. Bubble stabilization of the finite element methods for the linearized incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1996; **132**: 335–343.
5. Brezzi F, Franca LP, Hughes TJR, Rússo A. $b = \int g$. *Computer Methods in Applied Mechanics and Engineering* 1997; **145**: 329–339.
6. Franca LP, Frey S. Stabilized finite element methods: II. The incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**: 209–233.
7. Whiting CH, Jansen KE, Dey S. Hierarchical basis in stabilized finite element methods for compressible flows. SCOREC Report 11-2000, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY *Computer Methods in Applied Mechanics and Engineering* 2000 to appear.
8. Jansen KE, Johan Z, Hughes TJR. Implementation of a one-equation turbulence model within a stabilized finite element formulation of a symmetric advective–diffusive system. *Computer Methods in Applied Mechanics and Engineering* 1993; **105**: 405.
9. Jansen KE. A stabilized finite element method for computing turbulence. *Computer Methods in Applied Mechanics and Engineering* 1999; **174**: 299–317.
10. Taylor CA, Hughes TJR, Zarins CK. Finite element modeling of blood flow in arteries. *Computer Methods in Applied Mechanics and Engineering* 1998; **158**: 155–196.
11. Beall MW, Shephard MS. A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering* 1997; **40**(9): 1573–1596.
12. Hughes TJR. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall: Englewood Cliffs, NJ, 1987.
13. Shephard MS, Dey S, Flaherty JE. A straight forward structure to construct shape functions for variable p-order meshes. *Computer Methods in Applied Mechanics and Engineering* 1997; **147**: 209–233.
14. Demkowicz L, Oden JT, Rachowicz W, Hardy O. Toward a universal h-p adaptive finite element strategy. Part 1: constrained approximation and data structure. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**: 79–112.
15. Carnevali P, Morris RB, Tsuji Y, Taylor G. New basis functions and computational procedures for p-version finite element analysis. *International Journal for Numerical Methods in Engineering* 1993; **36**: 3759–3779.
16. Whiting CH. Stabilized finite element methods for fluid dynamics using a hierarchical basis. PhD thesis, Rensselaer Polytechnic Institute, 1999.
17. Hughes TJR, Mazzei L, Jansen KE. Large-eddy simulation and the variational multiscale method. *Computing and Visualization in Science* 2000; **3**: 47–59.
18. Chen Q, Babuška I. The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. *Computer Methods in Applied Mechanics and Engineering* 1996; **137**: 89–94.
19. Gresho PM. Some current CFD issues relevant to the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1991; **87**: 201–252.
20. Jansen KE, Collis SS, Whiting CH, Shakib F. A better consistency for low-order stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering* 1999; **174**: 153–170.
21. Jansen KE, Whiting CH, Hulbert GM. A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. Contributed to a special volume of *Computer Methods in*

Applied Mechanics and Engineering devoted to the Japan–US Symposium on FEM in Large-scale CFD, SCOREC Report 10-1999, 1999.
22. Shakib F. http://www.acusim.com, 2000.
23. Gropp W, Lusk E, Skjellum A. *Using MPI*. MIT Press: Cambridge, 1994.
24. Schloegel K, Karypis G, Kumar V. Multilevel diffusion algorithms for repartitioning of adaptive meshes. Technical Report # 97-013, University of Minnesota, Department of Computer Science and Army HPC Center, 1997.
25. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing: Boston, 1996.
26. Kovasznay LIG. Laminar flow behind a two-dimensional grid. *Proceedings of the Cambridge Philosophical Society* 1948; **44**: 58–62.
27. Gartling DK. A test problem for outflow boundary conditions—flow over a backward-facing step. *International Journal of Numerical Methods in Fluids* 1990; **11**: 953–967.
28. Johnson C. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press: Cambridge, 1987.
29. Ghia U, Ghia KN, Shin CT. High-*Re* solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics* 1982; **48**: 387–441.