STABILIZED FINITE ELEMENT METHODS FOR FLUID DYNAMICS USING A HIERARCHICAL BASIS

By

Christian H. Whiting

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mechanical Engineering

Approved by the
Examining Committee:

Dr. Kenneth E. Jansen, Thesis Adviser

Dr. Joseph E. Flaherty, Member

Dr. Zvi Rusak, Member

Dr. Mark S. Shephard, Member

Rensselaer Polytechnic Institute
Troy, New York

September 1999
(For graduation December 1999)

STABILIZED FINITE ELEMENT METHODS FOR FLUID DYNAMICS USING A HIERARCHICAL BASIS

By

Christian H. Whiting

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mechanical Engineering

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Dr. Kenneth E. Jansen, Thesis Adviser
Dr. Joseph E. Flaherty, Member
Dr. Zvi Rusak, Member
Dr. Mark S. Shephard, Member

Rensselaer Polytechnic Institute
Troy, New York

September 1999
(For graduation December 1999)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

Many people have helped make this thesis possible. First, I would like to thank the members of my committee, Joe Flaherty, Mark Shephard, and Zvi Rusak, for taking the time to review this thesis and provide many useful suggestions which have greatly improved its quality. I have also benefited from the many technical discussions with my committee members regarding this work, which have served to guide and strengthen it along the way. In particular, I would like to thank my advisor, Ken Jansen, for all that he has taught me during the past several years. Ken has been instrumental in the success of this research, and has been an excellent technical leader and role model. I would also like to acknowledge Saikat Dey, who spent many hours teaching me about the hierarchical basis, and who's research helped inspire this thesis.

I would like to acknowledge the Scientific Computation Research Center, which has been an incredibly fruitful environment to work in. Not only the computer resources, but also my fellow graduate students and research staff have greatly aided this work. These interactions have enhanced this research, by helping me solve many technical difficulties that have arisen during the course of the work. I would also like to acknowledge the support of NASA Langley Research Center, and my NASA technical advisor, Mark Carpenter, for providing technical guidance.

Finally, I would like to thank my parents for all that they have provided me throughout my educational experience, without which none of this work would have been possible. I would like to also thank Amy, whose encouragement and emotional support have been invaluable to me.

ABSTRACT

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. This work presents an application of mesh entity based, hierarchical basis functions to a new stabilized finite element formulation, which is shown to yield high accuracy and more cost effective simulations when compared with the traditional, linear basis methods. The new formulation is then demonstrated numerically to yield nearly optimal rates of convergence with respect to the interpolation error. A second-order accurate, implicit time integrator with user-controllable numerical dissipation is presented for advancing the semidiscrete system of equations in time. This time integrator is proven to be stable and second-order accurate for a linear model problem, and demonstrated to have desirable characteristics on more complicated flows. A variety of examples are provided that demonstrate 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 standard linear basis. The formulation has also been successfully applied to unsteady flows, and several examples will be given. An application to a direct numerical simulation (DNS) of a turbulent channel flow at $Re_\tau = 180$ is then presented to assess the usability of the hierarchical basis for a more complex turbulent flow. Postprocessing techniques are also described for the effective visualization of hierarchical solutions, as well as numerical evaluation of turbulent statistics.

CHAPTER 1
INTRODUCTION AND HISTORICAL REVIEW

Computational fluid dynamics (CFD) has been rapidly gaining popularity over the past several years for technological as well as scientific interests. For many problems of industrial interest, experimental techniques are extremely expensive or even impossible due to the complex nature of the flow configuration. Analytical methods are often useful in studying the basic physics involved in a certain flow problem, however, in many interesting problems, these methods have limited direct applicability. The dramatic increase in computational power over the past several years has led to a heightened interest in numerical simulations as a cost effective method of providing additional flow information, not readily available from experiments, for industrial applications, as well as a complementary tool in the investigation of the fundamental physics of turbulent flows, where analytical solutions have so far been unattainable. It is not expected (or advocated), however, that numerical simulations replace theory or experiment, but that they be used in conjunction with these other methods to provide a more complete understanding of the physical problem at hand. Turbulence researchers are now able to use direct numerical simulation (DNS) to study the basic physics of turbulent flows. Kim *et al.* [55] present an application of DNS to channel flow, and Le *et al.* [57] present a DNS application to flow over a backward-facing step. Both of these studies were conducted to gain new insight into the physical mechanisms involved in turbulent flow.

As computational power grows, the need for more advanced numerical algorithms also increases. There are many different techniques for constructing numerical solutions of fluid flow problems, e.g. finite difference methods, finite volume methods, and finite element methods, to name a few, and all have their strengths and weaknesses. Since the goal of the present research lies in the development of methods which may ultimately be used for large-scale applications of industrial interest, finite element methods have been chosen, given their accuracy as well as their ability to approximate arbitrarily complex geometric configurations. The finite element method applied to fluid dynamics has reached a level of maturity over the past two decades such that it is now being successfully applied to industrial strength problems including turbulent flows (for example, see Haworth and Jansen [35] for an application to reciprocating IC engines). Due to its robustness and proven accuracy, this numerical technique has been chosen for the foundation of the present research.

One of the goals of the present work is to use hierarchical basis functions as a means to attain more accurate and cost-effective finite element simulations of complex turbulent flows. It is hoped that this will enable simulations of fluid dynamical problems that are not presently feasible due to current cost restrictions. With these goals in mind, we have chosen a stabilized finite element formulation based on the formulation of Taylor *et al.* [71] for incompressible flows, that has been generalized to accommodate higher-order basis functions. This formulation has been demonstrated to be robust and accurate for laminar as well as turbulent flow simulations using linear basis functions. The new stabilized formulation builds global conservation into the weak formulation that is lacking in many previous formulations due to the presence of the stabilization of the continuity equation. This, combined with the higher-order accuracy that stabilized methods have been shown to attain, has influenced our selection of this formulation for constructing higher-order simulations.

Over the last two decades, stabilized finite element methods have grown in popularity, especially for fluid dynamics applications. Starting with the SUPG method of Brooks and Hughes [11] through the work of Hughes *et al.* [38] on the Galerkin/least squares (GLS) method, and the streamline diffusion method (related to the SUPG method) of Hansbo and Szepessy [30], a number of stabilized formulations have been proposed. Recent work on variational multiscale methods of Hughes [36] and related work on residual-free bubbles by Ruśso [59] and Brezzi *et al.* [10] have not only proposed new directions for these methods, but have also begun to uncover the theoretical basis for their design. Recent application of the variational multiscale method to large eddy simulation of turbulence by Hughes *et al.* [41] has also proven extremely fruitful. 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 Franca *et al.* [22], and Hughes *et al.* [38]). Johnson and Szepessy [51] have also carried out a non-linear analysis of the related streamline diffusion method for the Burgers equation. This implies that as the polynomial order of the underlying finite element space is increased, the error in the numerical solution is of the same order as the interpolation error. This property is crucial to the effective use of higher-order basis functions.

Over the past several years, many research groups have applied higher-order discretization methods to fluid dynamics simulations in an effort to achieve highly accurate simulations on unstructured grids. Sherwin and Karniadakis [67] developed a C^0 continuous hierarchical basis based on a generalized tensor product using mixed-weight Jacobi polynomials and applied it to a higher-order splitting scheme for the incompressible Navier-Stokes equations in [69]. They presented numerical results to verify the convergence properties of their method. For Euler flows, the discontinuous Galerkin method provides a straightforward way of constructing higher-order solutions (see Biswas *et al.* [8] and also Devine [20]). Oden *et al.* [58] have recently successfully applied the discontinuous Galerkin method to diffusion type problems using arbitrary polynomial order in each element. Others have generalized spectral methods to unstructured grids to achieve spectral accuracy without being restricted to regular domains (see Carpenter and Gottlieb [13] and Sherwin and Karniadakis [68]). All these methods, however, use the standard Galerkin method for the spatial discretization. The work of Bonhaus [9] uses a higher-order basis stabilized method (SUPG) for fluid dynamics simulations, however much of the emphasis was on 2-D problems, and compressible flows, and no turbulent flows were considered. The work presented here attempts to remedy this situation, and to quantify the potential benefit of using higher-order, stabilized finite element methods for fluid dynamics simulations.

In the present work, the spatial discretization of the stabilized formulation for the Navier-Stokes equations is carried out using a higher-order, hierarchical basis which is C^0 continuous between finite elements. The hierarchical basis used here is based on the abstract mesh data structure of Beall and Shephard [4], where basis functions are associated with the individual topological entities of the mesh. This type of basis construction was first introduced by Shephard *et al.* [66] (using the basis functions of Carnevali *et al.* [12]) who considered the basis functions to be associated with the mesh entities in a special way. Their mesh entity based hierarchical basis functions support non-uniform k-refinement of meshes of arbitrary element type, e.g. tetrahedral, hexahedral, and pyramidal, by employing an explicit decomposition of shape functions into element blends, ensuring the correct element support and entity level functions, giving the desired polynomial order on an entity. To gain this generality, we have dispensed with the traditional finite element mesh data structures consisting of only element nodal connectivity (see Hughes [42]) in favor of this more general and complete topological adjacency mesh representation. To maintain efficiency on large-scale problems, however, the abstract data structure is only currently used in the pre- and post-processing stages of the simulation, and is therefore not read by the analysis code. A compact data structure will be described that is simple to implement within existing finite element codes, as it represents a relatively straightforward generalization of the traditional data structures. Finally, note that we are using k to refer to the polynomial order of the finite element basis. This is in place of the more standard notation, p, which we reserve for the pressure variable.

Since we are interested in time-accurate large-eddy (LES) and direct numerical (DNS) simulations of turbulent flows (in addition to steady Reynolds averaged simulations) and we are using higher-order spatial discretization techniques, a time integrator of at least second order accuracy is deemed necessary. An implicit, second-order accurate time integrator is introduced for advancing the system of equations in time, coupled with Newton's method to solve the resulting nonlinear algebraic system in a predictor multi-corrector format. This time integrator has built into it a user controllable amount of numerical dissipation, which enables precise control over spurious, un-resolvable frequencies that may appear during the solution procedure (varying from simulation to simulation). After proving that the time integrator is second order accurate and unconditionally stable for a linear model problem, we apply it to the problem of vortex shedding behind a circular cylinder to study its temporal resolution properties.

Another key aspect of the present research is the use of parallel computers to effectively speed up computations. Finite element calculations are extremely well suited to parallel computing environments since much of the work is in computing element level integrals, and performing sparse matrix-vector products which both parallelize well. Several methods have been proposed which use parallel computers for finite element implementations, see, for example, Bastin [1], Johan and Hughes [47], Kennedy *et al.* [54], Bey *et al.* [7], and Biswas *et al.* [8]. Many of these implementations rely on some high level language, such as CM-Fortran (used by Johan and Hughes [47] as well as Kennedy *et al.* [54]), where interprocessor communication patterns are actually constructed by the compiler, requiring minimal coding effort, however performance is far from optimal (Bastin [1] showed these methods can take up to 15% of total CPU time for communication as opposed to 3% using pre-processed data structures). The current implementation is closely related to that used by Bastin [1], taking advantage of the MPI library for interprocessor communication using “message passing”. The use of message passing for these communications also enables the use of distributed computing environments which are quickly gaining popularity. To enable rapid communication of all information lying on partition boundaries during the analysis, the data structures necessary for parallel communication are pre-processed. This pre-processed data structure contains all the information necessary to carry out the interprocessor communication, which includes hierarchical degree-of-freedom information associated with mesh entities (edges and faces) that lie on the interprocessor boundary, as well as the linear vertex modes. Care has been taken to reduce communication cost by requiring that any pair of processors communicate no more than once.

Numerical simulations of the Navier-Stokes equations (through cubic polynomial order basis) will be presented that verify that nearly optimal convergence rates are observed for problems where analytical results are available, as well as for the linear advection-diffusion equation (through polynomial order 6). The method will then be applied to more complex (though still laminar) flow simulations which demonstrate a clear advantage of higher-order methods over the traditional, linear basis methods for the incompressible and compressible Navier-Stokes equations. Finally, an application will be made to a turbulent flow using direct numerical simulation (DNS). Although these turbulence results are still in the preliminary phase, the results are promising. It will be shown that, in the cases where a direct comparison is possible, the higher-order methods can provide the most cost-effective solutions in terms of both storage and computer time.

For several of the numerical simulations, a careful cost vs. accuracy study will be conducted to determine the cost-effectiveness of the hierarchical basis. This study will consider the cost with respect to various measures (including direct CPU time) which will quantify where improvements can be made to make the higher-order methods even more cost effective. The results presented will show that for steady problems, cubic basis simulations can be over four times more cost effective than the standard linear-basis methods. Results will also be presented for unsteady simulations, however, a quantitative comparison of the costs for these flows is difficult.

The work will be presented as follows. First, after providing necessary preliminary information relating to the abstract mesh data structures, Chapter 2 will describe the mesh entity based hierarchical basis to be used in the present work. This chapter will conclude with an application of the higher-order basis to the 2-D, linear advection-diffusion equation using the SUPG finite element formulation, and convergence rates will be verified. Chapter 3 develops the stabilized finite element formulation that will be used for incompressible flows, as well as the second order time integrator. Also presented in this chapter will be an application of the basis to compressible flows, and some examples will be provided. Several implementational aspects relating to the use of hierarchical basis functions will be discussed in Chapter 4, and numerical examples will be given in Chapter 5. Next, Chapter 6 will present an initial application of the methods to a direct numerical simulation of turbulence, and issues related to these types of simulations will be discussed. Conclusions and future research directions will be given in Chapter 7, and the contributions of the present research will be summarized.

CHAPTER 2
MESH ENTITY BASED HIERARCHICAL BASIS

The hierarchical basis functions used in the present work are based on the constructions of Shephard *et al.* [66] for specifying variable k-order meshes. These constructions are based on the topological hierarchy of mesh entities (vertices, edges, faces, and regions) which define the finite element mesh. Due to the restrictions of standard finite element data structures consisting only of nodal coordinates and element connectivity, variable k-order finite element meshes must rely on richer structures that allow the independent assignment of polynomial order over the elements as noted by Demkowicz *et al.* [19]. 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.* [12]). This chapter presents a detailed discussion of the finite element basis used in the present work. The description of a new compact mesh data structure that is used to maintain efficiency for large-scale problems will be presented in Chapter 4.

2.1 Abstract mesh data structure

In order to define the finite element basis, we will first introduce the abstract mesh data structure on which the element level basis will be defined (more detail on the mesh data structure used in the present work may be found in the work of Beall and Shephard [4]). The abstract mesh is represented by a data structure (mesh database) that maintains a complete set of adjacency relationships between the various entities in the finite element mesh, known as “mesh entities”. A mesh entity is defined as an individual topological object that is used to define the domain and boundary of a traditional finite element. These entities are of type: region, face, edge, and vertex. The first-order adjacencies between these mesh entities are as follows: a region is bounded by faces, a face is bounded by edges, and an edge is bounded by vertices.

We will refer to the abstract mesh data structure, including the adjacency relationships, as T_M. This mesh database is complemented by a set of functions which support general query operations such as first-order adjacencies (e.g. a function that returns the four vertices attached to a given tetrahedral mesh region), allows arbitrary data to be attached to mesh entities (or geometric model entities), and provides additional functionality. The mesh database is also a powerful tool for many other tasks relating to pre- and post-processing higher-order simulations (e.g. boundary conditions and parallel processing data structures rely heavily on the abstract mesh adjacency representation).

In addition to the mesh entity adjacencies (and their auxiliary functions), the mesh database also maintains a unique relationship between the finite element mesh and the geometric model of the underlying physical domain. This geometric model is represented in terms of “geometric model entities”, in analogy with mesh entities, and similar topological adjacency information is stored. The relationship between mesh and model is known as “classification” and defines the unique model entity that each mesh entity is classified on (more details of mesh-model classification may be found in Beall and Shephard [4]). Mesh-model classification is critical for the assignment of boundary conditions in a mesh-independent manner and greatly simplifies the application of boundary conditions (see Shephard [65]). As part of this work, a graphical user interface (GUI) was developed to enable the assignment of boundary conditions directly to the geometric model entities, which are subsequently inherited by the mesh entities based on their classification (there are generally *many* fewer model entities than mesh entities). Boundary conditions for a simulation are thus assigned without reference to a mesh, therefore, different meshes of the same physical model may be used without re-assigning boundary condition attributes.

In practice, the mesh database is a library of C++ classes that define the various mesh and model objects and have member functions that return the desired adjacency information. The concepts introduced here can be illustrated by the simple example C++ code fragment given in Program 2.1.1. First, the geometric model, `model.dmg`, and the mesh, `mesh.sms`, are loaded (it is presumed that the mesh is classified on this model). Then all regions associated with this mesh are visited and the list of vertices attached to the current region is retrieved. This vertex list may then be processed in any way, for example, coordinates or ID numbers could be collected into an array.

Program 2.1.1 Mesh database example

```
DiscreteModel *model = new DiscreteModel("model.dmg",0);

Mesh *mesh = MM_new(1,model);
M_load(mesh,"mesh.sms");

MRegion *region;
SimpleMeshRegionIter rIter = mesh->firstRegion();
while ( rIter(region) ) {
  SPList<MVertex*>  *vertices  = region->vertices();
  process list of vertices...
}
```

2.2 Finite element basis functions

To proceed with the definition of the element level basis, we first precisely define the finite element. The definition given here is similar to the standard finite element, although additional information is also included. Given the topological description of the mesh along with its adjacency relationships, T_M, we define:

Definition 2.1 *The closure of a finite element, denoted $\bar{\Omega}_e$, of dimension d_e, is defined as*

$$\bar{\Omega}_e = \{M_e^{d_e}, M_e^{d_e}\{M_j^{d_e-1}\}, \ldots, M_e^{d_e}\{M_j^0\}\}, \tag{2.1}$$

where $M_e^{d_e}$ represents mesh entity e of dimension d_e.

We have followed the notation of Beall and Shephard [4] for the mesh entity adjacencies as

$$M_e^{d_e}\{M_j^{d_j}\} \tag{2.2}$$

which is the j^{th} mesh entity of dimension d_j, bounding mesh entity e of dimension d_e. In other words, a finite element is a mesh region along with its lower order bounding mesh entities. For example, a tetrahedral finite element has four bounding vertices, six bounding edges, four bounding faces, and one region. Additional information, such as the direction a face (or edge) is used by a region (or face), is also maintained in the mesh database, and there are functions that return this information.

To construct the discrete, finite element solution, we expand the continuous quantities appearing in the weak form (given in the following chapter) in terms of a C^0 continuous, piecewise polynomial basis defined on each element (as described below). We define this element level basis with the aid of the piecewise polynomial space defined as:

Definition 2.2 *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, $N_a(\xi_i),\ a = 1 \ldots n_{es}$, contributed by the mesh entities in $\bar{\Omega}_e$. 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 associated with each bounding entity. The polynomial order assigned to each entity is used to compute the number of basis functions it will contribute. The local coordinate system, ξ_i, will be described below. 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 $\bar{\Omega}_e$, with the exception of vertex modes, which are linear, regardless of the basis order. The direct assignment of the polynomial order to each mesh entity, however, enables a straightforward extension to non-uniform k meshes and meshes of mixed-topology elements, and may also be useful to resolve strong gradients in a pre-determined spatial direction such as boundary layers where strong gradients occur in predictable directions.

2.2.1 Parametric coordinate systems

The basis functions are defined in terms of parametric coordinate systems, $\hat{\xi}_i$, associated with the individual mesh entities, as well as, ξ_i, the local coordinate of the element that is using the function. Each edge, face, and region in the finite element mesh has its own local coordinates. These coordinate systems need not be the same for all entities (of a given type) in the mesh, particularly when elements of different topologies are present, which is in contrast to Lagrange basis functions, that are defined solely in terms of a single element coordinate system. Since we will be dealing mainly with meshes composed entirely of tetrahedral elements, we will concentrate the discussion on "simplex" type co-

Topology	Parametric coordinates
Edge	$\hat{\xi}_1,\, \hat{\xi}_2 \equiv 1 - \hat{\xi}_1$
Face	$\hat{\xi}_1,\, \hat{\xi}_2, \hat{\xi}_3 \equiv 1 - \hat{\xi}_1 - \hat{\xi}_2$
Region	$\xi_1,\, \xi_2,\, \xi_3,\, \xi_4 \equiv 1 - \xi_1 - \xi_2 - \xi_3$

Table 2.1: Local simplex-type coordinate systems

ordinate systems; more details on coordinate systems useful for different types of element topologies can be found in Dey [21] and also Shephard *et al.* [66].

A general methodology has been developed by Shephard *et al.* [66] for the construction of k-version finite element meshes, which is used in the present work. Each basis function (for $k > 1$) is decomposed as

$$N(\xi_i) = \varphi(\xi_i(\hat{\xi}_j)) \times \psi(\xi_i) \tag{2.3}$$

where $\psi(\xi_i)$ is a blending function of fixed polynomial order ensuring that $N(\xi_i)$ has the correct global support, $\varphi(\xi_i(\hat{\xi}_j))$ is an entity level function giving the desired polynomial order on the entity, and $\xi_i(\hat{\xi}_j)$ represents the mapping from entity, $\hat{\xi}_j$, to element, ξ_i, coordinates. This decomposition allows for the efficient implementation of non-uniform k-order meshes as well as the use of meshes with mixed-topology elements. Since the blending function depends only on the element coordinate, it may differ for topologically different elements sharing the same mesh entity (which provides the correct polynomial order behavior, regardless of the topology of the bounding element). The decomposition of a shape function in terms of an entity level function and an element blend is illustrated in Figure 2.1 for a cubic basis function on a triangular element. In Figure 2.1, the element blend is shown in the upper left, and the entity level function (specific to the mesh edge) is shown on the upper right, their product is the resulting (cubic) shape function for the triangular element and is shown on the bottom.

To evaluate a basis function for a given element, the entity level function must first be mapped from the entity level coordinate (indicated by a hat) into the local coordinate system of this element. For simplex type elements, the local coordinate systems are shown in Table 2.1, where $0 \leq \xi_i, \hat{\xi}_i \leq 1$. The subscripts on the parametric coordinates indicate the local vertex where the coordinate takes on its maximal value, as given by the

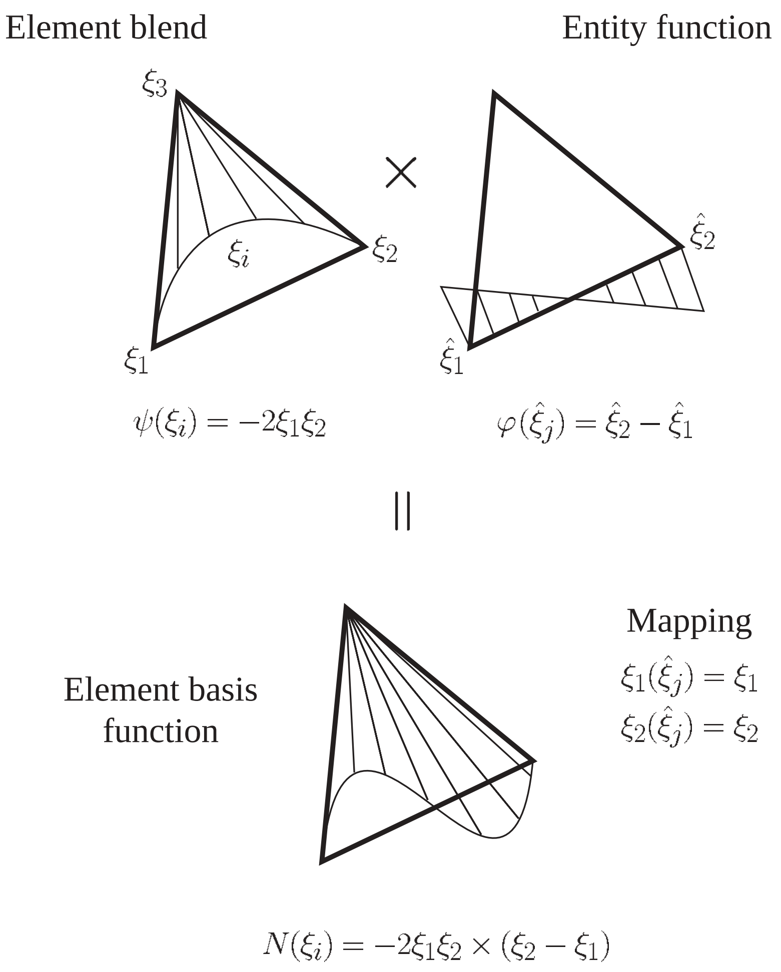

Element blend
Entity function
ξ_3
ξ_2
ξ_i
ξ_1
$\hat{\xi}_2$
$\hat{\xi}_1$
$\psi(\xi_i) = -2\xi_1\xi_2$
$\varphi(\hat{\xi}_j) = \hat{\xi}_2 - \hat{\xi}_1$
Mapping
$\xi_1(\hat{\xi}_j) = \xi_1$
$\xi_2(\hat{\xi}_j) = \xi_2$
Element basis function
$N(\xi_i) = -2\xi_1\xi_2 \times (\xi_2 - \xi_1)$

Figure 2.1: Shape function decomposition

vertex ordering in Figure 2.2. Also indicated in Figure 2.2 are the edge directions for a simplex-type element, crucial to the basis function constructions and the development of the compact data structures. For these parametric coordinates, the mapping from a bounding entity (edge or face) to the element (region) is relatively straightforward, involving only a possible sign change.

2.2.2 Blending functions

The blending function appearing in Equation 2.3, $\psi(\xi_i)$, depends only on the element, and ensures that each basis function has the correct global support (i.e. it must be zero on lower order mesh entities it does not bound). For tetrahedral regions, the blends will be defined as (see Dey [21]):

$$\psi(\xi_i) = -2\xi_i\xi_j \tag{2.4}$$

$$\psi(\xi_i) = \xi_i\xi_j\xi_m \tag{2.5}$$

for the edges and faces, respectively. The subscripts are defined by the local vertex ordering in Figure 2.2, for example $-2\xi_1\xi_2$ is the element blend for the edge between vertices 1 and 2. These choices of element blend are not the only possibility and additional ones are explored in Dey [21]. The blending functions are also useful for meshes comprised of multiple topology elements. Figure 2.1 shows the element blend and entity level function for a triangular element. Here, the blend is given by (2.4) and the entity level function (defined below) is $\varphi(\hat{\xi}_j) = \hat{\xi}_2 - \hat{\xi}_1$. Note that the mapping from edge (hat) coordinates to element coordinates before is also described below.

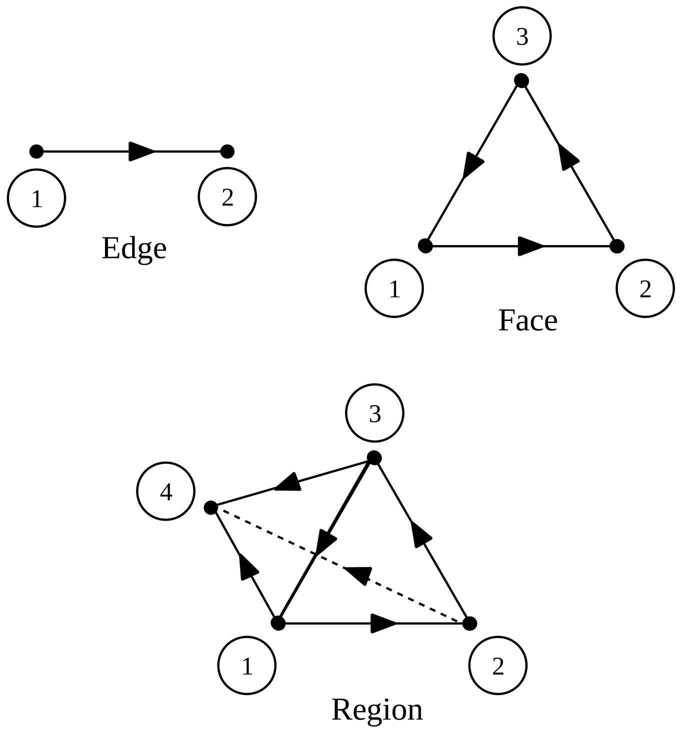


Figure 2.2: Local simplex-type vertex ordering and edge direction

2.2.3 Entity level functions

The entity level function (in Equation 2.3), $\varphi(\hat{\xi}_i)$ provides the desired polynomial order for a given entity's basis function. These functions can be comprised of any set of hierarchical basis functions, and in general are of order $(k-q)$, where q is the order of the blend, and k is the desired order. The hierarchical basis functions used here are taken from Carnevali *et al.* [12] and have been shown to yield better element level conditioning than standard Legendre polynomials for tetrahedral regions. The entity level functions are given by the following expressions (see also Dey [21]):

Edge ($k \geq 2$):

$$\varphi(\hat{\xi}_i) = \sum_{m=0}^{k-2} (-1)^m \frac{1}{m+1} \binom{k-2}{m} \binom{k-1}{m} \; \hat{\xi}_1^m \; \hat{\xi}_2^{\,k-2-m} \tag{2.6}$$

Face ($k \geq 3$):

$$\varphi(\hat{\xi}_i) = \sum_{i=0}^{\alpha_2-1} \sum_{j=0}^{\alpha_1-1} (\frac{-1}{2})^{i+j} \, i! \, j! \, (i+j)! \binom{\alpha_1-1}{j} \binom{\alpha_1}{j} \binom{\alpha_2-1}{i} \binom{\alpha_2}{i}$$
$$\times \frac{1}{\prod_{m=1}^{i+j} (m\,(\alpha_1+\alpha_2) - m\,(m-1)/2)} \; \hat{\xi}_1^{\,\alpha_1-1-j} \hat{\xi}_2^{\,\alpha_2-1-i} \tag{2.7}$$

where $\alpha_1,\ \alpha_2 = 1, \ldots, k-2$ and $\alpha_1+\alpha_2+\alpha_3 = k$.

Region ($k \geq 4$):

$$\varphi(\hat{\xi}_i) = \hat{\xi}_1 \, \hat{\xi}_2 \, \hat{\xi}_3 \, (A \times B \times C)$$

$$\begin{aligned}
A &= \sum_{i=0}^{\alpha_1-1} (-1)^i \, i! \binom{\alpha_1-1}{i} \binom{\alpha_1}{i} \frac{(2m+5-i)!}{2m+5} \hat{\xi}_1^{\,\alpha_1-1-i} \\
B &= \sum_{i=0}^{\alpha_2-1} i! \binom{\alpha_2-1}{i} \binom{\alpha_2}{i} \frac{(2n+3-i)!}{2n+3} \hat{\xi}_3^{\,\alpha_2-1-i} (\hat{\xi}_1 - 1)^i \\
C &= \sum_{i=0}^{\alpha_3-1} i! \binom{\alpha_3-1}{i} \binom{\alpha_3}{i} \frac{(2\alpha_3-i)!}{2\alpha_3} \hat{\xi}_3^{\,\alpha_3-1-i} (\hat{\xi}_1 + \hat{\xi}_2 - 1)^i
\end{aligned} \tag{2.8}$$

where $\alpha_1,\ \alpha_2,\ \alpha_3 = 1, \ldots, k-3$ and $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = k$ and also $m = \alpha_1 + \alpha_2 + \alpha_3 - 3,\ n = \alpha_2 + \alpha_3 - 2.$

In these formulas, $\binom{a}{b}$ represents the standard binomial coefficient, i.e. a items taken b at a time, which can also be expressed as

$$\binom{a}{b} = \begin{cases} \frac{a(a-1)\cdots(a-b+1)}{a!} & \text{if } b \geq 1 \\ 1 & \text{if } b = 0 \end{cases} \tag{2.9}$$

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

$$\phi^e(\xi_i, t) = \sum_{a=1}^{n_{es}} \phi_a(t) N_a(\xi_i) \tag{2.10}$$

where $\phi^e(\xi_i, t)$ is the finite element approximation of any variable (e.g. pressure or velocity) on element e and $\phi_a(t)$ are the desired coefficients with respect to the basis (since we are using a semi-discrete formulation, the coefficients depend on time). As mentioned above, the number of basis functions contributed by each mesh entity depends on the polynomial order assigned to the entity. When only C^0 continuity is desired, vertices contribute one basis function (equivalent to the standard linear Lagrange basis function). For an element level basis complete to order k, Table 2.2 provides the number of basis functions contributed by each mesh entity type. From this table, we can compute the total number of shape functions contributed by a tetrahedral element complete to order k as

$$\begin{aligned} n_{es} &= 4n_v + 6n_e + 4n_f + n_r \\ &= \frac{1}{6}(k+1)(k+2)(k+3). \end{aligned} \tag{2.11}$$

At this point we would like to point out some important differences between hierarchical and Lagrange basis functions. The main difference is that the hierarchical basis of order k is a subset of the basis of order $k+1$, i.e. $P_k(\bar{\Omega}_e) \subset P_{k+1}(\bar{\Omega}_e)$. This property greatly simplifies the generation of basis functions, and the varying of polynomial order. Hierarchical and Lagrange basis functions also differ as follows: for a given polynomial

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:	$n_r = \frac{1}{6}(k-1)(k-2)(k-3)$	$(k \geq 4)$

Table 2.2: Number of contributed shape functions

order, say N, all functions for the Lagrange basis are of order N, in contrast, the individual hierarchical basis functions will be of different order, however the complete polynomial order is still N. The polynomial order of each of the basis functions for each entity type is discussed in detail in Shephard *et al.* [66]. To get the total (global) number of basis functions, n_s (related to the total number of equations to be solved), 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 simply equals the number of "nodal points" in the mesh.) Another key difference is that the hierarchical basis function coefficients do not correspond to solution values at specific spatial locations (as they do for Lagrange elements), they are actually related to higher-order moments of the solution (and its derivatives) on the associated entity. This property makes many routine operations on finite element data more difficult to carry out. For example, post-processing and collecting turbulence statistics must rely on more advanced techniques when dealing with the higher-order basis functions. More will be said about these topics later.

2.3 Application: Advection-Diffusion equation

This section presents an application of the basis functions described above to the linear advection-diffusion equation, a simple model problem for the fluid flow equations containing many of their numerical difficulties. This application provides a good testbed for the hierarchical basis functions in the context of a well-understood linear problem. The finite element formulation used for this equation is the SUPG (Streamwise Upwind Petrov-Galerkin) method described in Franca and Frey [22]. The formulation will be thoroughly described for the Navier-Stokes equations in Chapter 3. For more details on the method, as well as the stability and convergence proofs, see Franca and Frey [22], also see Whiting *et al.* [76]. The formulation presented was proven stable and higher-order accurate by Franca and Frey [22] which implies that as the polynomial order of the basis is increased, the error in the finite element solution will converge at a rate equal to the interpolation error, or $O(h^{k+1})$. This important property is verified numerically below for polynomial order 1 through 6. An example will also be provided which shows the application to an advection-dominated case.

Consider the homogeneous-Dirichlet boundary value problem for the steady, advection-diffusion equation where $\phi = \phi(x_i)$ is sought such that

$$a_i \, \phi_{,i} - \kappa \, \phi_{,ii} = f \qquad \text{in} \quad \Omega, \tag{2.12}$$

$$\phi = g \qquad \text{on} \quad \Gamma_g \tag{2.13}$$

where Ω is the spatial domain of the problem and Γ_g is the portion of the boundary with prescribed essential boundary conditions (denoted by g), a_i are the Cartesian components of a divergence-free advective velocity field, $\kappa \; (> \; 0)$ is the diffusion coefficient, and $f(x_i)$ is a prescribed source term. Here and in what follows, the summation convention is in effect on repeated indices and an inferior comma denotes differentiation with respect to the variables following it.

2.3.1 Weak form

To proceed with the finite element discretization of (2.12), we first introduce the finite element approximation spaces for the advection-diffusion equation. Recall that Ω represent the physical spatial domain of the problem. $H^1(\Omega)$ represents the usual Sobolev space of functions with square-integrable values and derivatives on Ω. The domain Ω is discretized into n_{el} finite elements, $\bar{\Omega}_e$, which may be identified with the regions (or faces in 2 dimensional problems) in the mesh, T_M, along with their lower order bounding entities. With this, we can define the trial solution space as

$$\mathcal{S}_h^k = \{\phi | \phi \in H^1(\Omega), \phi|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e), \phi = \hat{g} \text{ on } \Gamma_g\}, \tag{2.14}$$

and the weight function space

$$\mathcal{W}_h^k = \{w | w \in H^1(\Omega), w|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e), w = 0 \text{ on } \Gamma_g\}, \tag{2.15}$$

where $P_k(\bar{\Omega}_e)$ (described in Section 2.2) is the piecewise polynomial space, complete to order $k \geq 1$, and Γ_g is the portion of the boundary where essential boundary conditions are prescribed. Also note that $\hat{g}$ represents the interpolation of the prescribed boundary conditions, g, in the finite element basis.

The discrete system of equations is derived by multiplying the original PDE (2.12), the so-called strong form, by a weight function (which is a member of the weight space), integrating over the physical domain, and performing integration by parts on the diffusive terms to reduce the continuity requirements on the solution space. This results in what is known as the standard Galerkin method applied to Equation (2.12). It is well known that the Galerkin method is unstable for advection dominated problems (see, for example, Brooks and Hughes [11]), so the weak form is modified. To the Galerkin term we add an SUPG stabilization term acting only in the "upwind" direction as originally introduced by Brooks and Hughes [11]. The weak form may then be stated as: find $\phi^{(h,k)} \in \mathcal{S}_h^k$ such that

$$B(w^{(h,k)}, \phi^{(h,k)}) = F(w^{(h,k)}), \qquad \forall w^{(h,k)} \in \mathcal{W}_h^k \tag{2.16}$$

with

$$\begin{aligned} B(\phi, w) &= (w, a_i \phi_{,i}) + (w_{,j}, \kappa \phi_{,j}) \\ &\quad + \sum_{e=1}^{n_{el}} (a_i w_{,i}, \tau(a_i \phi_{,i} - \kappa \phi_{,ii}))_{\bar{\Omega}_e} \end{aligned} \tag{2.17}$$

and

$$F(w) = (w, f) + \sum_{e=1}^{n_{el}} (\tau(a_i w_{,i}), f). \tag{2.18}$$

These equations show the Galerkin portion plus an SUPG stabilization parameter, τ, multiplying the residual of the strong form of the differential operator times the advective portion of the operator acting on the weight space. Note that the boundary integral resulting from the integration by parts has been omitted, which implies only Dirichlet or zero natural boundary conditions.

An important ingredient in these methods is the stabilization parameter, τ, which appears in the weak form (2.16). The particular form of τ we have chosen for the advection-diffusion equations is defined based on considerations of the error analysis, and is generalizable to higher polynomial order (see Franca *et al.*[23]); it takes the form:

$$\tau(Pe) = \frac{h_e}{2|\boldsymbol{a}|}\xi(Pe), \tag{2.19}$$

$$Pe = \frac{m_k|\boldsymbol{a}|h_e}{2\kappa}, \tag{2.20}$$

$$\xi(Pe) = \begin{cases} Pe & \text{if } 0 \le Pe < 1, \\ 1 & \text{if } Pe \ge 1 \end{cases} \tag{2.21}$$

$$m_k = \min(\frac{1}{3}, 2C_k), \tag{2.22}$$

$$\sum_{\bar{\Omega}_e \in T_M} h_e^2 \|w_{,ii}\|^2 \le C_k \|w_{,i}\|^2, \quad \forall w \in \mathcal{W}_h^k. \tag{2.23}$$

In the definition of τ, h_e represents a suitably chosen element diameter and C_k is a constant that depends on the polynomial order of the basis and represents a modification of τ for higher-order elements. The existence of such a constant follows from standard inverse estimates (see Ciarlet [16]), however, since it appears in the formulation, we need to have a numerical value for this constant. Some guidance as to how to choose this constant for some polynomial orders is provided by Harari and Hughes [31], however the results are somewhat limited. Our experience has shown, however, that the exact value for this parameter is not critical to the performance of the higher-order methods. It has been shown that the SUPG method with the stabilization parameter τ defined above is stable and converges at the same rate as the interpolation error for any polynomial order basis (see Franca and Frey [23]), the convergence rate given by $O(h^{k+1})$.

This is not the only possible definition of τ and another definition will be provided below in the context of the Navier-Stokes equations, where the situation is more complicated, this is also well documented in Shakib [61]. It is possible that the hierarchical basis functions may be applied to the development of new stabilized methods based on recent subgrid scale analysis (see Hughes [36] and related work on residual free bubbles by Rusśo [59] and Brezzi *et al.*[10]). These new formulations may be used to explicitly solve for the stabilization parameters based on local element level problems coupling neighboring elements. The solutions to the local problems may include higher-order polynomials in their basis, increasing the accuracy of the local problem. It is hoped that these new methods will yield more accurate simulations while maintaining the same desirable stability properties.

The weak formulation described above was implemented in Trellis, an object oriented framework for the numerical solution of PDE's by Beall and Shephard [5]. A complete description of this numerical framework, and many more details on its use, can be found in the work of Beall [3]. By using this framework, we are able to generate numerical simulations to the advection-diffusion equation by implementing (for the most part) only the element level contributions to the weak form and information pertaining to boundary condition specification. In this manner we can test the convergence of the formulation through polynomial order 6 (Trellis uses the same hierarchical basis functions described here). Details of the Trellis implementation for the advection-diffusion equation are included in Appendix A. It should be emphasized that it takes a relatively small amount of code (less than a total of 1000 lines) to implement the 2-D equation and gain access to all of the features of the Trellis analysis framework. These features include: hierarchical basis functions (through polynomial order 8), various linear solvers and preconditioners, geometry based boundary condition specification, and other features described in Beall [3]. Of particular importance to the present work is the ability to test the convergence of the stabilized formulation for the advection-diffusion equation for $k = 1 \ldots 6$. Many other implementational features are also in place in Trellis, e.g. numerical integration of the element integrals and force vectors, which greatly reduce the coding effort.

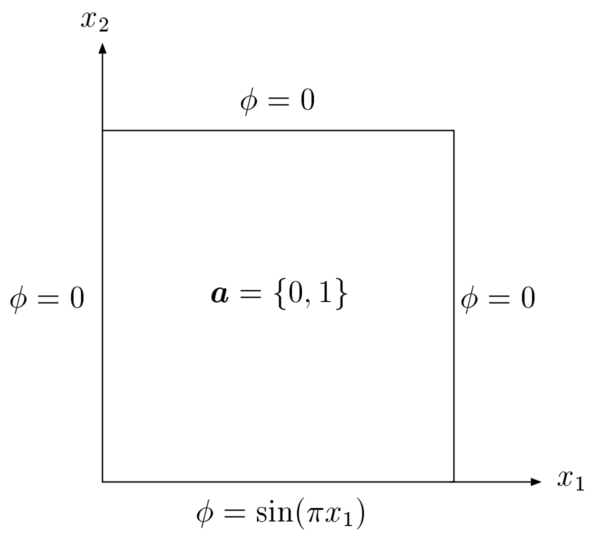


Figure 2.3: Geometry for one-direction advection-diffusion problem

2.3.2 Advection-diffusion example: convergence study

The finite element formulation for the advection-diffusion equation was programmed in Trellis as described in Appendix A. It should also be pointed out that for these simulations, the entire rich mesh data structure was used (as opposed to the compact data structures to be introduced later). We have chosen to present an example with an exact analytical solution which is used to verify the theoretical error estimates as the polynomial order of the basis is varied (additional examples may be found in Whiting *et al.* [76]). The geometry is described in Figure 2.3 where $0 \leq x_i \leq 1$, and the exact solution (readily obtained using separation of variables) is given by

$$\phi(x_i) = \frac{1}{e^{m_2 - m_1} - 1}(e^{m_2 - m_1} e^{m_1 x_2} - e^{m_2 x_2}) \sin(\pi x_1) \tag{2.24}$$

where

$$m_{1,2} = \frac{1 \mp \sqrt{1 + 4\kappa^2 \pi^2}}{2\kappa} \tag{2.25}$$

Figure 2.4 illustrates the convergence to the exact solution for $k = 1 \ldots 6$ (and

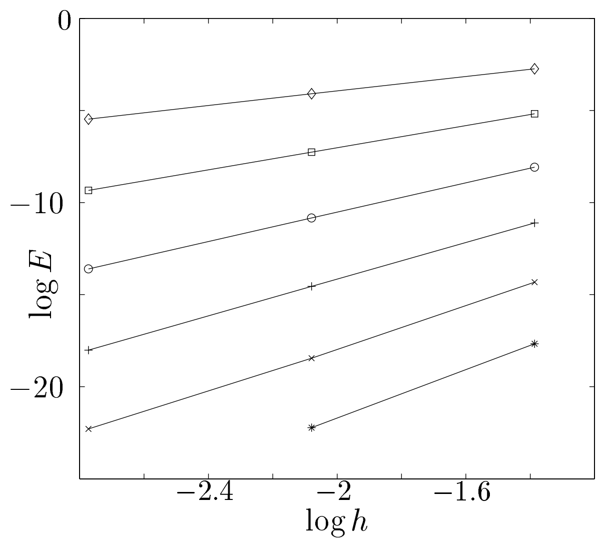


Figure 2.4: Normalized L^2 error $vs.$ h for $k = 1 \ldots 6$

$\kappa = 1.0$) in terms of the normalized L^2 error (evaluated using numerical quadrature):

$$E^2 = \frac{\int_\Omega (\phi - \phi^{(h,k)})^2 \; dx}{\int_\Omega \phi^2 \; dx} \tag{2.26}$$

The slopes of these curves are given in Table 2.3 and clearly obey the optimal convergence results (or $O(h^{k+1})$). The only slight exception is $k = 6$, where the error is close to the machine precision and therefore questionable. Note that, over the range of interest to accuracy, the curves do not cross, indicating that even on coarse meshes there is great advantage to using higher polynomial order.

k	symbol	slope
1	$\diamond$	2.0
2	$\circ$	3.0
3	◻	4.0
4	+	5.0
5	x	5.8
6	$*$	6.6

Table 2.3: Convergence rates for one directional advection-diffusion

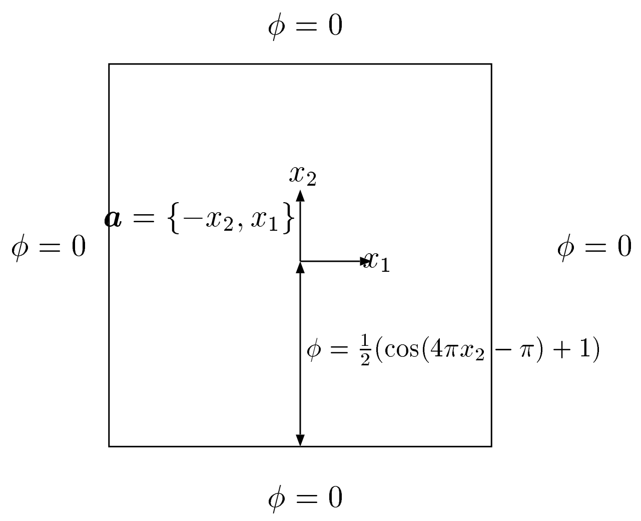

$\phi = 0$
$\phi = 0$
$\phi = 0$
$\phi = 0$
x_2
$\boldsymbol{a} = \{-x_2, x_1\}$
x_1
$\phi = \frac{1}{2}(\cos(4\pi x_2 - \pi) + 1)$

Figure 2.5: Geometry for rotating field problem

2.3.3 Advection-diffusion example: Advection in a rotating field

This problem illustrates an application to a problem with a large Peclet number (advection dominated), with $\kappa = 10^{-6}$. It is solved on a fixed coarse mesh of 32 uniform triangular elements (5 vertices across the box) for $k = 1 \dots 6$. The geometry and problem parameters are shown in Figure 2.5, where the problem domain is given by $-1/2 \leq x_i \leq 1/2$. Figure 2.6 presents line plots of $\phi(x_1 = 0, x_2)$ for $0 \leq x_2 \leq 1/2$. Symbols for the various polynomial orders may be found in Table 2.3. Since κ is small, the exact solution is approximately given by the data along the internal line, rotated around the center of the domain. This plot demonstrates the ability of the high order simulations to almost exactly represent the solution, even with only two edges (3 vertices) across the profile.

2.4 Chapter summary

This chapter presented the hierarchical basis that will be used for the fluid dynamics simulations presented in the following chapters. The shape function decomposition in terms of entity level function and element blend was discussed in detail, as was the use of the abstract mesh data structure, on which the constructions are based. The chapter concluded with an application of the hierarchical basis to the linear advection-diffusion equation using the SUPG finite element formulation. We then verified the convergence rate of the formulation through polynomial order 6, for a problem with a known analytical

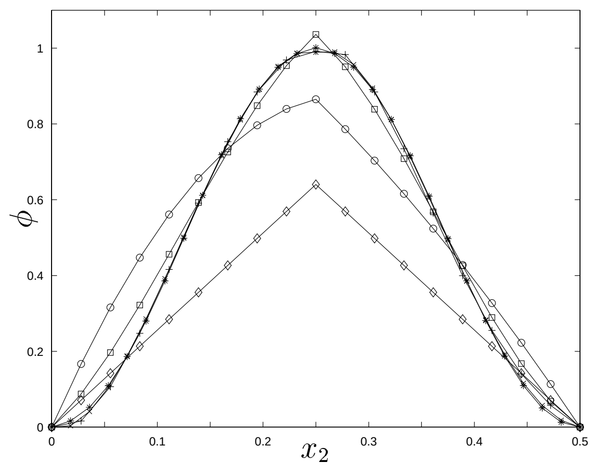


Figure 2.6: Line plots of ϕ for $k = 1 \ldots 6$

solution. An additional problem was also described to further demonstrate the accuracy of the formulation. In the following chapters, the hierarchical basis will be used for numerical simulations of both compressible and incompressible flow, and its accuracy and cost effectiveness will be demonstrated.

CHAPTER 3
NUMERICAL SOLUTION OF THE NAVIER-STOKES EQUATIONS

This chapter presents the finite element formulation for the Navier-Stokes equations. Although the main thrust of this research is with incompressible flow, an application will be made at the end of the chapter to the compressible system of equations. A semi-discrete finite element formulation that has restored conservation properties is presented which uses the hierarchical basis functions for the spatial discretization. Stabilized finite element formulations have been used by several researchers and have been shown to be robust, accurate, and stable on a variety of examples from steady and unsteady laminar flows to large eddy simulations (LES) and Reynolds averaged simulations of complex turbulent flows (see, for example, Jansen [44], Jansen [43], Tezduyar *et al.* [73], Hughes and Jansen [39], Bastin [1], and Taylor *et al.* [71]). The temporal discretization is based on the generalized-α method of Chung and Hulbert [15], generalized to first-order systems in Jansen *et al.* [46]. This new implicit time integrator is proven to be second-order accurate (on linear model problems) and contains a user specified amount of numerical dissipation.

Stabilized finite element methods have been proven to be stable and higher-order accurate for a linear advective-diffusive system (the closest model problem to the Navier-Stokes equations) in Hughes *et al.*[38], for the linearized incompressible Navier-Stokes equations in Franca and Frey [22], and for a representative nonlinear problem (the Burgers equation) in Johnson and Szepessy [51]. The higher-order accuracy properties, as well as the robustness on complex flows has motivated our choice of finite element formulation. We first present the strong form of the incompressible Navier-Stokes equations, followed by a description of the semi-discrete, stabilized finite element formulation used to discretize the spatial portion of the associated weak form. The generalized-α method time integrator is then introduced to integrate the system of ordinary differential equations resulting from the spatial integration. This time integrator is proven stable and second order accurate for a linear model problem. An example is presented to explore the characteristics of the time integrator in the context of a well understood flow. An application of the hierarchical basis to the compressible Navier-Stokes equations is then given, and examples are presented to demonstrate the optimal convergence and application to a more complicated flow. Finally, the chapter concludes with a discussion of the numerical evaluation of the diffusive flux terms that appear in the formulation, and methods will be presented for use with both linear and higher-order basis functions.

3.1 Incompressible strong form

Consider the application of the mesh entity based hierarchical basis functions (described in Chapter 2) to the time-dependent, incompressible Navier-Stokes equations. First, consider the strong (or differential) form of the continuity and momentum equations written in the so-called advective form (see Gresho and Sani [29])

$$u_{i,i} = 0 \tag{3.1}$$

$$\dot{u}_i + u_j u_{i,j} = -p_{,i} + \tau_{ij,j} + f_i \tag{3.2}$$

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

$$\tau_{ij} = \nu(u_{i,j} + u_{j,i}) \tag{3.3}$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity, and the summation convention is used throughout (sum on repeated indices). We have chosen to write the diffusive terms in the stress-divergence form, which gives rise to a more meaningful set of natural boundary conditions. This system of equations is supplemented with an appropriate set of prescribed boundary conditions on $\Gamma = \partial\Omega$. The incompressible Navier-Stokes equations can be written in many equivalent forms (for the continuous system) which are not necessarily equivalent when discretized. A complete description of the various forms of the equations and the strengths and weaknesses of each, as well as a complete discussion of boundary conditions, are described in the book by Gresho and Sani [29].

3.2 Weak form – Finite element discretization

Finite element methods are based on the weak form (or integral form) of the Navier-Stokes equations (3.1) and (3.2) which is obtained by dotting the entire system from the left by a vector of weight functions and integrating over the spatial domain. The diffusive term, pressure term, and continuity equation are all integrated by parts. The diffusive term is integrated by parts to reduce continuity requirements, otherwise we would have second derivatives on our solution space. The pressure term is integrated by parts to provide symmetry with the continuity equation which is integrated by parts to provide discrete conservation of mass. The consequences of not integrating the pressure term by parts are discussed in detail in Gresho and Sani [29] pages 449–450.

The finite element formulation is based on finite dimensional subspaces of the continuous weight and solution spaces. 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. Subsequently Ω is discretized into n_{el} finite elements, $\bar{\Omega}_e$, as defined above. With this, we can define the discrete trial solution and weight spaces for the semi-discrete formulation as

$$\boldsymbol{\mathcal{S}}_h^k = \{\boldsymbol{v} | \boldsymbol{v}(\cdot, t) \in H^1(\Omega)^N, t \in [0,T], \boldsymbol{v}|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^N, \boldsymbol{v}(\cdot, t) = \hat{\boldsymbol{g}} \text{ on } \Gamma_g\}, \tag{3.4}$$

$$\boldsymbol{\mathcal{W}}_h^k = \{\boldsymbol{w} | \boldsymbol{w}(\cdot, t) \in H^1(\Omega)^N, t \in [0,T], \boldsymbol{w}|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^N, \boldsymbol{w}(\cdot, t) = \boldsymbol{0} \text{ on } \Gamma_g\}, \tag{3.5}$$

$$\mathcal{P}_h^k = \{p | p(\cdot, t) \in H^1(\Omega), t \in [0,T], p|_{x \in \bar{\Omega}_e} \in P_k(\bar{\Omega}_e)\} \tag{3.6}$$

where $P_k(\bar{\Omega}_e)$ is as defined in Definition 2.2. Here, $\hat{\boldsymbol{g}}$ represents an approximation to the prescribed boundary condition in the finite element basis. Let us emphasize that the local approximation space, $P_k(\bar{\Omega}_e)$, is the same for both the velocity and pressure variables (although this is not necessary, it is computationally convenient, especially when working with higher-order discretizations). This equal-order interpolation is possible due to the stabilized nature of the formulation to be introduced below, without which, attention must be paid to the Babuška-Brezzi condition. Note that here and throughout, we have omitted the superscript h that is normally included in the discrete representation of the continuous variables, as in $u_i^{(h,k)}$, for notational simplicity. Where there is any chance of confusion, the full notation is retained.

The stabilized formulation used in the present work is based on that described by Taylor *et al.* [71] generalized 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 (3.1) as:

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

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

$$\begin{aligned} B_G(w_i, q; u_i, p) = \int_\Omega & \{w_i\left(\dot{u}_i + u_j u_{i,j} - f_i\right) + w_{i,j}\left(-p\delta_{ij} + \tau_{ij}\right) - q_{,i} u_i\} dx \\ + \int_{\Gamma_h} & \{w_i\left(p\delta_{in} - \tau_{in}\right) + q u_n\}\, ds \end{aligned} \tag{3.7}$$

for all $\boldsymbol{w} \in \boldsymbol{\mathcal{W}}_h^k$ and $q \in \mathcal{P}_h^k$. 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 all the weight coefficients are arbitrary, this gives us a separate equation for each of the i components (and for each of the basis functions). The standard Galerkin method is well known to be unstable for advection-dominated flows (see Brooks and Hughes [11]) and in the diffusion dominated limit for equal-order interpolation of the velocity and pressure, i.e. the Babuška-Brezzi condition. Stabilized methods are well-known to address both of these issues (see Brooks and Hughes [11] and Hughes *et al.* [37], respectively). To remedy both of these situations we add additional stabilization terms as follows:

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

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

$$\begin{aligned} B(w_i, q; u_i, p) = B_G(w_i, q; u_i, p) \\ + \sum_{e=1}^{n_{el}} \int_{\bar{\Omega}_e} \{\tau_M(u_j w_{i,j} + q_{,i})\mathcal{L}_i + \tau_C w_{i,i} u_{j,j}\}\, dx \\ + \sum_{e=1}^{n_{el}} \int_{\bar{\Omega}_e} \{w_i \overset{\Delta}{u}_j u_{i,j} + \bar{\tau} \overset{\Delta}{u}_j w_{i,j} \overset{\Delta}{u}_k u_{i,k}\}\, dx \end{aligned} \tag{3.8}$$

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

$$\mathcal{L}_i = \dot{u}_i + u_j u_{i,j} + p_{,i} - \tau_{ij,j} - f_i \tag{3.9}$$

The second line in the stabilized formulation, (3.8), represents the typical SUPG stabilization added to the Galerkin formulation for the incompressible set of equations (see Franca and Frey [22]). The first term in the third line of (3.8) was introduced by Taylor *et al.* [71] to overcome the lack of momentum conservation introduced as a consequence of the momentum stabilization in the continuity equation. The second term on this line 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 (3.8) which leaves only boundary terms if we choose

$$\overset{\Delta}{u}_i = -\tau_M \mathcal{L}_i \tag{3.10}$$

which may be identified with a modified, conservation-restoring, advective velocity. This term must itself be stabilized since it is an advective type term which will lead to advective instabilities. The stabilization parameters for continuity and momentum are defined as

$$\tau_M = \frac{\rho}{\sqrt{c_1/{\Delta_t}^2 + u_i g_{ij} u_j + c_2 \nu^2 g_{ij} g_{ij}}} \tag{3.11}$$

$$\tau_C = \frac{1}{8\tau_M \mathrm{tr}(g_{ij})} \tag{3.12}$$

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

$$\bar{\tau} = \frac{\rho}{\sqrt{\overset{\Delta}{u}_i g_{ij} \overset{\Delta}{u}_j}} \tag{3.13}$$

where c_1 and c_2 are defined based on considerations of 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 coordinates. It should be noted that for tetrahedral elements, this mapping depends on the orientation of the element, and therefore must be corrected to create an invariant element length-scale by permuting the possible choices of orientation. This term may be identified with the element length-scale, and is hence a mesh dependent parameter. These stabilization parameters are related to those proposed by Shakib [61] and were also used (in a slightly different form) by Taylor *et al.* [71]. The constant c_2 is a modification 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 [22]. There is some guidance as to how to select this parameter, however, experience has shown the method to be relatively insensitive to its choice. Currently we use $c_2 = 36, 60, 128$ for linear, quadratic, and cubic basis, respectively, for the modification, which has provided good results in all cases presented. The parameter c_1 is related to the temporal portion of the stabilization, and we have selected it to be 4 for most problems.

To derive a discrete system of algebraic 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 (c.f. Equation 2.10). Gauss quadrature of the spatial integrals results in a system of first-order, nonlinear differential-algebraic equations which can be written as

$$\boldsymbol{R}_A(\boldsymbol{u}_i, \dot{\boldsymbol{u}}_i, \boldsymbol{p}) = 0, \quad A = 1 \ldots n_s \tag{3.14}$$

where we have assumed the coefficients of the weight functions to be arbitrary, indicated by the index A, and $\boldsymbol{u}_i, \dot{\boldsymbol{u}}_i$, and $\boldsymbol{p}$ are vectors of the basis coefficients for the discrete representations of these flow variables. The generalized-α method described below is used to solve this nonlinear system in a predictor corrector format utilizing Newton's method.

3.3 Generalized-α time integrator

While several methods have been proposed for the integration of the Navier-Stokes system (both semi-discrete as well as space-time), there has yet to emerge a clear favorite. For example, space-time finite element methods where proposed and analyzed by Shakib *et al.* [62, 63] and expanded and used extensively by Tezduyar *et al.* [74, 75, 49, 50]. Here, as the name implies, the weight and solution space are both given a temporal dependence in addition to the usual spatial dependence. While these methods have yielded very accurate results, the cost has only been justifiable on problems with a moving domain such as free surface flows and/or deforming spatial domains that account for moving solid boundaries. In these cases, the additional cost of space-time methods is put to good use by providing a consistent tracking of the moving boundary.

In cases where the boundary is not moving, semi-discrete methods remain in favor (see Behr *et al.*[6]). Part of the attraction to semi-discrete methods is their long history of use in computational solid dynamics. Many algorithms have been proposed, analyzed and even designed to provide particular behaviors needed in particular conditions. Of particular interest is the behavior of these algorithms in situations where a broad range of temporal scales are present, such as the case of turbulent flows. In this case, the time step is often chosen (out of necessity) such that certain frequencies are only marginally resolved or perhaps even completely unresolved. Given the nonlinearities present in most interesting engineering systems, it is of great importance to ensure that there is temporal damping for frequencies beyond the chosen resolution level. However, it is equally important that this damping not effect the frequencies within the chosen resolution level, leading to a degradation of accuracy (see Jansen *et al.* [46]).

3.3.1 Analysis of Generalized-α Method

The system of nonlinear differential-algebraic equations resulting from the spatial integration of the stabilized Navier-Stokes formulation is nonlinear, making it unwieldy for analysis. Much insight into the properties of the time integrator may be gained by studying the application to a simple, linear problem. As described in Hughes [42] the linearized version of the nonlinear system can be un-coupled into many single degree of freedom problems by diagonalizing the linear operator in terms of its orthogonal eigen-vectors. This procedure results in the following model problem

$$\dot{y} = \lambda y \tag{3.15}$$

where λ is the eigenvalue associated with the chosen mode.

We proceed to introduce the generalized-α method for integrating (3.15) from t_n to t_{n+1} (i.e. $\Delta_t = t_{n+1} - t_n$)

$$\dot{y}_{n+\alpha_m} = \lambda y_{n+\alpha_f} \tag{3.16}$$

$$y_{n+1} = y_n + \Delta_t \dot{y}_n + \Delta_t \gamma (\dot{y}_{n+1} - \dot{y}_n) \tag{3.17}$$

$$\dot{y}_{n+\alpha_m} = \dot{y}_n + \alpha_m (\dot{y}_{n+1} - \dot{y}_n) \tag{3.18}$$

$$y_{n+\alpha_f} = y_n + \alpha_f (y_{n+1} - y_n) \tag{3.19}$$

where α_m, α_f and γ are, at this point, free parameters. These four equations can be rewritten in the form

$$\boldsymbol{a}\boldsymbol{y}_{n+1} = \boldsymbol{b}\boldsymbol{y}_n, \qquad \text{or} \qquad \boldsymbol{y}_{n+1} = \boldsymbol{c}\boldsymbol{y}_n \tag{3.20}$$

where the solution vector at t_n is given by $\boldsymbol{y}_n = \{y_n, \Delta_t \dot{y}_n\}^T$ (similarly for $\boldsymbol{y}_{n+1}$) and the amplification matrix $\boldsymbol{c} = \boldsymbol{a}^{-1}\boldsymbol{b}$ is

$$\boldsymbol{c} = \frac{1}{d}\begin{bmatrix} \alpha_m - (\alpha_f - 1)\gamma\Lambda & \alpha_m - \gamma \\ \Lambda & \alpha_m - 1 + \alpha_f\Lambda(1-\gamma) \end{bmatrix} \tag{3.21}$$

where $\Lambda = \lambda\Delta_t$ and $d = \alpha_m - \alpha_f\gamma\Lambda$. It is fairly straightforward to show (see Hughes [42]) that

$$y_{n+1} = \mathbf{trace}(\boldsymbol{c})y_n - \mathbf{det}(\boldsymbol{c})y_{n-1} \tag{3.22}$$

If we further substitute a Taylor series expansion of y_{n+1} and y_{n-1} about y_n in time we find that second order accuracy can be obtained so long as

$$\gamma = \frac{1}{2} + \alpha_m - \alpha_f \tag{3.23}$$

This is the same result found by Chung and Hulbert [15] for a second order system.

Stability can be assessed by looking at the eigenvalues of $\boldsymbol{c}$. To make our model problem reflective of both advective and diffusive phenomena requires that Λ be complex. We are interested in proving stability for the left half of the complex plane since we will assume positive diffusive coefficients in our fluid dynamics problems. Stability will be attained so long as the modulus of each eigenvalue is less than or equal to one. The expressions for the eigenvalues of (3.21) are too lengthy to express here. Instead we will illustrate the stability constraints on α_m and α_f through the limiting values of Λ.

First consider the case when the time step is taken to be very small. Regardless of the value of λ, Λ vanishes, and the eigenvalues of $\boldsymbol{c}$ in this limit are

$$\lim_{\Delta_t \to 0} \xi_i = \left\{1 - \frac{1}{\alpha_m}, 1\right\} \tag{3.24}$$

from which we may deduce that stability requires

$$\alpha_m \geq \frac{1}{2} \tag{3.25}$$

We next consider the limit of an infinite time step for any eigenvalue in the left complex half plane (i.e Λ tending to complex infinity). The eigenvalues of $\boldsymbol{c}$ in this limit are

$$\lim_{\Delta_t \to \infty} \xi_i = \left\{ \frac{-1 + 2(\alpha_m - \alpha_f)}{1 + 2(\alpha_m - \alpha_f)}, 1 - \frac{1}{\alpha_f} \right\} \tag{3.26}$$

from which we may deduce that stability also requires

$$\alpha_m \geq \alpha_f \geq \frac{1}{2} \tag{3.27}$$

Again, this is the same result obtained by Chung and Hulbert [15] for the second order system. Since they had an additional eigenvalue (and an additional parameter in their method) they had a third constraint that is not present here.

While having two parameters free in the method has a certain appeal, we recall that our goal was to find a method with strict control of high frequency damping. Therefore it is enlightening to express the two parameters α_m and α_f in terms of the spectral radius of an infinite time step or maximum absolute value of the eigenvalue as Δ_t tends to infinity, what Chung and Hulbert referred to as ρ_∞,

$$\rho_\infty = \lim_{\Delta_t \to \infty} \max(\xi_1, \xi_2) \tag{3.28}$$

By requiring the ρ_∞ from each eigenvalue in (3.26) to take on the same value we can express α_m and α_f in terms of ρ_∞, viz.

$$\alpha_m = \frac{1}{2}\left(\frac{3 - \rho_\infty}{1 + \rho_\infty}\right), \qquad \alpha_f = \frac{1}{1 + \rho_\infty} \tag{3.29}$$

thereby defining a second-order accurate family of methods with a specified high frequency damping.

The importance of casting the parameters in this way is that one has precise control over the damping of frequencies that are high relative to the resolution level. If ρ_∞ is chosen to be zero, the method is said to annihilate the highest frequency in one step (only for a linear problem). This method has the same spectral stability as Gear's two step backward difference method [25]. If ρ_∞ is chosen to be one, then the highest frequency (as well as all others) are preserved (for the linear problem). This method corresponds to the midpoint rule which is equivalent to the trapezoidal rule for linear problems.

For linear problems with all frequencies resolved, the midpoint rule has the very nice property of introducing no damping, regardless of the time step. However, when the eigenvalue λ is purely imaginary (i.e. when the flow is convection dominated) with the modulus equal to one, the eigenvalue $\lim_{\Lambda\to\infty}\xi_1 = -1$, which has the effect of causing the solution to switch sign on each step. This behavior is clearly unacceptable. In these cases it is important to have ρ_∞ strictly less than one so that high frequencies do not spoil long term integrations. The example of vortex shedding from a circular cylinder (presented below) will illustrate this effect.

3.3.2 Generalized-α Method for the Navier-Stokes Equations

In addition to the application of the time integrator to a nonlinear system, the application of the generalized-α method to the Navier-Stokes equations introduces the difficulty of integrating the pressure in time, which has no explicit temporal dependence. This type of a system is technically referred to as a differential-algebraic equation (or DAE), and the theory for integrating such systems is quite involved (see Gresho and Sani [29]). The pressure here is not really integrated in time, it is just iterated to remain consistent with the velocity which is integrated with the generalized-α method. A comprehensive study of the many alternative methods for temporal integration of the DAE's and their implications, though interesting, is well beyond the scope of the present research. A more complete discussion of such topics may be found in the work of Gresho and Sani [29]. The other primary difficulty in extending the work from the previous section to the full Navier-Stokes equations is the nonlinearity that is introduced. We first recall from Section 3.2 that, once spatially discretized, the momentum and continuity equations may be written in the form:

$$\boldsymbol{R}_A(\boldsymbol{u}_i, \dot{\boldsymbol{u}}_i, \boldsymbol{p}) = 0, \quad A = 1 \ldots n_s \tag{3.30}$$

which will be the starting point of the application. Note that this system can also be written as:

$$\begin{pmatrix} \boldsymbol{R}_m \\ \boldsymbol{R}_c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.31}$$

where $\boldsymbol{R}_m$ and $\boldsymbol{R}_c$ represent the residuals of the momentum and continuity equations, respectively (i.e. the portions of $\boldsymbol{R}$ multiplying w_i and q, respectively), and we have dropped the subscript A, related to the weight space, to clarify the presentation.

With these considerations in mind, application of the method introduced in Section 3.3.1 yields the following set of equations describing the time integration algorithm. The first equation is the nonlinear residual with the velocity and acceleration evaluated at the intermediate time steps $t_{n+\alpha_f}$ and $t_{n+\alpha_m}$, respectively

$$\boldsymbol{R}(\boldsymbol{u}_{n+\alpha_f}, \dot{\boldsymbol{u}}_{n+\alpha_m}, p_{n+1}) = 0 \tag{3.32}$$

followed by the update equations relating the velocity to its time derivative,

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta_t \dot{\boldsymbol{u}}_n + \gamma \Delta_t (\dot{\boldsymbol{u}}_{n+1} - \dot{\boldsymbol{u}}_n) \tag{3.33}$$

and finally the equations that relate the temporal locations n and $n+1$ to $n+\alpha_m$ and $n+\alpha_f$

$$\dot{\boldsymbol{u}}_{n+\alpha_m} = \dot{\boldsymbol{u}}_n + \alpha_m (\dot{\boldsymbol{u}}_{n+1} - \dot{\boldsymbol{u}}_n) \tag{3.34}$$

$$\boldsymbol{u}_{n+\alpha_f} = \boldsymbol{u}_n + \alpha_f (\boldsymbol{u}_{n+1} - \boldsymbol{u}_n) \tag{3.35}$$

Here we have introduced $\boldsymbol{R}$ to be the vector of nodal values of the nonlinear residual including both the continuity and momentum equations. The nonlinearities are best handled by introducing a predictor-multicorrector algorithm similar to those proposed by Brooks and Hughes [11]. By making a prediction of the solution and its time derivative at time t_{n+1}, we start the algorithm. Since we will be making multiple corrections, we introduce a superscript (inside parentheses) to represent the corrector iteration number. In this notation our predictor is initialized with an iteration count of zero and is given by

$$p_{n+1}^{(0)} = p_n \tag{3.36}$$

$$\boldsymbol{u}_{n+1}^{(0)} = \boldsymbol{u}_n \tag{3.37}$$

$$\dot{\boldsymbol{u}}_{n+1}^{(0)} = \frac{\gamma - 1}{\gamma}\dot{\boldsymbol{u}}_n \tag{3.38}$$

where (3.36) and (3.37) predict that the solution will be the same as it was at the previous time step and (3.38) is the time derivative at t_{n+1} that is consistent with (3.33) (i.e. the predictor that preserves second order accuracy). Other choices of predictors are also possible.

After making the prediction, the algorithm enters a loop of multi-corrector passes with i initialized to zero. The first operation within the loop is the calculation of velocity at $t_{n+\alpha_f}$ and the acceleration at $t_{n+\alpha_m}$

$$\boldsymbol{u}_{n+\alpha_f}^{(i)} = \boldsymbol{u}_n + \alpha_f(\boldsymbol{u}_{n+1}^{(i-1)} - \boldsymbol{u}_n) \tag{3.39}$$

$$\dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)} = \dot{\boldsymbol{u}}_n + \alpha_m(\dot{\boldsymbol{u}}_{n+1}^{(i-1)} - \dot{\boldsymbol{u}}_n) \tag{3.40}$$

These quantities enable the evaluation of $\boldsymbol{R}^{(i)}(\boldsymbol{u}_{n+\alpha_f}^{(i)}, \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}, p_{n+1}^{(i)})$ which, for small i, can be expected to be far from its desired value of $\mathbf{0}$. To find an improvement to the current values of (3.39) and (3.40) we use a Newton type linearization of $\boldsymbol{R}^{(i)}$ with respect to the acceleration, $\dot{u}_i$, for both the momentum and continuity residuals which yields a matrix problem to solve for the acceleration and pressure increments, given by

$$\begin{pmatrix} \boldsymbol{K}^{(i)} & \boldsymbol{G}^{(i)} \\ \boldsymbol{D}^{(i)} & \boldsymbol{C}^{(i)} \end{pmatrix} \begin{pmatrix} \Delta\dot{\boldsymbol{u}}_{n+1}^{(i)} \\ \Delta\boldsymbol{p}_{n+1}^{(i)} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{R}_m^{(i)} \\ \boldsymbol{R}_c^{(i)} \end{pmatrix} \tag{3.41}$$

which is solved for each corrector pass and the solution is updated according to

$$\dot{\boldsymbol{u}}_{n+1}^{(i+1)} = \dot{\boldsymbol{u}}_{n+1}^{(i)} + \Delta\dot{\boldsymbol{u}}_{n+1}^{(i)} \tag{3.42}$$

$$\boldsymbol{u}_{n+1}^{(i+1)} = \boldsymbol{u}_{n+1}^{(i)} + \gamma\Delta_t\Delta\dot{\boldsymbol{u}}_{n+1}^{(i)} \tag{3.43}$$

$$p_{n+1}^{(i+1)} = p_{n+1}^{(i)} + \Delta p^{(i)} \tag{3.44}$$

and i is incremented. The definition and numerical evaluation of the sub-matrices $\boldsymbol{K}^{(i)}$, $\boldsymbol{G}^{(i)}$, $\boldsymbol{D}^{(i)}$, and $\boldsymbol{C}^{(i)}$ is discussed below. If $i < i_{\max}$ the algorithm returns to solve (3.41) thus initiating the next corrector pass. Otherwise, the solution at time step t_{n+1} is updated, and the algorithm proceeds to the next time step. This completes the step from $t_n \rightarrow t_{n+1}$. If more time steps are required, n is incremented and the algorithm returns to the prediction phase for the next step (i.e. (3.39) and (3.40)). The entire algorithm is summarized in Algorithm 3.3.1.

The linear system of equations, (3.41), is difficult, and special care should be taken in setting up and solving it. The linear algebra solver of Shakib [64] (a highly optimized linear algebra package for the incompressible Navier-Stokes equations) is used to solve this linear system, after it is set up as described below. This linear solver is based on a Generalized Minimum Residual (GMRES, see Shakib [61]) type solution method for the velocity and a conjugate gradient projection method for the pressure.

The matrices appearing in (3.41) are the tangent matrices of the residual vectors with respect to the acceleration and pressure at time t_{n+1}, and are defined as follows:

$$\boldsymbol{K}^{(i)} \approx \frac{\partial \boldsymbol{R}_m^{(i)}(\boldsymbol{u}_{n+\alpha_f}^{(i)}, \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}, p_{n+1}^{(i)})}{\partial \dot{\boldsymbol{u}}_{n+1}^{(i)}} \tag{3.45}$$

$$\boldsymbol{G}^{(i)} \approx \frac{\partial \boldsymbol{R}_m^{(i)}(\boldsymbol{u}_{n+\alpha_f}^{(i)}, \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}, p_{n+1}^{(i)})}{\partial p_{n+1}^{(i)}} \tag{3.46}$$

$$\boldsymbol{D}^{(i)} \approx \frac{\partial \boldsymbol{R}_c^{(i)}(\boldsymbol{u}_{n+\alpha_f}^{(i)}, \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}, p_{n+1}^{(i)})}{\partial \dot{\boldsymbol{u}}_{n+1}^{(i)}} \tag{3.47}$$

$$\boldsymbol{C}^{(i)} \approx \frac{\partial \boldsymbol{R}_c^{(i)}(\boldsymbol{u}_{n+\alpha_f}^{(i)}, \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}, p_{n+1}^{(i)})}{\partial p_{n+1}^{(i)}} \tag{3.48}$$

The approximation symbols are used here to indicate that these matrices are only approximations to the consistent tangent matrices (given on the right-hand-sides of Equations (3.45)-(3.48)) which have been shown to yield better convergence, and are given in detail below. Care should be taken in computing (3.45) through (3.48), e.g. the differentiation in equation (3.45) gives rise to a mass term since $\dot{\boldsymbol{u}}_{n+\alpha_m}$ is related to $\boldsymbol{u}_{n+\alpha_f}$ through

```
Given solution at time $t_n$: $\boldsymbol{u}_n$, $\dot{\boldsymbol{u}}_n$, and $p_n$

predict:
    $\boldsymbol{u}_{n+1}^{(0)} = \boldsymbol{u}_n$
    $\dot{\boldsymbol{u}}_{n+1}^{(0)} = \frac{\gamma - 1}{\gamma}\dot{\boldsymbol{u}}_n$
    $p_{n+1}^{(0)} = p_n$

correct:
    for $i = 1$ to $i_{max}$
        (compute intermediate solution values)
        $\boldsymbol{u}_{n+\alpha_f}^{(i)} = \boldsymbol{u}_n + \alpha_f(\boldsymbol{u}_{n+1}^{(i-1)} - \boldsymbol{u}_n)$
        $\dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)} = \dot{\boldsymbol{u}}_n + \alpha_m(\dot{\boldsymbol{u}}_{n+1}^{(i-1)} - \dot{\boldsymbol{u}}_n)$

        (solve linear system)
        $\begin{pmatrix} \boldsymbol{K}^{(i)} & \boldsymbol{G}^{(i)} \\ \boldsymbol{D}^{(i)} & \boldsymbol{C}^{(i)} \end{pmatrix} \begin{pmatrix} \Delta\dot{\boldsymbol{u}}_{n+1}^{(i)} \\ \Delta\boldsymbol{p}_{n+1}^{(i)} \end{pmatrix} = -\begin{pmatrix} \boldsymbol{R}_m^{(i)} \\ \boldsymbol{R}_c^{(i)} \end{pmatrix}$

        (update solution values)
        $\dot{\boldsymbol{u}}_{n+1}^{(i+1)} = \dot{\boldsymbol{u}}_{n+1}^{(i)} + \Delta\dot{\boldsymbol{u}}_{n+1}^{(i)}$
        $\boldsymbol{u}_{n+1}^{(i+1)} = \boldsymbol{u}_{n+1}^{(i)} + \gamma\Delta_t\Delta\dot{\boldsymbol{u}}_{n+1}^{(i)}$
        $p_{n+1}^{(i+1)} = p_{n+1}^{(i)} + \Delta p^{(i)}$

    end
```

Algorithm 3.3.1: Predictor-multi-corrector algorithm

equation (3.33). Since we are differentiating with respect to $\dot{\boldsymbol{u}}_{n+1}^{(i)}$, we also need to use the chain rule, i.e.

$$\begin{aligned}\frac{\partial \boldsymbol{R}_m^{(i)}}{\partial \dot{\boldsymbol{u}}_{n+1}^{(i)}} &= \frac{\partial \boldsymbol{R}_m^{(i)}}{\partial \boldsymbol{u}_{n+\alpha_f}^{(i)}}\frac{\partial \boldsymbol{u}_{n+\alpha_f}^{(i)}}{\partial \dot{\boldsymbol{u}}_{n+1}^{(i)}} + \frac{\partial \boldsymbol{R}_m^{(i)}}{\partial \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}}\frac{\partial \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}}{\partial \dot{\boldsymbol{u}}_{n+1}^{(i)}} \\ &= \alpha_f \gamma \Delta_t \, \frac{\partial \boldsymbol{R}_m^{(i)}}{\partial \boldsymbol{u}_{n+\alpha_f}^{(i)}} + \alpha_m \, \frac{\partial \boldsymbol{R}_m^{(i)}}{\partial \dot{\boldsymbol{u}}_{n+\alpha_m}^{(i)}}\end{aligned} \tag{3.49}$$

which enables us to directly update our solution to t_{n+1} after the linear solve.

Following the standard finite element assembly process described in Hughes [42], the matrices are formed by evaluating element level integrals (using numerical quadrature). The matrices are given by

$$\begin{aligned}K_{ij}^{ab} = \int_{\bar{\Omega}_e} \{&\alpha_m N_i^a N_j^b + \alpha_f \gamma \Delta_t [\bar{u}_k N_i^a N_{j,k}^b \\ &+ N_{i,k}^a (\mu N_{j,k}^b + \tau_M u_k u_m N_{j,m}^b + \bar{\tau} \mathcal{L}_k \mathcal{L}_m N_{j,m}^b) \\ &+ \mu N_{(i),(j)}^a N_{(j),(i)}^b + \tau_C N_{(i),(i)}^a N_{(j),(j)}^b]\} \, dx\end{aligned} \tag{3.50}$$

$$G_p^{ab} = -\int_{\bar{\Omega}_e} N_{i,i}^a N_p^b \tag{3.51}$$

$$D_i^{ab} = \int_{\bar{\Omega}_e} N_{p,i}^a N_p^b \tag{3.52}$$

$$C^{ab} = -\tau_M \int_{\bar{\Omega}_e} N_{p,i}^a N_{p,i}^b \tag{3.53}$$

where, here, $a, b = 1 \ldots n_{es}$ refer to the individual basis function contributions, the subscripts $i, j = 1 \ldots 3$ are included to indicate the basis functions related to the momentum equations (velocity degrees of freedom) and the subscript p indicates continuity equation (pressure degrees of freedom). Here, indices enclosed in parentheses imply that no sum should be carried out. Since, as mentioned above, we are interpolating velocity and pressure with the same basis functions, the subscripts i and j are only used to indicate the place of these terms in the resulting matrices. The terms we have chosen to include in the tangent matrices given above are essentially formed from the frozen coefficient assumption while differentiating the equations. Assumptions on these matrices also enable the relationship, $\boldsymbol{D} = -\boldsymbol{G}^T$ which is a desirable symmetry property between the discrete divergence and gradient operators ($\boldsymbol{G}$ and $\boldsymbol{D}$, respectively). Gresho and Sani [29] provide additional details pertaining to this symmetry.

3.3.3 Temporal accuracy: flow past a circular cylinder

Vortex shedding around a circular cylinder in an incompressible flow at a Reynolds number of 100 (based on the cylinder diameter and inflow velocity) will provide an application of the generalized-α method to the relatively well understood flow. Since we are primarily interested in studying the temporal accuracy, we have chosen to present results from the linear basis calculations; however the results for other polynomial order simulations are similar. The problem geometry and boundary conditions are depicted in Figure 3.1. In addition to the boundary conditions shown, we have imposed zero x_3-

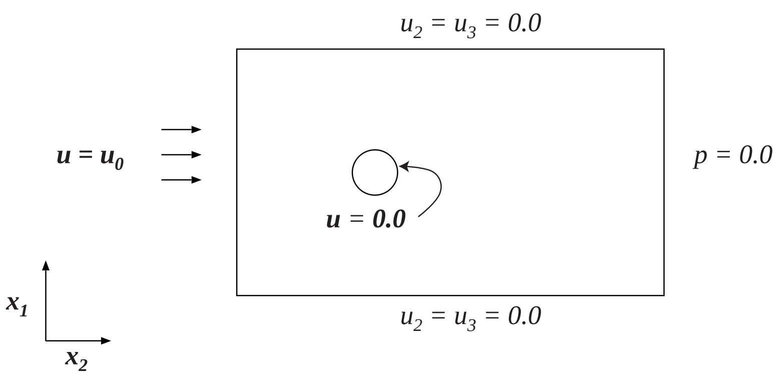


Figure 3.1: Cylinder geometry and boundary conditions

velocity and no tangential traction on the two x_3 planes to simulate 2-D conditions with our 3-D code. A complete description of this problem can be found in a variety of references: Shakib *et al.*[63], Hauke and Hughes [34], and Behr *et al.* [6]. At a Reynolds number of 100, the salient feature of this flow is the periodic shedding of vortices from the cylinder creating time varying lift and drag forces determined by integrating the forces on the cylinder surface. This single, dominant frequency allows a convenient study of the new time integrator.

In this context, we will study the effect of the high-frequency damping parameter,

ρ_∞, on the lift and drag profiles. The flow is solved in time assuming the cylinder is initially at rest, and is immediately accelerated to a velocity of $\boldsymbol{u}_0$ (often referred to as an impulsive start). We have taken a time step of 0.1 (normalized by the cylinder diameter) and for each time step, three Newton corrector passes are performed, insuring that the normalized change in the velocity increment is less than 5×10^{-4} for each step. This time step affords 60 steps per period of the lift force and 30 steps per period of the drag force, which should be sufficient to completely resolve this frequency.

The first four plots in Figure 3.2 show the lift and drag forces (f_l and f_d, respectively) plotted against the non-dimensional time, t^*, for $\rho_\infty = 0.0,\ 0.25,\ 0.5$, and 0.75 for two different time windows (the one on the right being a zoomed view). From these plots we make the following observations: i) the period and amplitude of both the lift and the drag are very weak functions of ρ_∞ (which might be expected from the observation that 30 points per wave length is adequate for second-order accurate methods), ii) a small amplitude undulation is present in the $\rho_\infty = 0.75$ case, iii) the presence of this undulation for $\rho_\infty > 0.5$ reflects the increasing difficulty the method faces as ρ_∞ approaches 1, iv) the $\rho_\infty = 0$ case appears to start its transition from a steady separation to a saturated unsteady flow much earlier than the other methods.

The last observation is somewhat counter-intuitive. One might expect the method with the highest damping would be the last to leave a steady flow in favor of an unsteady flow. The final two plots in Figure 3.2 shed some light on this mystery. They indicate that the impulsive start can introduce a rather large, high-frequency unsteadiness to the flow. Furthermore, these plots indicate the severe errors that may occur when using $\rho_\infty = 1.0$ (trapezoidal rule) for this flow. The first of these two plots shows the time window including the impulsive start. This plot clearly shows the strong damping characteristics of the $\rho_\infty = 0$ case, where the initial disturbance due to the impulsive start is damped out within a couple of time steps. Also clear from this plot is that the initial disturbance is almost completely preserved in the $\rho_\infty = 1$ case, polluting the entire solution. Intermediate values of ρ_∞, annihilate this un-resolvable frequency in a manor predictable by their proximity to the two extreme cases. It is conjectured that the physical instability leading to limit cycle vortex shedding is in the well resolved and almost completely undamped range for all chosen values of ρ_∞ ($0 \leq \rho_\infty \leq 1$). However, by taking less time to anni-

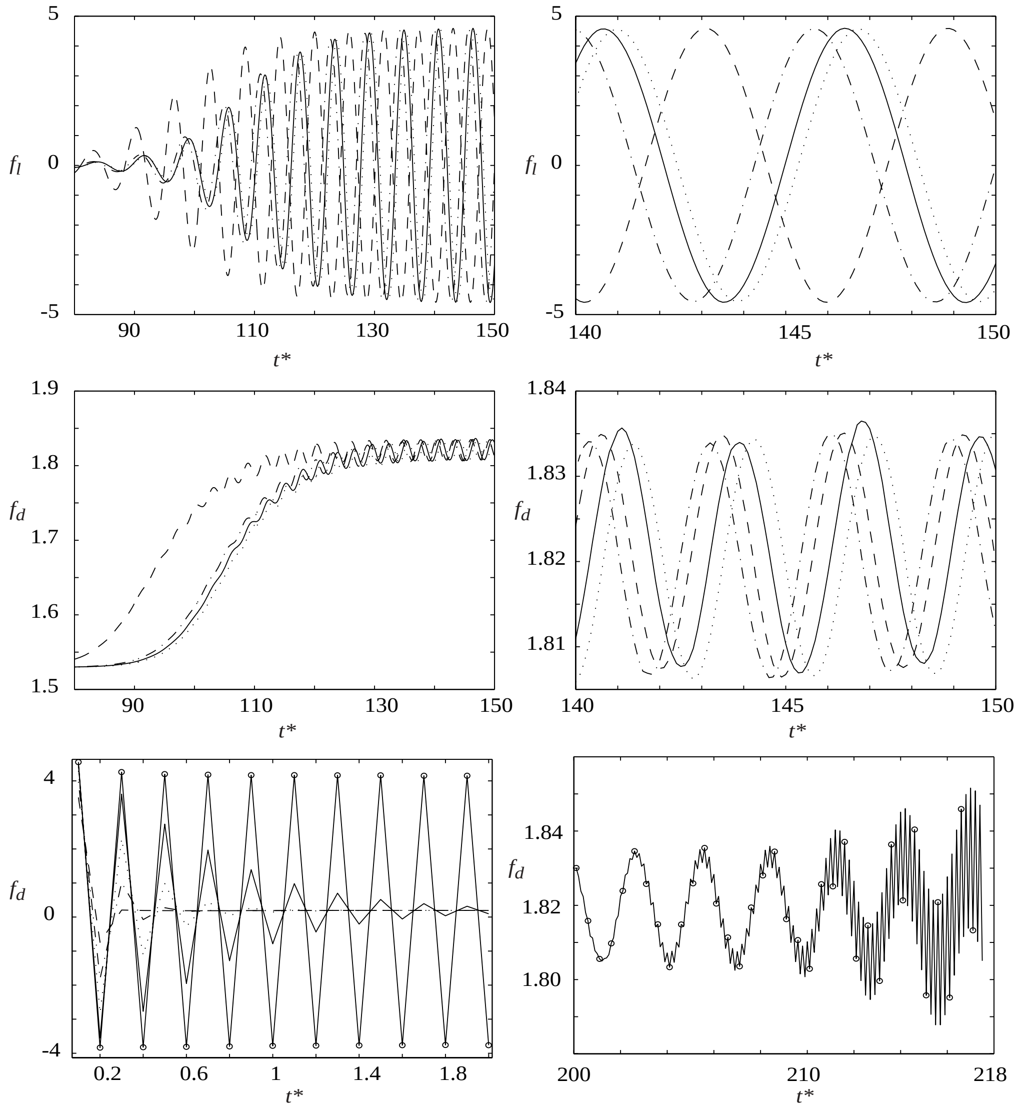


Figure 3.2: Lift, f_l, and drag, f_d, forces on the cylinder for different time windows: ---- $\rho_\infty = 0.00$, —·— $\rho_\infty = 0.25$, ······· $\rho_\infty = 0.50$, —— $\rho_\infty = 0.75$, and –○– $\rho_\infty = 1.0$

hilate the highest, un-resolvable frequency, the lower values of ρ_∞ actually pick up the physical instability (which starts at extremely low values and grows exponentially before saturating on the limit cycle) sooner. The higher values of ρ_∞ are unable to pick up this very low amplitude physical instability until they bring the un-resolvable frequency below its level. This should not be misconstrued as an advocation of $\rho_\infty = 0.0$ in all cases but it does demonstrate the benefit in at least one case of having the ability to annihilate a large, un-resolvable frequency rapidly. In simulations of practical interest, this ability must be balanced by the fact that there often exist a continuous range of frequencies that one is interested in resolving, rather than two, widely separated ones as shown here. In those cases, higher values of ρ_∞ are desirable to maintain the ability to accurately integrate waves with significantly less then 30-60 time steps per period, which is often the case for turbulence simulations.

The final plot in Figure 3.2 shows that the highest frequency can be excited even without an impulsive start. In this case the flow was restarted from the $\rho_\infty = 0.0$ solution (at $t^* = 200$ where no high frequencies were visibly present). However, the nonlinearities inherent to the Navier-Stokes equations needed little time to build up energy in this highest frequency when the time integrator was switched to $\rho_\infty = 1.0$, a time integrator that is powerless to control these frequencies. This highest frequency mode does saturate in amplitude, though, leaving a surprisingly accurate signal that can be recovered by filtering this signal in time. This is again fortuitous to this case though, owing to the wide separation of the frequencies causing little, if any, interaction. Again it must be stressed that in the problems of interest, the continuous range of scales will suffer much greater contamination due to the stronger interaction of waves of close proximity in frequency space. In these cases the energy in the highest frequencies may also fail to saturate leading to a breakdown of the solution technique. Clearly, some capacity to insure the annihilation of un-resolvable waves is critical to maintain the fidelity of the well and marginally resolved waves, thus the motivation for the design of the method with careful control through ρ_∞ and the advocacy for the availability and use of intermediate values.

3.4 Application of the hierarchical basis to compressible flow

This section presents an application of the hierarchical basis described in the previous chapter to the compressible system of Navier-Stokes equations. The stabilized finite element formulation used here, which is the standard SUPG formulation (equivalent to the Galerkin-Least Squares (GLS) formulation for linear basis functions, see Hauke and Hughes [33]), differs from that introduced above for the incompressible equations, but retains the desirable properties of stability and higher-order accuracy. In fact, the formulation presented here for the compressible system of equations is applied to the conservative form of the equations, and the additional terms terms introduced to restore conservation to the incompressible formulation need not be added. This formulation has also been more rigorously studied and analytically proven to be stable and higher-order accurate for model problems (see Shakib and Hughes [62]). More details pertaining to the formulation for the compressible equations may be found in Whiting *et al.* [76]. The compressible formulation also differs in the way pressure is treated. Here the pressure is solved as a thermodynamic quantity, (rather than a constraint equation), which is defined through thermodynamic considerations and the energy equation (we assume an ideal gas relationship, although this is not a necessary assumption). The pressure is therefore not treated distinctly from the velocity as it is in the incompressible formulation, but rather, all flow quantities are considered with a single weight space containing 5 variables (including total energy).

3.4.1 Compressible Navier-Stokes equations

Consider the application of the mesh entity based hierarchical basis functions to the time-dependent, compressible Navier-Stokes equations, written in conservative form as

$$\boldsymbol{U}_{,t} + \boldsymbol{F}^{\text{adv}}_{i,i} - \boldsymbol{F}^{\text{diff}}_{i,i} = \boldsymbol{\mathcal{S}} \tag{3.54}$$

with the conservation variables given by

$$\boldsymbol{U} = \left\{ \begin{array}{c} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \right\} = \rho \left\{ \begin{array}{c} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{\text{tot}} \end{array} \right\} \tag{3.55}$$

and the advective and diffusive fluxes are defined as

$$\boldsymbol{F}_i^{\text{adv}} = u_i \boldsymbol{U} + p \left\{ \begin{array}{c} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{array} \right\}, \quad \boldsymbol{F}_i^{\text{diff}} = \left\{ \begin{array}{c} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} u_j - q_i \end{array} \right\} \tag{3.56}$$

The equations are closed through the introduction of constitutive relationships given by

$$\tau_{ij} = 2\mu(S_{ij}(\boldsymbol{u}) - \frac{1}{3} S_{kk}(\boldsymbol{u})\delta_{ij}), \qquad S_{ij}(\boldsymbol{u}) = \frac{u_{i,j} + u_{j,i}}{2} \tag{3.57}$$

$$q_i = -\kappa T_{,i}, \qquad e_{\text{tot}} = e + \frac{u_i u_i}{2}, \qquad e = c_v T \tag{3.58}$$

The variables are: velocity u_i, pressure p, density ρ, temperature T and total energy e_{tot}. The constitutive laws relate the stress, τ_{ij}, to the deviatoric portion of the strain, $S_{ij}^d = S_{ij} - \frac{1}{3} S_{kk}\delta_{ij}$, through a molecular viscosity, μ. Similarly, the heat flux, q_i, is proportional to the gradient of temperature with the proportionality constant given by a molecular conductivity, κ. While the formulation is not limited to an ideal gas, $p = \rho R T$, and constant specific heats at constant pressure, c_p, and at constant volume, c_v, these assumptions are also made here. Furthermore, since we are generally interested in low Mach number flows where temperature variation is low, we have assumed a constant molecular viscosity and constant conductivity through a constant Prandtl number, although this again is not a necessary simplification. Finally, $\boldsymbol{S}$ is a body force (or source) term.

For the specification of the stabilized formulation for the compressible system of equations, it is helpful to define the quasi-linear operator (with respect to some yet to be defined variable vector $\boldsymbol{Y}$) related to (3.54) as

$$\boldsymbol{\mathcal{L}} \equiv \boldsymbol{A}_0 \frac{\partial}{\partial t} + \boldsymbol{A}_i \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i}(\boldsymbol{K}_{ij} \frac{\partial}{\partial x_j}) \tag{3.59}$$

from which $\boldsymbol{\mathcal{L}}$ can be naturally decomposed into time, advective, and diffusive portions

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}}_{\text{t}} + \boldsymbol{\mathcal{L}}_{\text{adv}} + \boldsymbol{\mathcal{L}}_{\text{diff}}. \tag{3.60}$$

Here $\boldsymbol{A}_i = \boldsymbol{F}^{\text{adv}}_{i,\boldsymbol{Y}}$ is the i^{th} Euler Jacobian matrix, $\boldsymbol{K}_{ij}$ is the diffusivity matrix, defined such that $\boldsymbol{K}_{ij}\boldsymbol{Y}_{,j} = \boldsymbol{F}^{\text{diff}}_i$, and $\boldsymbol{A}_0 = \boldsymbol{U}_{,\boldsymbol{Y}}$ is the change of variables metric. For a complete description of $\boldsymbol{A}_0$, $\boldsymbol{A}_i$ and $\boldsymbol{K}_{ij}$, the reader is referred to Hauke [32]. Using this, we can write (3.54) as simply $\boldsymbol{\mathcal{L}Y} = \boldsymbol{\mathcal{S}}$.

The finite element spaces used for the compressible formulation are similar to those introduced above for the incompressible equations with the exception that the solution vector is now treated as a single vector of unknowns (pressure is not given its own space). The solution space is now given by

$$\boldsymbol{\mathcal{S}}_h^k = \{\boldsymbol{v}|\boldsymbol{v}(\cdot,t) \in H^1(\Omega)^m, t \in [0,T], \boldsymbol{v}|_{x\in\bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^m, \boldsymbol{v}(\cdot,t) = \hat{\boldsymbol{g}} \text{ on } \Gamma_g\}, \tag{3.61}$$

and the weight function space

$$\boldsymbol{\mathcal{W}}_h^k = \{\boldsymbol{w}|\boldsymbol{w}(\cdot,t) \in H^1(\Omega)^m, t \in [0,T], \boldsymbol{w}|_{x\in\bar{\Omega}_e} \in P_k(\bar{\Omega}_e)^m, \boldsymbol{w}(\cdot,t) = \boldsymbol{0} \text{ on } \Gamma_g\}, \tag{3.62}$$

where $P_k(\bar{\Omega}_e)^m$ is as defined before, and $m = 5$ representing our 5 unknown variables (u_i, p ,and T).

To derive the weak form of (3.54), we proceed the same as for the incompressible system by dotting the entire system from the left by a vector of weight functions, $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_h^k$, and integrating over the spatial domain. Integration by parts is then performed on the diffusive *and* advective terms to move the spatial derivatives onto the weight functions (reducing the continuity requirements). Recall that for the incompressible formulation, the advective term was not integrated by parts. This process leads to the semi-discrete SUPG weak form for the compressible equations (see Hauke and Hughes [33]): find $\boldsymbol{Y} \in \boldsymbol{\mathcal{S}}_h^k$ such that

$$\begin{aligned}
&\int_{\Omega} \left(\boldsymbol{W} \cdot \boldsymbol{U}_{,t} - \boldsymbol{W}_{,i} \cdot \boldsymbol{F}_i^{\text{adv}} + \boldsymbol{W}_{,i} \cdot \boldsymbol{F}_i^{\text{diff}} \right) d\Omega \\
&\quad - \int_{\Gamma} \boldsymbol{W} \cdot \left(-\boldsymbol{F}_i^{\text{adv}} + \boldsymbol{F}_i^{\text{diff}} \right) \, n_i \, d\Gamma \\
&+ \sum_{e=1}^{n_{el}} \int_{\bar{\Omega}_e} \boldsymbol{\mathcal{L}}_{\text{adv}}^T \boldsymbol{W} \cdot \boldsymbol{\tau} \left(\boldsymbol{\mathcal{L}} \boldsymbol{Y} - \boldsymbol{\mathcal{S}} \right) d\Omega = 0
\end{aligned} \tag{3.63}$$

for all $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_h^k$. The first and second lines of (3.4.1) contain the Galerkin approximation (interior and boundary) and the third line contains the SUPG stabilization. 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. The use of the advective portion of the operator, $\boldsymbol{\mathcal{L}}_{\text{adv}}^T$, in the stabilization term could be replaced by $\boldsymbol{\mathcal{L}}^T$ to yield the full GLS method. Doing so, however, would require that we reconstruct the second derivative of the weight space (using the local reconstruction method for the diffusive flux), which is a costly operation. It is also not clear that the GLS formulation is more accurate than SUPG (theoretically, they have both been shown to be optimally accurate for model problems).

The stabilization parameter, $\boldsymbol{\tau}$, is now a 5×5 matrix, and its extension to time-dependent systems of equations is well documented in Shakib [61], Franca and Frey [22], and Hughes and Mallet [40]. Based on efficiency considerations, we have chosen to use a diagonal $\boldsymbol{\tau}$ similar to the one introduced by Hauke [32] for nearly incompressible flows, given by

$$\boldsymbol{\tau} = \text{diag}(\tau_c, \tau_m, \tau_m, \tau_m, \tau_e) \tag{3.64}$$

where

$$\tau_c = \frac{|\boldsymbol{u}| h_1^e}{2} \text{min}(1, Re^h) \tag{3.65}$$

$$\tau_m = \min\left(\frac{\Delta_t}{\rho}, \frac{h_2^e}{2\rho|\boldsymbol{u}|}, \frac{m_k(h_3^e)^2}{4\mu}\right) \tag{3.66}$$

$$\tau_e = \min\left(\frac{\Delta_t}{\rho c_v}, \frac{h_3^e}{2\rho c_v|\boldsymbol{u}|}, \frac{m_k(h_3^e)^2}{4\kappa}\right) \tag{3.67}$$

and

$$Re^h = \frac{\rho|\boldsymbol{u}|h_1^e}{2\mu}, \quad m_k = \min(1/3, 2C_k). \tag{3.68}$$

Here h_i^e are element length scales for continuity, momentum, and energy and Δ_t is the time step; the other parameters have been defined above. This $\boldsymbol{\tau}$ has proven effective on a variety of flows as shown by Hauke and Hughes [33], and is computationally efficient, as it can be computed from existing solution variables. More advanced forms of $\boldsymbol{\tau}$ which are not diagonal can be formed and are documented in Hauke and Hughes [33], however, for low Mach number flows, they showed that the diagonal form given above performed well.

Note that we have chosen to solve for $\boldsymbol{Y}$ instead of $\boldsymbol{U}$. As discussed in Hauke and Hughes [34, 33], $\boldsymbol{U}$ is often not the best choice of solution variables, particularly when the flow is nearly incompressible. The computations performed herein employ pressure-primitive variables, viz.

$$\boldsymbol{Y} = \begin{Bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \end{Bmatrix} = \begin{Bmatrix} p \\ u_1 \\ u_2 \\ u_3 \\ T \end{Bmatrix} \tag{3.69}$$

By inspecting (3.56)-(3.58) it is clear that all quantities appearing in (3.4.1) may be easily calculated from (3.69). The use of pressure-primitive variables also facilitates specification of boundary conditions, as these are the variables most often constrained for nearly incompressible applications.

Another reason we have chosen to use the pressure primitive variables is that they are well behaved in the incompressible limit, i.e. the matrices in (3.59) are well behaved

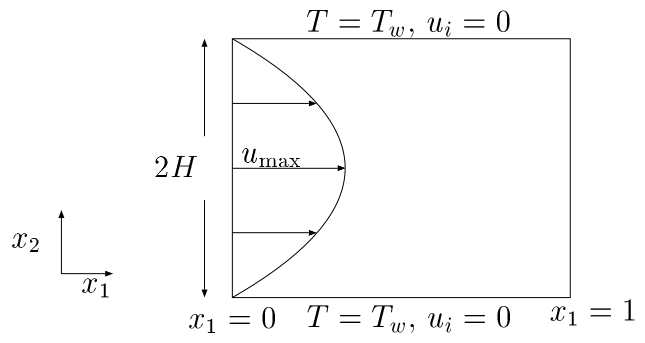


Figure 3.3: Channel flow geometry and problem description

as the compressibility coefficients α_P and β_T approach zero. This makes it a simple matter to construct a purely incompressible code from the above formulation providing a uniform approach to compressible and incompressible flows as described by Hauke and Hughes [34]: simply set $\alpha_P = \beta_T = 0$ in the coefficient matrices. Experience has shown, however, that the formulation presented in Section 3.2 based on the advective form of the equations is far more accurate as well as much more efficient for an incompressible flow, and is preferred for this type of simulation over the conservation variable formulation presented here.

The generalized-α method is again used for the temporal discretization, yielding a system of nonlinear algebraic equations which is solved in a predictor-multicorrector format yielding successive linear problems. Subsequently, each linear problem is solved using the Matrix-Free Generalized Minimal Residual (MF-GMRES) solution technique with a block diagonal preconditioner developed by Johan *et al.* [48] or element-by-element GMRES techniques. These linear algebra solvers have both proven to be effective for compressible flows.

3.4.2 Example: compressible channel flow

This example is presented as a verification of the theoretical convergence results for the SUPG formulation applied to the compressible equations. It demonstrates that under some circumstances, results derived for model problems carry over to the nonlinear systems of practical interest. Consider the fully developed, laminar, constant viscosity, compressible flow between two flat plates, shown in Figure 3.3.

Assuming the viscosity to be independent of temperature de-couples the energy equation allowing a closed-form, analytical solution as follows:

$$u_1 = u_{\max}\left[1-\left(\frac{x_2}{H}\right)^2\right], \qquad u_2 = u_3 = 0 \tag{3.70}$$

$$T = T_w\left[1+\frac{Pr\,\hat{E}c}{3}\left\{1-\left(\frac{x_2}{H}\right)^4\right\}\right], \qquad p = p_o - \frac{2\rho_o u_{\max}^2}{Re_H}\frac{x_1}{H} \tag{3.71}$$

where H is half the height of the channel, p_o is the pressure at $x_1 = 0$, ρ_o is the density at $x_1 = 0$, $u_{\max}$ is the centerline velocity, T_w is the prescribed wall temperature, $Pr = \frac{\mu}{c_p\kappa}$ is the Prandtl number, $Re_H = \frac{u_{\max}H}{\nu}$ is the Reynolds number based on the channel half-height, and $\hat{E}c = \frac{u_{\max}^2}{c_pT_w}$ is a modified Eckert number using the specific heat c_p. This solution holds for compressible or incompressible flow so long as the viscosity is assumed constant and provides a good test case for a convergence study since the temperature field is fourth order, enabling us to verify the convergence rate through $k = 3$.

We have chosen to simulate this flow assuming periodic boundary conditions in the streamwise direction (implying fully-developed flow), in addition to $T = T_w$ and $\boldsymbol{u} = 0$ at the upper and lower walls. The periodic boundary condition prescription is obtained by forcing all variables on the outflow plane to be identical with those on the inflow plane. To do this requires that the linear portion of the pressure be interpreted as a body force, i.e.,

$$p = p_o + B\frac{x_1}{H}. \tag{3.72}$$

Then the constant portion, p_o can be assumed periodic along with the other variables, u_i and T, which are independent of x_1. The linear portion ($B = -\frac{2\rho_o u_{\max}^2}{Re_H}$ is a constant coefficient) of the pressure is included in the formulation as a source term, which “drives” the flow.

This problem is solved on three different meshes for $k = 1\ldots 3$. The L^2 error in the temperature finite element solution (normalized by the L^2 norm of the analytical temperature profile) $vs.$ the number of elements in the x_2 direction and $vs.$ the total number of degrees of freedom are shown in Figure 3.4 . The first of the two plots shows that through

$k = 3$ the finite element solution is converging at a slope of $k + 1$, as predicted by theory. The second plot demonstrates the benefit of the higher-order solutions compared to the linear solution for a fixed number of degrees of freedom. Note that due to the stabilization term, even the coarse meshes benefit from the increased polynomial order. Of course, the number of degrees of freedom in the system is only an indication of the total cost of solution, since degrees of freedom associated with k-refinement are typically more expensive than those associated with h-refinement. More careful studies of the simulation cost for higher-order elements will be given later for incompressible flows.

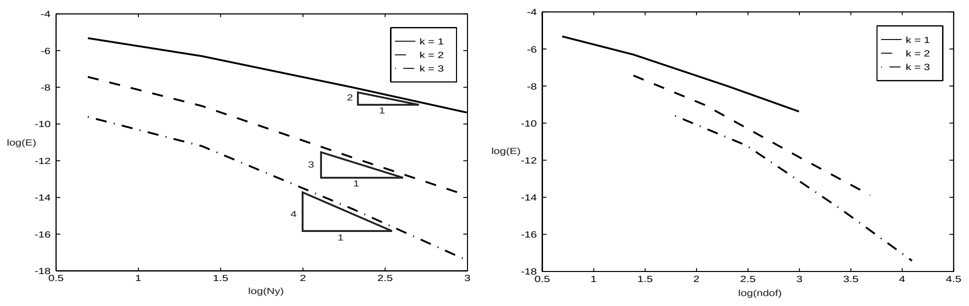


Figure 3.4: Convergence of temperature profiles for channel flow

3.4.3 Example: vortex shedding behind a circular cylinder

The second example for the compressible formulation, is of the time dependent flow about a circular cylinder at a Reynolds number of 100 (based on the cylinder diameter), the geometry and boundary conditions are shown in Figure 3.5. A complete description

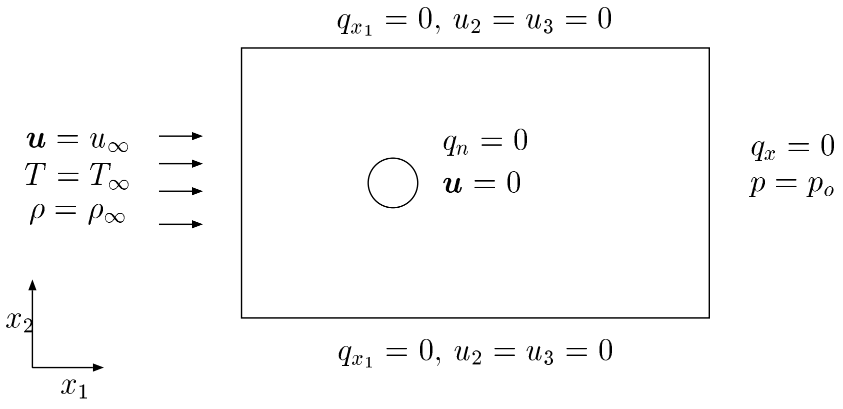


Figure 3.5: Cylinder geometry and boundary conditions

of this compressible flow can be found in variety of references: Shakib *et al.*[63], Hauke and Hughes [34], Behr *et al.* [6]. We choose boundary conditions consistent with those of Shakib *et al.*[63]. At a Reynolds number of 100, the features of this flow are similar to the incompressible flow introduced above: the periodic shedding of vortices from the cylinder creating a time varying lift determined by integrating the forces on the cylinder surface. Since it is now a compressible flow, we have set the Mach number at infinity, $M_\infty = 0.2$, making the flow nearly incompressible. The physical dimensions leading to such flow parameters may be questionable, a quick calculation yields a cylinder diameter of $D = 1.5 \times 10^{-5}$ meters, quite small, however the simulations yield results that are consistent with more physically realizable dimensions.

This simulation is used to show the increased accuracy in the prediction of the forces on the cylinder gained from quadratic as opposed to linear basis functions. Particularly the drag forces, which are dominated by the viscous forces on the cylinder, are much better resolved by the quadratic solution, as can be seen in Figure 3.6. The drag from the linear basis is observed to have a low frequency amplitude modulation which has been eliminated in the quadratic solution. The force in the x_2 direction, or lift, is less sensitive to an increase in the basis order due to the fact that it is largely dominated by the pressure forces which can be interpolated well by the linear basis. A careful comparison of the cost *vs.* accuracy for the compressible equations was not carried out. In fact, this flow was simulated on the same mesh for the linear and quadratic simulations, hence, the quadratic solution has many more degrees of freedom associated with it. It is only presented as an initial application to a compressible flow. Careful comparisons of cost and accuracy will be studied in the context of incompressible flow.

3.5 Diffusive flux computation

We would like to conclude the chapter with details of the computation of the diffusive flux terms appearing in both stabilized finite element formulations presented above. Both the incompressible and compressible systems of equations require this term, however it takes on slightly different forms in each case, and we will concentrate on the incompressible version. Careful inspection of the weak form, (3.8), and in particular the momentum residual equation, (3.9), reveals that it is necessary to calculate the second derivative of the solution variable when evaluating the residual of the diffusive flux stabi-

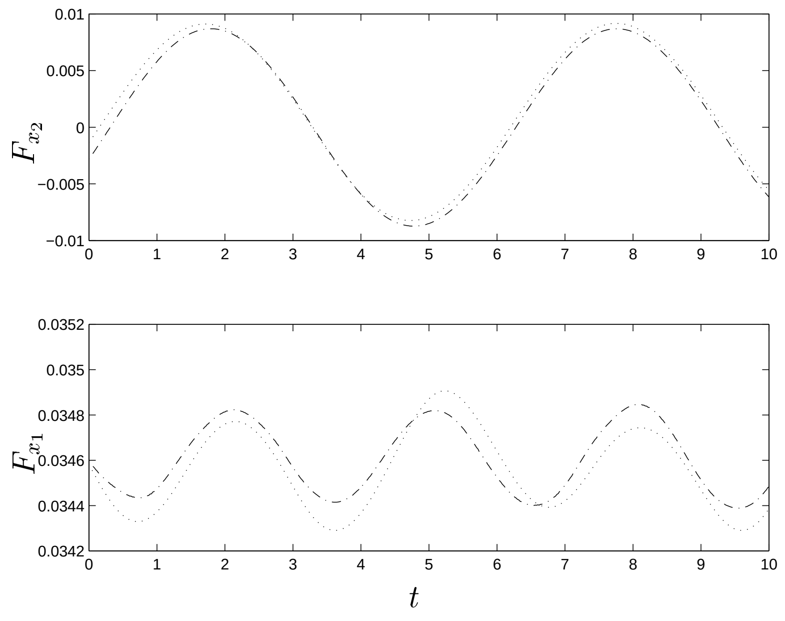


Figure 3.6: Forces on cylinder surface:$\cdots k = 1$,$- \cdot - k = 2$

lization terms (for the incompressible equations)

$$q_i \equiv \tau_{ij,j} = \nu(u_{i,j} + u_{j,i})_{,j} \tag{3.73}$$

While these terms are often neglected for linear basis calculations (with some justification), their inclusion is vital to the accuracy of higher-order simulations (examples run without these terms show a significant degradation of solution quality). It is possible to evaluate these terms directly from the second derivatives of the basis functions, however this involves the evaluation of the second derivative of the mapping if curved elements are used, which is a costly operation. We opt instead for a more efficient method using a local reconstruction of the diffusive flux terms based on an L^2 projection followed by a re-interpolation. A procedure has also been developed for creating a global reconstruction of the diffusive flux which can be used for linear elements to provide more accurate simulations at a negligible additional cost.

3.5.1 Local, element-level reconstruction

The local reconstruction technique provides a relatively straightforward method to compute an approximation to the diffusive flux involving only element level data. This technique is more cost effective than directly evaluating the second derivatives of the basis functions which involves the second derivative of the geometric mapping for non-straight-sided elements, since the inverse of the element-level projection matrix needs to be computed only one time and stored, and may be used for all subsequent evaluations.

The general idea is to project the viscous stress field, τ_{ij}, (which may be computed with the first derivatives of the basis) onto the element basis, then re-interpolate it with the first derivative of the basis to form the diffusive flux field, i.e. $q_i \equiv \tau_{ij,j}$. The projection is constructed such that the L^2 error is minimized over each element independently, i.e., find $\hat{\tau}_{ij} \in \boldsymbol{\mathcal{S}}_h^k$ such that

$$\int_{\bar{\Omega}_e} w\left(\hat{\tau}_{ij} - \tau_{ij}^{(h,k)}\right)\, dx = 0 \tag{3.74}$$

for all $w \in \boldsymbol{\mathcal{W}}_h^k$, where $\tau_{ij}^{(h,k)}$ represents the current finite element approximation of the stress field. It should be noted that each of the components in $\tau_{ij}^{(h,k)}$ is projected independently. Expanding the weight function in terms of the basis functions yields a system of linear equations to be solved for the basis coefficients of $\hat{\tau}_{ij}$, of the form

$$\boldsymbol{M}\hat{\boldsymbol{\tau}}_{ij} = \boldsymbol{R}_{ij} \tag{3.75}$$

where,

$$\boldsymbol{M} = [\boldsymbol{M}_{ab}] = \int_{\bar{\Omega}_e} N_a N_b\, dx, \quad \boldsymbol{R} = \{\boldsymbol{R}_a\} = \int_{\bar{\Omega}_e} N_a \tau_{ij}^{(h,k)}\, dx \tag{3.76}$$

This system is solved for the stress projection coefficients, $\hat{\boldsymbol{\tau}}_{ij} = \{\hat{\tau}_{ij}^a\}$, for each element, which are then re-interpolated with the gradients of the basis functions to form an approximation to q_i as

$$q_i = \sum_{a=1}^{n_{es}} N_{a,j}\hat{\tau}_{ij}^a. \tag{3.77}$$

The system of equations, 3.75, is inverted one time and stored, therefore the evaluation of the projection coefficients involves a single integral evaluation, which is computed using the same Gauss quadrature rule as the integration of the residual.

The inclusion of this term is vital to the performance of higher-order methods, since without it, the formulation no longer maintains its weighted residual character, i.e., consistency is violated. The effect of not including this term is illustrated by means of the simple example of incompressible Poiseuille (channel) flow between two infinite, parallel plates. The exact solution is quadratic in velocity and linear in pressure, i.e. within the finite element approximation space for quadratic elements. The applied boundary conditions consist of setting the exact velocity profile at the inlet, $x_1 = 0$, and setting the pressure at the outflow, $x_1 = 1.0$, in addition to the no-slip condition at the upper and lower walls. Figure 3.7 shows a comparison of the velocity and pressure profiles plotted against the x_1 coordinate along the centerline of the channel. Since the velocity has no x_1 dependence, it should remain equal to the centerline inlet velocity of 1.5 and the exact pressure is linear in x_1. This figure clearly shows the inability of the incomplete residual

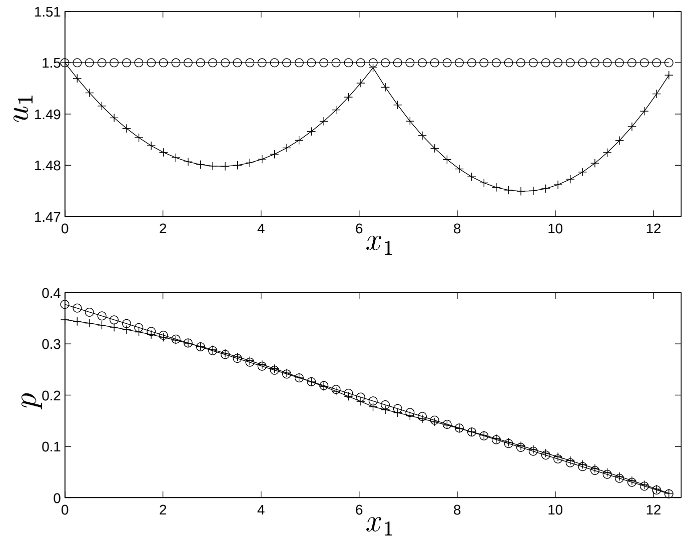


Figure 3.7: Centerline velocity and pressure with, ○ , and without, + , the reconstructed diffusive flux.

method to obtain the exact solution, even though it is within the finite element space, and large errors are incurred in both the pressure and velocity if the diffusive flux terms are neglected. The situation illustrated here for this simple example is expected to be even more severe for complicated flows.

3.5.2 Global reconstruction (linear elements)

The use of the second derivatives in the stabilization residual terms has been shown to yield much more accurate simulations for the higher-order methods (it is in fact necessary to maintain a weighted residual formulation). These terms are often neglected for linear basis computations since they rely on second derivatives of the solution, which are zero when using the linear basis, however it will be shown that including them results in increased accuracy at a negligible additional cost (more details as well as result for other flows may be found in Jansen *et al.* [45]).

The local reconstruction will clearly not work for linear elements since there is no way to approximate a higher-order quantity with a single linear element. To circumvent this problem, a global reconstruction algorithm has been developed following the same logic as in Section 3.5.1 for the local reconstruction. Here $Q_i \equiv \tau_{ij,j}$ is the diffusive flux in the i^{th} direction which must be reconstructed. The reconstructed diffusive flux is then given in terms of the the element shape functions as

$$Q_i = \sum_{A=1}^{n_v} N_{A,j} \hat{\tau}_{ij}^A \tag{3.78}$$

where $\hat{\tau}_{ij}^A$ is the diffusive flux in the j^{th} direction at global node A (the capital Q_i is to distinguish it from our locally reconstructed diffusive flux, q_i). This quantity is not defined in the usual finite element sense due to the discontinuous nature of the derivatives of low-order piece-wise polynomials. It can, however, be reconstructed to be a continuous variable using a global L^2 projection operator. The procedure is described as follows:

$$\boldsymbol{M}\hat{\boldsymbol{Q}}_j = \boldsymbol{R}_j \tag{3.79}$$

$$\boldsymbol{M} = [\boldsymbol{M}_{AB}]; \qquad \boldsymbol{R}_j = \{\boldsymbol{R}_{Aj}\}; \qquad \hat{\boldsymbol{Q}}_j = \{\boldsymbol{Q}_{Bj}\} \tag{3.80}$$

$$\boldsymbol{M}_{AB} = \boldsymbol{I}\int_\Omega N_A N_B d\Omega; \qquad \boldsymbol{R}_{Aj} = \int_\Omega N_A \boldsymbol{Q}_j(\boldsymbol{U}(\boldsymbol{x}))d\Omega \tag{3.81}$$

where these matrices are computed in the usual finite element manner, assembling ele-

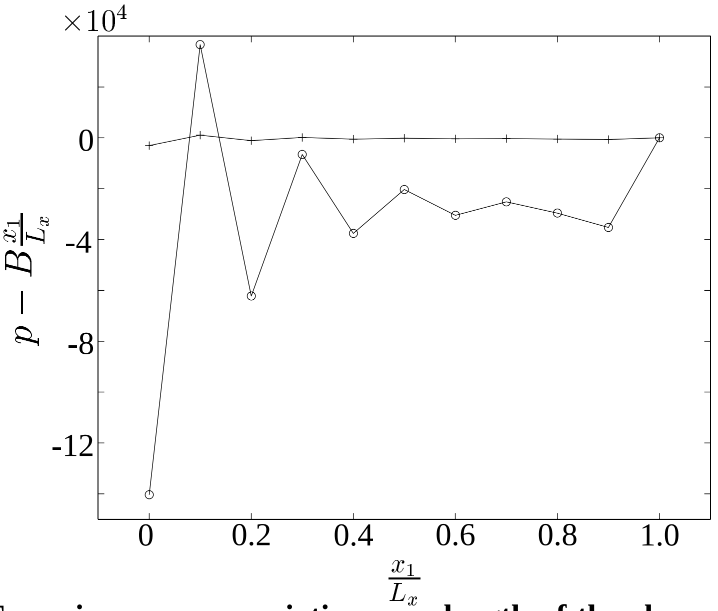


Figure 3.8: Error in pressure variation over length of the channel. With residual completion + , without residual completion ○ .

ment level contributions to form the global counterparts. Since the solution procedures for the Navier-Stokes equations involve successive iterations of solving a linearization of the nonlinear problem, it is quite easy to lag the solution for $\boldsymbol{Q}_{jA}$ by one iteration. Furthermore, it is usually sufficient to replace $\boldsymbol{M}_{AB}$ by its lumped mass equivalent. We prefer to use the special lumping described in Hughes [42].

The compressible channel flow considered above for the local reconstruction is considered here using linear elements and the global reconstruction technique. It should be stressed that for linear elements the exact solution, which is quadratic velocity and linear pressure, can not be exactly represented by the linear basis. However, use of the globally reconstructed diffusive flux yields a much more accurate result as shown in Figure 3.8.

3.6 Chapter summary

This chapter introduced the stabilized finite element formulation for the incompressible Navier-Stokes equations using mesh entity based hierarchical basis functions for the spatial discretization. A second order accurate, implicit time integrator with user-controllable numerical dissipation was introduced for integrating the equations in time, and shown to have desirable characteristics for the flow behind a circular cylinder. The implementation involves relatively few modifications to a highly efficient linear basis solver, allowing us to maintain efficiency for large-scale problems. To achieve this goal, much of the computational effort has been transferred to the pre-processing stage of the analysis, where the data structures are created and written to disk for high efficiency when used by the flow solver. One key difference between linear and higher-order basis functions is the treatment of the diffusive portion of the residual in the stabilization terms, a problem unique to stabilized methods. New methods for dealing with this term were presented for higher-order computations (local reconstruction) as well as for the linear basis (global reconstruction), and the increase in accuracy was demonstrated in both cases. The next chapter will describe many implementational details that are encountered when using hierarchical basis methods for fluid dynamics.

CHAPTER 4
HIGHER-ORDER SIMULATIONS IN A PARALLEL COMPUTING ENVIRONMENT

The effective use of hierarchical basis functions for solving the Navier-Stokes equations requires that considerations of efficiency be paramount in designing and implementing the flow solver as well as the pre- and post-processing software. Since we are primarily interested in large-scale simulations of turbulent flows in complicated domains, the design of these software tools must rely on advanced programming techniques and algorithms for successfully implementing and using the basis functions described in Chapter 2. Careful attention was paid to the design of the higher-order code using experience gained from previous linear basis implementations wherever possible. The flow-solver implementation described in the present work requires relatively few modifications to a highly optimized, parallel, linear element Navier-Stokes solver, enabling us to maintain this efficiency for higher-order elements. Maintaining the basic structure (and thus, efficiency) of the flow solver has its cost, though, placing much more of the design burden on the pre- and post-processing software. It should be pointed out, however, that advanced optimization techniques to improve the higher-order implementation have not been investigated here, and it is expected that efficiency improvements can still be made. Modification of the methodology described here may be required when non-uniform k-adaptivity is used, as well as dynamic h-adaptivity. However, since the target problems for the present research involve turbulent flows where long time integrations are necessary, it is expected that adaptivity (h and k) will be accomplished at relatively large time intervals (not at each time step), which will reduce the relative cost of pre-processing, which in this case would be carried out after each modification to the basis or mesh.

Pre-processing the finite element data for numerical simulations of the Navier-Stokes equations is becoming an increasingly more important stage of the analysis process. The set up of boundary conditions and communication data structures should be carried out during the pre-processing phase of the analysis, whenever possible, to ensure efficient computations for large-scale simulations. In addition, hierarchical basis functions also need more advanced data structures than Lagrange basis functions, which are also set up during pre-processing. A general methodology is put forth for geometry based boundary condition specification, techniques for set up and use of compact data structures for element level computations are presented, and the design of efficient parallel communication structures is described. Also presented in this chapter is a new methodology for creating h-adaptive meshes for time dependent problems based on collecting error indicators from time-averaged statistical quantities. This h-refinement is combined with the uniform, higher-order k basis to produce highly accurate simulations.

4.1 Overview of parallel computing environment

A methodology has been developed, as mentioned above, wherein the actual computational code uses compact data structures similar to the standard finite element data structures (described below). To maintain the generality of the basis function constructions, the full mesh data structure is used by the pre-processor (and post-processor). This creates an efficient computational environment, without sacrificing the generality of the mesh database, which is well suited for tasks such as setting up boundary conditions and creating communication structures. The rich data structures are once again used for post-processing. It must be admitted that this type of environment requires us to effectively maintain two sets of data files, one containing the rich mesh database and another containing the compact structures, however the efficiency gained is well worth this additional cost. An overview of the methodology is illustrated in Figure 4.1.

Figure 4.1 highlights the key technologies that are involved in the entire simulation process, and provides the “flow” of events that takes place. Software tools are enclosed in rectangular boxes, inputs are enclosed in diamonds (inputs may be in the form of data files or interactive user input as in the assignment of attributes to the geometric model), and decision processes are shown in circles. Arrows are included to indicate the direction of “flow”; bold lines indicate operations that may take place multiple times during a simulation, while thin lines indicate an event which occurs only once. The user begins with a geometric model and assigns various attributes in a Graphical User Interface (GUI) to specific model entities (e.g. boundary and initial conditions). The pre-processor then reads the output from the GUI as well as the classified mesh (and model), and associates

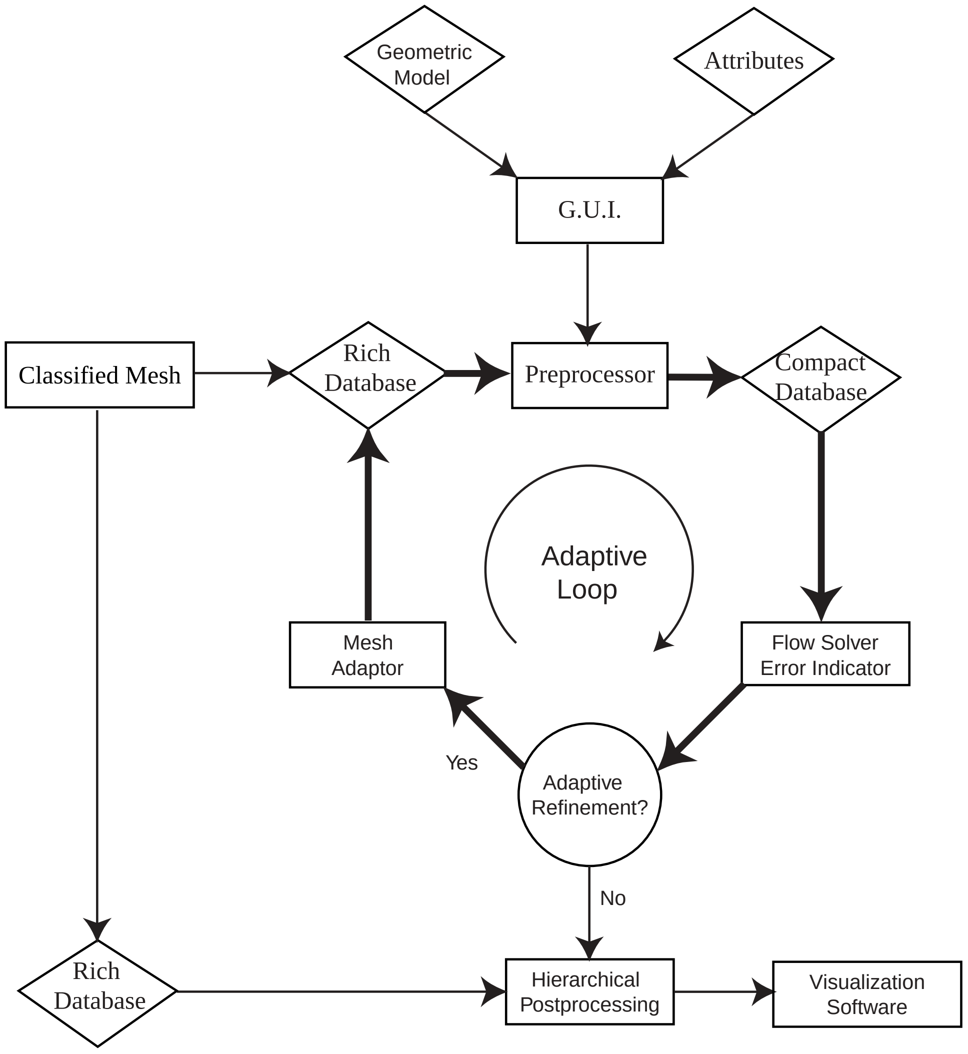


Figure 4.1: Computing environment

the attributes with the individual mesh entities, collecting all data into the compact data structure to be used in the analysis code. If mesh adaptivity is desired, statistics based error indicators are collected during the solve, and used to adaptively refine the mesh using SCOREC meshing tools (see de Cougney and Shephard [18]), generating a new rich data structure and subsequently re-entering the pre-processor. If adaptivity is not desired, the post-processing software is invoked and ultimately the data is visualized with external, third-party software.

4.2 Compact data structure

To maintain efficiency for large-scale fluid dynamics simulations, the rich mesh data structure is used only in the pre-processing phase of the analysis. A compact data structure has been designed which includes all information necessary for higher-order degrees of freedom, while maintaining the simplicity of the traditional finite element data structures which store only nodal (vertex) coordinates and element connectivity information. The compact data structure stores the degree of freedom connectivity information and nodal coordinates as well as information indicating the sign of each basis function. It should be pointed out that if curved elements are used, the higher-order coefficients related to their approximation may also be computed at this time. The compact data structure presented here is also limited to cubic (and lower order) basis functions. For higher than cubic, an additional data structure must be added which indicates which set of linearly independent basis functions are being used for each mesh face (see Dey [21]).

The first step in the setup of the data structures is the assignment of global equation numbers to all mesh entities. This is done by visiting each mesh entity, determining the number of shape functions it contributes based on their polynomial order (as described in Chapter 2), and assigning a unique equation number for each of these functions. Next, all elements (regions) in the mesh are visited, and the equation numbers associated with its bounding lower-order entities are collected, e.g a tetrahedral region may collect equation numbers from 4 vertices, 6 edges, and 4 faces (and itself if $k > 3$). This procedure is similar to that described by Hughes [42] for meshes of Lagrange elements where the global node numbers associated with each finite element are stored in the data structure. For hierarchical basis functions, additional information needs to be maintained (for $k > 2$) which emanates from the mapping from entity to element.

This connectivity information provides a complete description of the mapping between the element level computations and the global degrees of freedom (where the linear equations are set up and solved). For hierarchical basis functions of degree 3 and higher, some of the basis functions need to have their sign reversed since the mapping from the entity to the element coordinate system introduces a sign change for some of the bounding elements. The situation is illustrated by a simple example shown in Figure 4.2. In the figure a 2-D situation is shown in which two mesh faces share a common edge; we will

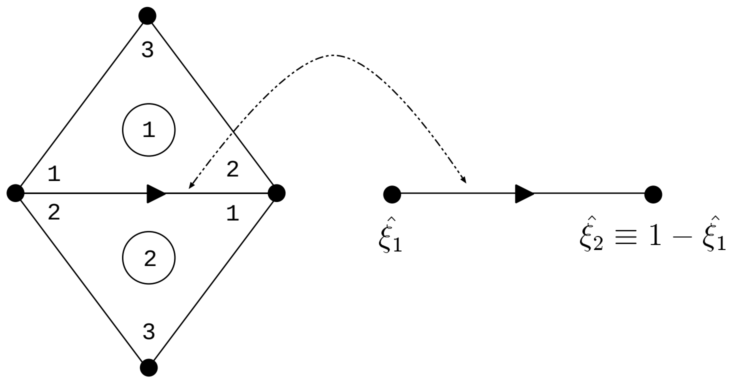


Figure 4.2: Mesh elements illustrating the reversal of shape functions

consider each face as an element (this generalizes to regions in 3-D). This figure shows the element numbers in circles, as well as each local degree of freedom number with respect to each of the elements. The edge is also shown along with its local coordinate system, $\hat{\xi}_1$, which is directed as indicated by the arrow (note that the global direction of the edge is determined by the vertex ordering stored in the mesh database). As described in Section 2.2.1, the local coordinates of the edge must be mapped to the coordinate system of each bounding element in order to evaluate the function. With the data structure described here, it is possible to evaluate a single set of element shape functions to be used for all elements in the mesh. This enables the basis functions to be pre-computed and tabulated for each quadrature point, as commonly done for Lagrange-type elements.

Returning to the example, suppose $k = 3$ has been set on the edge depicted in Figure 4.2. It will therefore contribute two functions, one quadratic and one cubic, to the local basis of each of the two bounding triangular elements (see Chapter 2), given by

$$N_2(\hat{\xi}_i) = -2\hat{\xi}_1\hat{\xi}_2 \tag{4.1}$$

$$N_3(\hat{\xi}_i) = -2\hat{\xi}_1\hat{\xi}_2(\hat{\xi}_2 - \hat{\xi}_1) \tag{4.2}$$

where the parametric coordinates for this edge are

$$\hat{\xi}_1 \quad \text{and} \quad \hat{\xi}_2 \equiv 1 - \hat{\xi}_1 \tag{4.3}$$

and the subscripts on the basis functions refer to their respective polynomial orders. When the coordinates are mapped from the edge to the element coordinates, the problem becomes apparent, i.e.

Element 1:

$$\xi_1 = \hat{\xi}_1, \qquad \xi_2 = \hat{\xi}_2 \tag{4.4}$$

Element 2:

$$\xi_1 = \hat{\xi}_2, \qquad \xi_2 = \hat{\xi}_1 \tag{4.5}$$

and the basis functions become:

Quadratic:

$$\begin{aligned} N_2^{(1)} &= -2\xi_1\xi_2 && (4.6)\\ N_2^{(2)} &= -2\xi_2\xi_1 \\ &= N_2^{(1)} && (4.7) \end{aligned}$$

Cubic:

$$\begin{aligned} N_3^{(1)} &= -2\xi_1\xi_2(\xi_2 - \xi_1) && (4.8)\\ N_3^{(2)} &= -2\xi_2\xi_1(\xi_1 - \xi_2) \\ &= -N_3^{(1)} && (4.9) \end{aligned}$$

where the superscript indicates the element that the function is associated with. The cubic function on element 2 is the negative of that on element 1, while the quadratic function is the same, regardless of the edge direction. This case generally occurs when an element uses an edge in the opposite direction than the edge is defined. Since the cubic edge function is different for each of the bounding elements, a single set of basis functions will clearly not suffice to completely describe the basis. To overcome this difficulty, during pre-processing the local degree of freedom numbers that correspond to shape functions that must be negated are flagged (e.g. there equation numbers are negated). This information is then used in the flow solver to create the correct element basis functions from the pre-computed table of element functions. For quadratic or linear basis, no functions need to be negated, and the data structures may be used as they are. This also implies that when using the hierarchical code with linear elements, no significant penalty is paid for having the generality of higher-order basis functions in the same code.

In addition to element connectivity, finite element computations also rely on nodal coordinates to compute a mapping from global to element coordinates, needed to evaluate the Jacobian matrix, $\xi_{i,j}$, (the reader unfamiliar with such topics should consult a text on finite element analysis such as Hughes [42]). For straight-sided elements, a linear mapping, involving only the vertices, to element coordinates is sufficient. To accomplish this *sub-parametric* mapping within the hierarchical basis, we can simply use the vertex functions, since they are linear regardless of the polynomial order of the basis, i.e.

$$\boldsymbol{x}^e = \sum_{a=1}^{n_v} N^a \boldsymbol{x}_a^e \tag{4.10}$$

and the Jacobian can be formed by inverting

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}^e = \sum_{a=1}^{n_v} N^a_{,\boldsymbol{\xi}} \boldsymbol{x}_a^e \tag{4.11}$$

to obtain $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}^e$. Here, n_v is the number of vertices per element. In this case, only the coordinates of the vertices need to be stored in the data structure. For meshes containing curved elements, additional information must be stored to compute these nonlinear mappings. This is a detailed topic, and a thorough discussion may be found in the work of Dey [21]. It suffices to say that no matter what technique is used to create the higher-order mapping, the basis coefficients necessary to compute the mapping can be included in the coordinate data structure, and the mappings in equations (4.10) and (4.11) will be summed over n_{es}, the number of element shape functions, rather than simply the number of vertices, n_v. All of the computations performed in the present work use a linear mapping, since in most cases considered, the geometries are well represented by straight-sided elements.

4.3 Post-processing hierarchical solutions

4.3.1 Effective visualization of hierarchical solution data

Post-processing 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 straight-forward approach is to generate a refined "visualization" mesh, evaluate the hierarchical solution at each of the new nodes, and generate new element connectivity before using a standard, linear visualization package. The process is illustrated in Figure 4.3. Here there have been three new vertices created on each original mesh edge, yielding a total of 16 new elements for each original element. Details of the algorithm used to generate these visualizations are given in Appendix B.

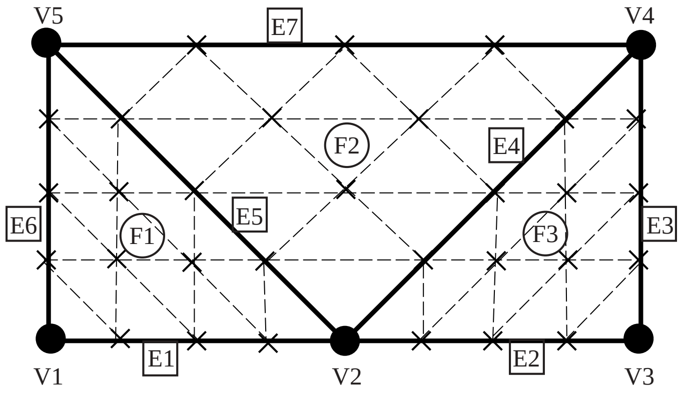


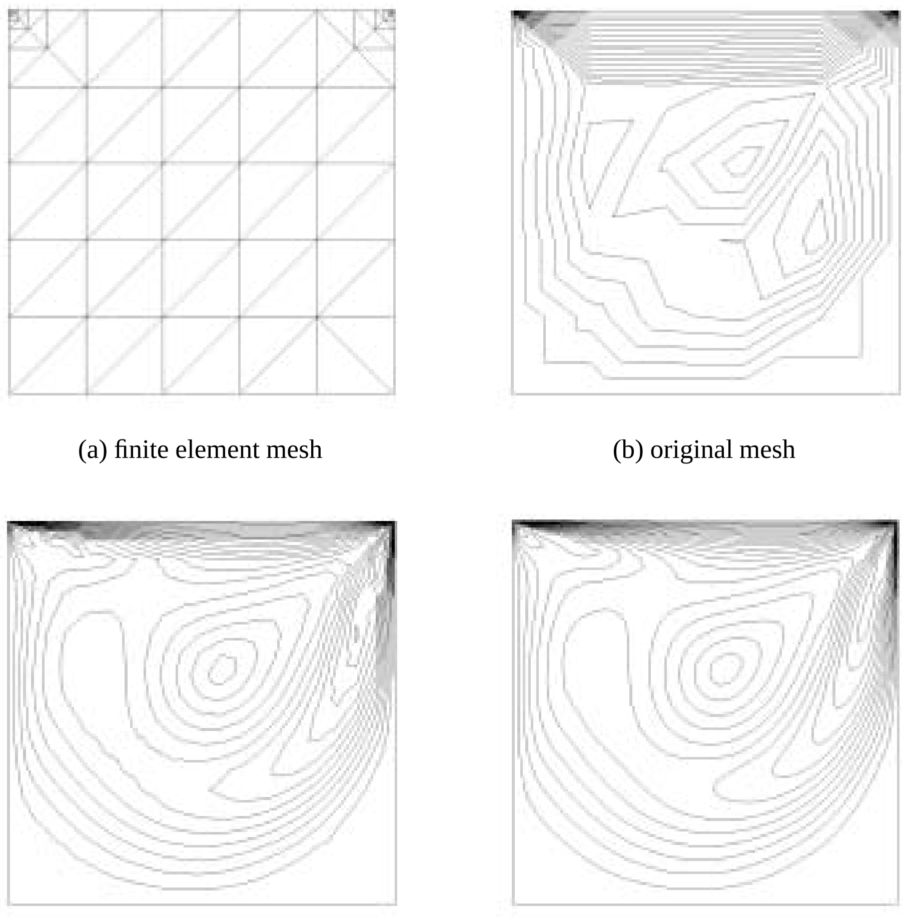
Figure 4.3: Mesh subdivision for post-processing, new mesh shown with $\times$'s and dotted edges, original mesh with solid edges

These concepts can be illustrated by means of a simple example. Consider Figure 4.3, which displays a single model face which originally had 3 triangular mesh faces (F1-F3), 7 mesh edges (E1-E7), and 5 mesh vertices (V1-V5); each of the triangles will be sub-divided into new elements and nodes, and the hierarchical solution will be evaluated at each of the new nodes. In this case, the user has specified 3 refinement levels (i.e. 3 new vertices for each original mesh edge, $N_{vis} = 3$). This new mesh will be output to the linear visualization package. The number of refinement levels necessary to achieve good results depends on the problem at hand. For coarse cubic meshes, typically 10 to 15 new nodes per edge are required in the visualization mesh, while more refined quadratic meshes may require only 2 or 3 new nodes. To create a good visualization, the number of subdivisions is successively increased until the resulting visualization no longer visibly changes. This technique is currently used on 2-D model faces (of 3-D meshes), so the time required to generate the visualizations is insignificant (even for a large number of subdivisions, e.g. 10-15), since no search is required.

To emphasize the importance of this post-processing technique, we consider the cubic simulation of a lid-driven cavity at $Re = 400$ on a very coarse mesh. This problem will be revisited in greater detail later. Figure 4.4 shows contours of fluid speed for the cubic simulation on the 5×5-element mesh along with three different levels of post-processing refinements. These plots clearly show the strong advantage, in fact the necessity, of visualizing the higher-order contributions of the solution, i.e. visualizing only the linear modes seriously degrades the solution quality.

4.3.2 Line plots of hierarchical solutions

Line plots of solution quantities are obtained by evaluating the (higher-order) finite element solution at a series of locations in the global coordinate system. This operation involves a search through the elements to determine what element a given global point lies in before the solution is evaluated. The search speed is improved by taking advantage of the adjacency relationships that are available in the mesh data structure, after a point is found (in an element), a pointer to that element is stored. If the next point lies within the same element, it is immediately returned, if not, the point is sought in the neighboring elements. While this approach can be costly for low polynomial order solutions where a large number of elements are required, the burden is reduced with high-order solutions due to the dramatic reduction in the number of elements necessary to attain solutions of similar quality. More advanced search algorithms could be investigated to reduce the burden for low polynomial order methods, containing more elements to search. All line plots presented in the current thesis were generated using these techniques, and took very little time to complete. The number of sampled points depends on the polynomial order, e.g. linears sample approximately at the same spacing as the vertices, while higher-order



(a) finite element mesh

(b) original mesh

(c) 16 new elements per face

(d) 100 new elements per face

Figure 4.4: Fluid speed for 5×5 lid driven cavity with successively refined visualization meshes

simulations typically take several points per element to achieve acceptable results.

4.4 Application of boundary and initial conditions

Essential and natural boundary condition attributes may be quite complex for fluid dynamics simulations, where the user may wish to set different boundary conditions on different portions of the geometric model. In addition, periodic boundary conditions are handled differently than other essential boundary conditions and pose additional difficulties for parallel communication since periodic partners may lie on different processors.

There are several different ways in which boundary conditions may be set up and used by a finite element code. The classical approach is to define the boundary conditions based on a pre-determined knowledge of where each vertex (or node) is physically located in space, and this information is pre-processed. This procedure is cumbersome, since it relies on a separate program for each new physical problem that is to be solved, possibly even for each new mesh. The advantage of this approach, however, is that since the boundary conditions are associated during pre-processing, there assignment in the flow solver is extremely efficient. A second approach takes advantage of the geometric model and mesh classification information. When boundary conditions are set, the geometric model entity on which the mesh entity is classified is queried to determine if a boundary condition has been specified. If so, the function is evaluated, the degrees of freedom are constrained, and the corresponding basis coefficients are set to their appropriate values. This method is more general (and also more costly) than the classical approach since all spatial information for the mesh entities is contained in their classification information, and boundary conditions are assigned with respect to the geometric model entities in a mesh-independent manner. An additional constraint when using this method is that the flow solver must maintain the mesh-model classification information, currently only used by the pre-processor.

The method of boundary condition application presented here is a combination of these two approaches. We propose to compute the boundary condition coefficients using the pre-processor with the full mesh-model classification information. This enables us to retain the generality of geometry based boundary condition specification and the efficiency of pre-processed boundary condition data structures. It also enables us to more easily set boundary conditions on the higher-order modes attached to mesh edges and faces.

Higher-order simulations using Lagrange basis functions enforce essential boundary conditions in a relatively straightforward manner, as the basis coefficients correspond to solution values at nodes, *vis.* the Lagrange interpolation equation $N_a(\boldsymbol{\xi}_b) = \delta_{ab}$. Since the solution coefficients with respect to the Hierarchical basis do not correspond to solution values at spatial locations, more work must be done to determine the coefficients to impose as the boundary values. To accomplish this, we will interpolate the known Dirichlet boundary function with the hierarchical basis by solving a linear system of equations for the unknown basis coefficients. Additionally, a unique set of interpolation points must be chosen since there are no particular spatial locations associated with the higher-order coefficients.

The element level interpolation may be constructed by solving a linear system of equations for the coefficients on each element in the domain. Suppose we wish to specify that $\phi(x_i) = g(x_i)$ over some portion of the boundary, where $\phi(x_i)$ could be any of our solution variables. (This process may be trivially generalized to include initial conditions, in which case we seek an approximation over the entire domain, not just the boundary face.) We can find the coefficients of an approximation to $g(x_i)$ (for each element, e) as

$$g(x_i) \approx \hat{g}^e(x_i) = \sum_{a=1}^{n_{ip}} g_a^e N_a^e \tag{4.12}$$

where N_a^e are the basis functions for element e, g_a^e 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.,

$$\boldsymbol{M}\boldsymbol{g} = \boldsymbol{R} \tag{4.13}$$

$$\boldsymbol{M} = [M_{ab}] = N_a^e(\boldsymbol{\xi}_b^{int}), \text{ and } \boldsymbol{R} = [\boldsymbol{R}_b] = g(\boldsymbol{x}(\boldsymbol{\xi}_b^{int})) \tag{4.14}$$

where $\boldsymbol{\xi}_b^{int}$ is the b^{th} interpolation point (in element e's coordinates). This system of linear equations is solved for the basis coefficients, g_a^e, which are used when needed by the analysis code to evaluate $\phi(x_i)$ (which is expanded in the same basis as $\hat{g}(x_i)$). Since we are using an element level interpolation (which only couples local degrees of freedom) the resulting interpolation is not guaranteed to be continuous between elements. One solution to this problem is to average the coefficients, which is done in the present work. Another approach is to assemble the data to global arrays and solve a global problem, however the additional cost is not deemed worth the effort, as averaging has proven to work well for all cases we have considered. For computations using the Lagrange basis, the interpolation points are simply the nodal coordinates, and the matrix in (4.13) is the identity matrix. The interpolation points used in the present work are taken from the work of Chen and Babuška [14] where they derived an optimal set of interpolation points for a tetrahedral region. The system of equations described above may be simplified somewhat by statically condensing the coefficients since not all functions are coupled. In practice, however, the interpolation is only computed during pre-processing, making the time savings less significant.

The procedure described above for essential boundary conditions may also used to set an initial condition in cases where the exact initial condition is relevant to the simulation. However, experience has shown that in cases where such accuracy is not necessary (as is usually the case), using the linear interpolation of the initial conditions is sufficient to ensure convergence. The linear interpolation is obtained by simply setting all higher-order coefficients equal to zero.

4.4.1 Periodic boundary conditions

The application of periodic boundary conditions poses additional difficulties in the context of hierarchical basis functions since all mesh entities must be identical on periodic planes (including edge and face directions). A general methodology has been developed for the application of periodic boundary conditions. The data necessary to enforce periodic boundary conditions can be contained in a single array which specifies the "periodic master" of each mesh entity. When essential boundary conditions are set, periodic boundary conditions are also set by copying the solution coefficients of the periodic masters to their periodic slaves. This operation is simply an indirect address of the solution array using the periodic boundary conditions array. The equations corresponding to the periodic entities are eliminated from the system by using this array to zero the corresponding residual components.

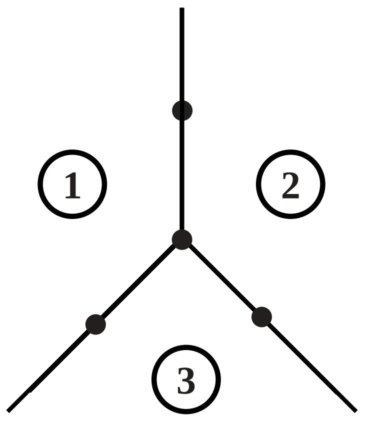
4.5 Parallel communications

Parallel computers have gained wide popularity in the finite element CFD community due to the local nature of most of the work which can be trivially parallelized and will be shown to yield nearly perfect scalability (98%) on large problems. We present a methodology for pre-processing the necessary data structures to be used in conjunction with the MPI library of message passing routines. The method is designed based on the abstract mesh topology and easily handles *dof*'s (degrees of freedom) associated with mesh edges, faces, and regions, necessary for higher-order k simulations. The methods presented herein are a straightforward generalization of the methods used by Bastin [1] for linear basis computations.

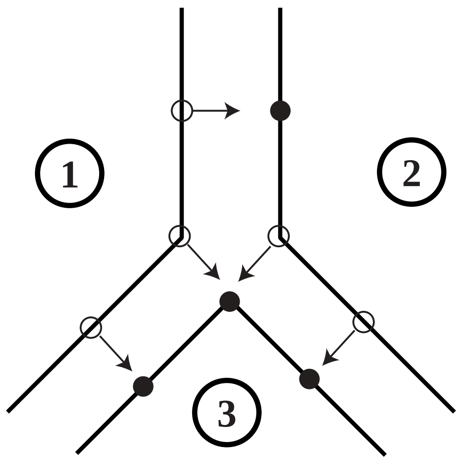
To reduce the computational effort during the analysis phase, the structures specifying the interprocessor communications are pre-processed. The domain-decomposition technique is used, whereby the finite element mesh is physically decomposed into multiple partitions (mesh partitioning software such as METIS [53] can be used), each of which is assumed to be associated with a unique processor. Each processor then executes its own copy of the analysis code, reading the pre-processed input data relating to its partition of the mesh, as well as information relating to other processors it must communicate with. This section describes the types of data that processors must communicate to each other as well as the construction of these data structures.

As mentioned above, each processor maintains a complete collection of data representing its portion of the finite element mesh and analysis information (e.g. boundary conditions). This includes equation numbers and connectivity information as well as boundary condition data for all nodes that are physically on the partition or its interprocessor boundary. The finite elements are uniquely partitioned among the processors, so each element will be found on only one processor. The other mesh entities (faces, edges, and vertices) that contribute to the element level integrals, however, appear in multiple partitions if they are on an interprocessor boundary. Element level computations are performed completely local to each processor and must be communicated only when assembled to the global equations. In fact, only degrees of freedom associated with the interprocessor boundary must be communicated, all others are assembled locally, on their respective processor. The global assembly procedure involves the sending and receiving of *dof* information between processors using MPI library functions to carry out these tasks. Another case in which processors must exchange information is when periodic boundary conditions are present with periodic partners residing on different processors. These two types of communications will be described below.

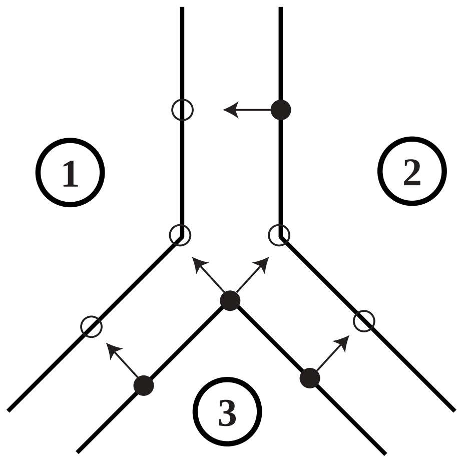
The basic idea behind the parallel communication of finite element information can be described with the aid of Figure 4.5. This first of the three figures 4.5(a) illustrates



(a) Composite mesh



(b) Type 1 communication: slaves send and masters receive and add



(c) Type 2 communication: masters send to slaves

Figure 4.5: Parallel communication

the intersection of three processors and vertices on an interprocessor boundary. For simplicity, only vertices are shown since edges and faces are handled the same way. The element residuals associated with each of these vertices is first assembled from elements on each of the bounding processors. After this local assembly, these values are sent in the direction of the arrows to the values on the master processor and added. This is illustrated in Figure 4.5(b), where the master vertices are shown as solid dots, and the slaves as circles. The sending processor (referred to as the slave) then zeroes its residual values, essentially removing this vertex from the slave processor's system. In this manner, all equations associated with entities on the interprocessor boundary are only solved by a single processor, known as the entity's master image. After these equations are solved, the solution values are copied from the entity's image on master processor to all of its slave images as shown in Figure 4.5(c). The creation of the necessary data structures and the execution of these tasks is described in the next section.

4.5.1 Implementation of parallel communication structures

For the processors to exchange the degree of freedom information during the computation, each must maintain a data structure describing its communications. The MPI routines MPI_send and MPI_recieve use this information to exchange data as described below. The procedure is as follows:

1. Each processor first computes its element level residual vector and tangent matrix values without any need for communication.

2. These element level contributions are assembled to global arrays (on processor) using the traditional finite element assembly procedures (Hughes [42]).

3. An additional interprocessor assembly (described above) is then performed between processors to account for *dof*'s on the interprocessor boundary as shown in Figure 4.5(b).

4. Parallel communications are also carried out during the solution of the linear system of equations. Before and after each sparse matrix-vector product, data must be exchanged.

5. Finally, after each Newton iteration or time step, the solution values related to mesh entities on the interprocessor boundary are copied to all of their images on each of the adjacent processors (Figure 4.5(c)). This includes periodic boundaries.

Let us first define

Definition 4.1 *A* **communication stage** *is defined as the process which involves all processors making all their necessary communications. There are two types of communication stage. Type 1: residuals are added from the slaves to the masters, then zeroed on the slave. Type 2: solution data is copied from the masters to the slaves*

Each communication stage consists of each processor sending data to and receiving data from each processor with which it must communicate (as determined by the partition). The cases in which two processors will need to communicate have been described above and we will denote by N_P^i the number of processors with which processor i must communicate. Both types of communication stage require the same information and differ only slightly. A single type 1 communication stage is necessary each time the element level residual formation and local assembly is completed and a type 2 communication stage is necessary each time the boundary conditions are set on the solution vector.

From the perspective of a single processor, say i, a communication stage may be described as a sequence of *tasks*, defined as

Definition 4.2 *A* **communication task** *(or a* **task** *for short), denoted* $T_j^i,\ j = 1 \ldots N_P^i$*, is a communication between processors* i *and* j *in which data is exchanged.* N_P^i *represents the total number of processors with which processor* i *must communicate.*

To minimize the communication overhead, we require that two processors may communicate only one time during each communication stage. This forces us to designate one of the processors in the task as master, and one as slave. This dictates which processor will be master to each of the mesh entities on the interprocessor boundary. Since only one communication can occur between any two processors, the set of tasks, T_j^i, can be represented as a directed graph, with vertices and edges of the graph representing processors and communications, respectively. This directed graph indicates which processor each mesh entity will be solved on, it therefore must yield a unique master for each mesh entity. For example consider the communication between processors 1 and 3 in Figure 4.5(a). If the direction between these two processors was reversed, the vertex at the intersection of all three processors would have no unique master. This requirement poses the additional constraint that the graph be *acyclic*, i.e. contain no closed loops. A graph of this type is commonly referred to as a Directed Acyclic Graph, or DAG, and there are many advanced techniques to generate a DAG from a graph (see Sedgewick [60]). The procedure used here to create the directed graph is as follows:

1. Mesh partitioning software such as METIS [53] is used to assign each mesh region to a unique partition.

2. Each mesh entity is visited and the processor ID number for each of its adjacent regions is associated with the mesh entity (only once if multiple bounding regions are in the same partition).

3. These processor adjacency sets (for each mesh entity) are used to create the partition graph (no direction on graph edges yet).

After the graph is set up, a simple method is used to create a directed graph from the undirected one. The direction associated with T_j^i is set to point to the greater of i and j, which is guaranteed to produce a DAG. A consistent graph having been created, it is a simple matter to visit each mesh entity and associate the unique processor which is to be its master.

To use MPI to carry out the communications described above, details of the data to be exchange between processors i and j must be provided for each task, T_j^i. To this end, T_j^i has associated with it the following integer data:

tag: A unique tag associated with T_j^i which distinguishes this send and receive for the MPI functions.

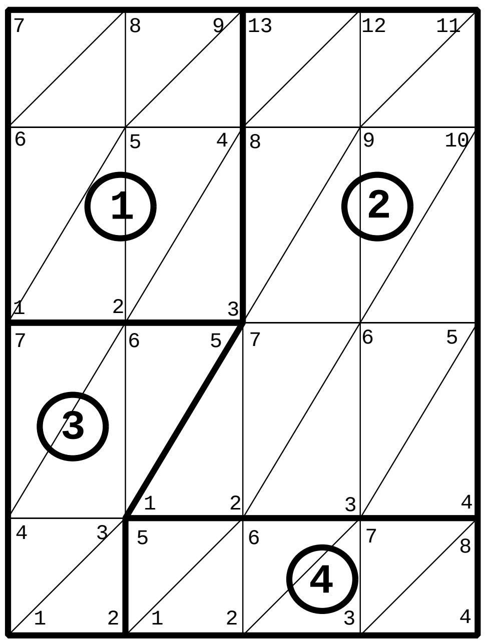
type: Denotes whether this processor is master ($= 1$) or slave ($= 0$) in the current communication. For a type 1 communication, a master calls MPI_receive(...) to receive and add the data, while a slave calls MPI_send(...) to send the data. In a type 2 communication, the slaves receive and copy and the masters send; there is no need to zero anything.

partner: PID number of the partner processor involved in this task.

numSeg: Number of data segments to be sent or received (see below).

segData: For each of the numSeg data segments, the local *dof* numbers on the other processor to send to or receive data from.

Here a data segment is defined as a continuously numbered group of *dof* numbers pointing to data such as residual or solution values. Each segment also contains its length and starting index. In the beginning of execution of the analysis code, the segment data is used in conjunction with the MPI_TYPE_HVECTOR(...) function to create new MPI data types which are used during the communication stages. These data types are simply masks that describe where information can be found on the various processors for each of the segments in T_j^i.


(a) Mesh

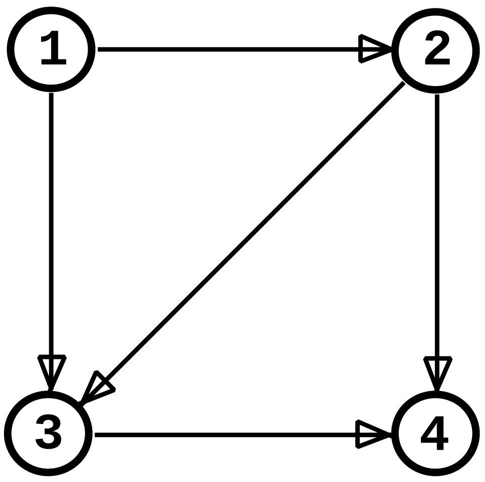

(b) Graph

Figure 4.6: Sample multiprocessor mesh and associated communication graph

4.5.2 Multiprocessor communication example

The above concepts can be clarified through a simple example. Consider the 2-d mesh shown in Figure 4.6(a) which is decomposed into four partitions. This mesh can be considered as a single geometric face of a 3-d model. The bold encircled numbers indicate processor ID numbers and the small numbers indicate local *dof* numbers on each partition. For simplicity, only vertex numbers are shown, however, edges (if $k \geq 2$) and faces (if $k \geq 3$) also get *dof* numbers associated with them and are handled identically to vertices. We also assume, for simplicity of discussion, that there are no periodic boundary conditions applied to the geometric model. A consistent graph corresponding to this mesh, created using the algorithm described above, is shown in Figure 4.6(b). Let us consider only the tasks associated with processor 2, T_j^2, where $j = 1, 3, 4$, since $N_P^2 = 3$. T_1^2 involves a communication where processor 2 is master, and *dof*'s 3, 4, and 9 on processor 1 are received and added to the contributions of *dof*'s 7, 8, and 13, respectively, on processor 2. The other two tasks associated with processor 2, T_3^2 and T_4^2, are both slave communications. Here, *dof*'s 7 and 1 are sent to processor 3 (where they are added to 5 and 3, respectively) and $1 - 4$ are sent to processor 4 and added to $5 - 8$, then these values are zeroed on processor 2.

The Fortran90 code fragment given in Program 4.5.1 illustrates how these data structures are used within the analysis code to carry out a type 1 communication stage. In this program listing, `global` is a double precision vector to be operated on and `ilwork` is the local work array which contains the integer data described above. It should be emphasized that each processor uses its own unique `ilwork` array using common tag numbers to match segments for processor pairs as described above.

4.5.3 Parallel scalability

To demonstrate the effectiveness of the parallel implementation, we will consider the time dependent flow around a square cylinder at a Reynolds number of 100 (based on the cylinder edge length). A more complete description of this problem will be presented later. This flow was advanced in time (at a time step of 0.1) from an initial condition of a shedding solution for 50 time steps with 2 Newton iterations per time step. The problem was solved on 1, 2, 4, and 8 processors and the results are shown in Figure 4.7. This figure shows the total solution time multiplied by the number of processors *vs.* the number of processors, normalized by the time for the single processor simulation. It is clear from the figure that the parallel implementation is nearly perfectly scalable (98%).

The cost of the parallel communication also depends, to some extent, on the polynomial order of the basis. As some detailed timings show (see Chapter 5) the distribution of cost between element level computation and linear solver, also has an effect on the

Program 4.5.1 Type 1 communication stage

```
  dimension global(nshg,n), temp(max), ilwork(nlwork)
  ...
  numtask   = ilwork(1)
  do itask  = 1, numtask
    itag    = ilwork (itkbeg + 1)
    type    = ilwork (itkbeg + 2)
    partner = ilwork (itkbeg + 3)
    numseg  = ilwork (itkbeg + 4)
    isgbeg  = ilwork (itkbeg + 5)
    if (type .EQ. 0) then
      call MPI_SEND(global(isgbeg,1), 1,
&                  sevsegtype(itask,kdof),
&                  partner,          itag,
&                  MPI_COMM_WORLD,   ierr)
    else
      lfront = 0
      do is = 1,numseg
        lenseg = ilwork (itkbeg + 4 + 2*is)
        lfront = lfront + lenseg
      enddo
      call MPI_RECV(temp(1), lfront*n, MPI_DOUBLE_PRECISION,
&                  partner, itag,     MPI_COMM_WORLD,
&                  status,  ierr)
      itemp = 1
      do idof = 1,n
        do is = 1,numseg
          isgbeg = ilwork (itkbeg + 3 + 2*is)
          lenseg = ilwork (itkbeg + 4 + 2*is)
          isgend = isgbeg + lenseg - 1
          global(isgbeg:isgend,idof) =
&                            global(isgbeg:isgend,idof)
&                          + temp (itemp:itemp+lenseg-1)
          itemp = itemp + lenseg
        enddo
      enddo
    endif
    itkbeg = itkbeg + 4 + 2 * numseg
  enddo
```

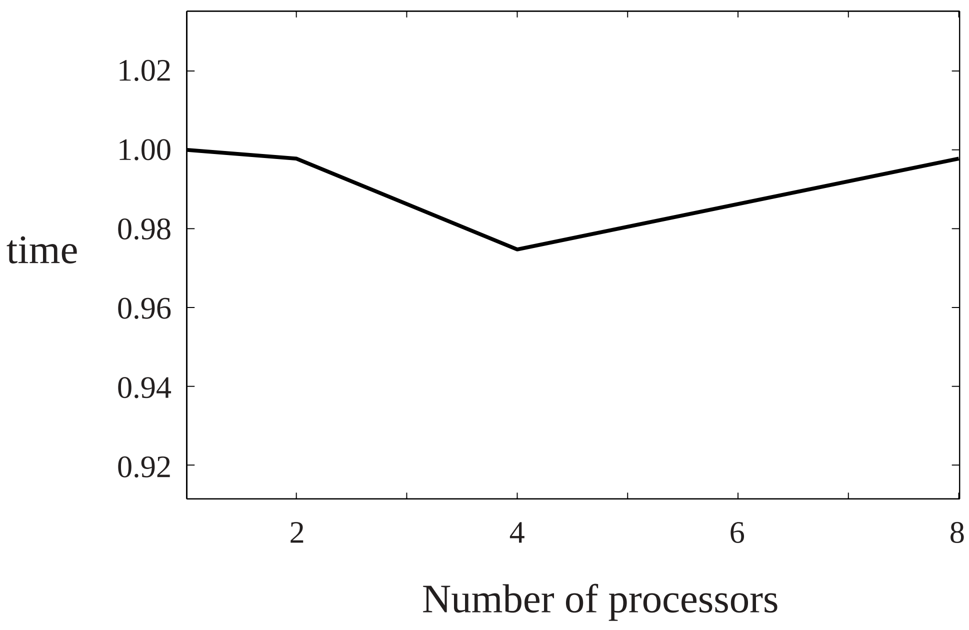


Figure 4.7: Parallel efficiency for vortex shedding problem. Time is the aggregate time used by all processors.

parallel communications. For example, the cubic basis spends the majority of its time computing and forming the tangent matrices, thus involving fewer communications. In all cases, however, the cost of communication was less than 15% of the total simulation time.

4.6 Statistics based error indicators for h-adaptive simulations

As pointed out in the Introduction, some level of h-refinement is necessary in conjunction with the use of higher-order basis functions to achieve the most accurate and cost effective simulations possible given certain cost restraints. Traditional methods of error indication and estimation are not commonly used or easily applied to time dependent simulations that often contain structures that evolve in time. If mesh adaptation is carried out at each time step, much effort may be used creating a refined mesh for a flow structure that is quickly convected from that portion of the mesh (i.e. the mesh refinement must constantly follow the flow structures). However, many problems of scientific and engineering interest contain regions of the flow that are statistically stationary, while the instantaneous structures are rapidly varying in time. For example the turbulent flow in a boundary layer has a well defined region where the flow is statistically steady, although the turbulent eddies are constantly evolving. Attempting to resolve a turbulent flow of this type based on an instantaneous flow state may not be the most efficient approach.

To remedy this situation, we introduce new error indicators based on time-averaged statistical quantities. Based on this methodology, the error indicators are collected during several hundred or thousand time steps, and evaluated. Based on these quantities, the mesh is adaptively refined using SCOREC meshing tools (see de Cougney and Shephard [18]). It should be noted that there is still no general theory to guide the use of these new error indicators, though the preliminary results are promising. These new adaptive refinement techniques are demonstrated below on the problem of vortex shedding behind a square cylinder for uniform polynomial orders. This flow is well-suited for demonstrating the ideas since it has a long wake region with a periodic vortex street, which for a small time window is rapidly varying, but is statistically steady over large time windows.

Several error indicators are proposed, and are given by the following expressions:

$$\frac{1}{T}\sum_{n=1}^{N}\int_{\Omega} w\bar{\mathcal{L}}_i^2 \; d\Omega \tag{4.15}$$

$$\frac{1}{T}\sum_{n=1}^{N}\int_{\Omega} w q_{i,i}^2 \; d\Omega \tag{4.16}$$

$$\frac{1}{T}\sum_{n=1}^{N} {u'}_i^2 \tag{4.17}$$

$$\frac{1}{T}\sum_{n=1}^{N} {p'}^2 \tag{4.18}$$

Here, $\bar{\mathcal{L}}_i$ is the residual of the strong form of the Navier-Stokes equations using the conservation restoring advective velocity, $\overset{\Delta}{u}_i$, i.e.

$$\bar{\mathcal{L}}_i = \dot{u}_i + \overset{\Delta}{u}_j u_{i,j} + p_{,i} - \tau_{ij,j} - f_i \tag{4.19}$$

We have also used u'_i and p' to represent the fluctuating portion of the velocity and pressure field (similar to the quantities introduced for turbulence statistics). The sums in Equations (4.15) through (4.18) indicate that these quantities are time-averaged by collecting them at each time step and averaging over the number of time steps. The adaptive procedure is accomplished by running the time dependent simulation on a coarse mesh and collecting these quantities. These time-averaged quantities are then visualized and combined to yield an indication of where the mesh should be refined. Meshing tools available in the SCOREC software system are then used to refine the mesh entities that are identified based on the chosen error indicators.

4.7 Chapter summary

This chapter discussed several issues relevant to fluid dynamics computations using hierarchical basis functions. It is clear from this chapter that we advocate the use of pre-processed data structures for element level computations, boundary condition data, and parallel communication, by using the rich mesh database to create these structures using pre-processing software. This has allowed us to maintain the efficiency of a highly optimized linear basis Navier-Stokes solver. Post-processing hierarchical basis simulations was also discussed in some detail, as it is more involved than visualizing simulations based on the Lagrange basis. The techniques described in this chapter were all used to set up and analyze the simulations presented in the following chapters.

CHAPTER 5
NUMERICAL EXAMPLES: STEADY AND UNSTEADY LAMINAR FLOW

This chapter presents numerical simulations which are used to explore the methods introduced in the previous chapters. The problems presented here will verify the convergence rate of the finite element formulation and quantify the cost associated with various polynomial order basis functions. This chapter will demonstrate the cost effectiveness of the higher-order basis functions when compared with the linear basis. Also presented here will be results for h-adaptive simulations based on time averaged statistics, for uniform polynomial order simulations.

5.1 Kovasznay flow

The first simulation is used to verify the convergence rate of the finite element formulation. It is well known that the interpolation error should converge at a rate of h^{k+1}, where h is a suitable measure of the element size (see Johnson [52]). Since we are simulating this flow on a structured, uniform grid, h is simply taken as the length of the element in the x_1 or x_2 direction, (i.e. Δx_1).

The Kovasznay flow may be identified with the incompressible flow some distance downstream from a rectangular grid (see Kovasznay [56]) and has a closed form analytical solution given by:

$$u_1 = 1 - e^{\lambda x_1}\cos(2\pi x_2) \tag{5.1}$$

$$u_2 = \frac{\lambda}{2\pi}e^{\lambda x_1}\sin(2\pi x_2) \tag{5.2}$$

with

$$\lambda = \frac{\text{Re}}{2} - \sqrt{\frac{\text{Re}^2}{4} + 4\pi^2} \tag{5.3}$$

and we have taken Re $= 40$ for the present study. The flow is considered on a rectangular domain of $-\frac{1}{2} \leq x_1 \leq 1$ and $-\frac{1}{2} \leq x_2 \leq \frac{3}{2}$ with the exact solution imposed as an essential boundary condition at the inflow and upper and lower walls, while the pressure was set to zero at the outflow. Since this is a non-trivial essential boundary condition, the interpolation technique described in Section 4.4 is employed to determine the basis coefficients on the boundary. Failure to correctly interpolate the boundary condition results in sub-optimal convergence rates (particularly for the higher-order simulations). The qualitative behavior of the solution is depicted in Figure 5.1 which shows contours of fluid speed for the cubic simulation on the 21×21 mesh.

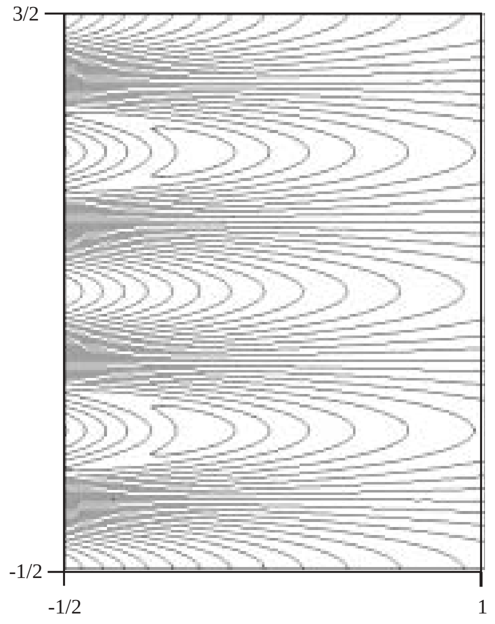
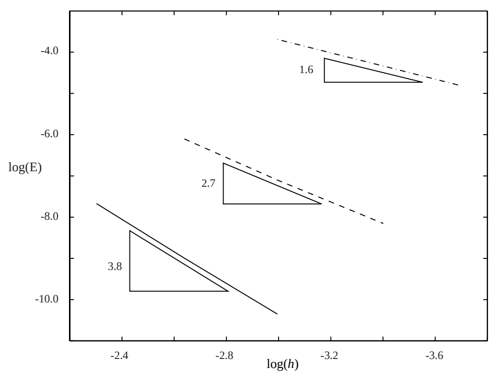


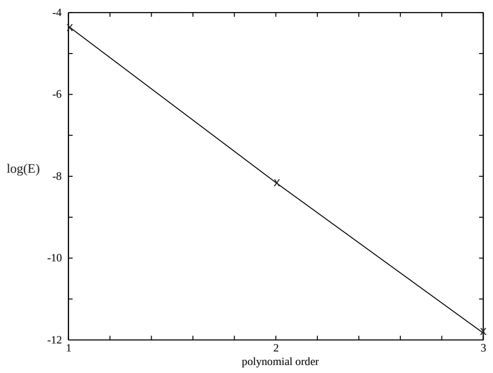
Figure 5.1: Kovasznay flow. Contours of fluid speed for cubic simulation on 21×21 **mesh**

A convergence study was performed for this flow to determine the accuracy of the different polynomial order simulations as functions of the mesh size, Δx_1, and the polynomial order of the basis. Figures 5.2(a) and 5.2(b) show the log of the normalized L^2 error in the velocity field versus $\log(\Delta x_1)$ and polynomial order, respectively. Here, the L^2 error is computed numerically from the formula

$$E^2 = \frac{\int_\Omega e_i e_i \; dx}{\int_\Omega u_i u_i \; dx} \tag{5.4}$$



(a) log of L^2 error vs. log(Δx_1). $\cdots$:$k=1$, $--$:$k=2$, and $-$ $k=3$.



(b) log of L^2 error vs. k

Figure 5.2: Kovasznay flow convergence study

where

$$e_i = u_i - u_i^{(h,k)} \tag{5.5}$$

represents the difference between the exact, u_i, and finite element, $u_i^{(h,k)}$, solutions for the velocity.

Figure 5.2(a) enables us to determine the rates of convergence of the different simulations to be 1.6, 2.7, and 3.8 for the linear, quadratic, and cubic simulations, respectively. These values compare well with the theoretical predictions for the interpolation error (i.e. 2, 3, and 4 for linear, quadratic and cubic). It is clear from Figure 5.2(a) that the constant in the error estimate also greatly improves for the higher-order simulations, making the higher-order basis most attractive even on the coarsest meshes. It should be noted that for this flow, the cubic simulation is the closest to its predicted value, while the linear is the most deficient. The under-performance of the linear and quadratic solutions may be due to the severe penalty incurred by a method's inability to capture maxima and minima of the prescribed, essential boundary conditions. Figure 5.2(b) 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, incompressible 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 [24]. The problem is specified by a fully developed flow entering a confined channel which, at $Re = 800$, has been demonstrated by numerous researchers to be steady and stable (see Gresho *et al.* [27]). A complete description of the physical problem requires modeling the region upstream of the step, and careful attention to the singularity that may develop at the step corner. However, since the objective of this study was a 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 in Gartling [24]. This allows for a more accurate comparison with his benchmark results.

The geometry and boundary conditions are shown in Figure 5.3. The initial condition consists of a parabolic (Poiseuille) velocity profile with the same mass flow-rate as the inlet 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. The steady state is also verified by steady shear stress on the channel walls.

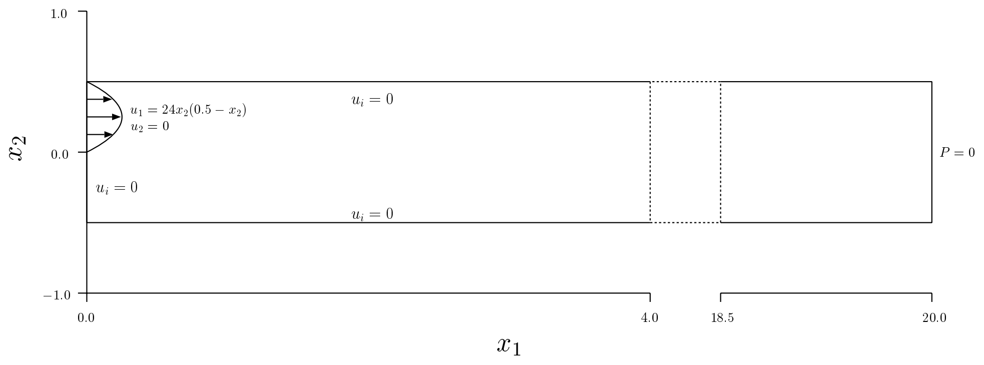


Figure 5.3: Step flow geometry and problem description

Numerical solutions were obtained on a variety of uniform tetrahedral meshes for several different polynomial orders. The mesh statistics are shown in Table 5.1 where each successive mesh represents a uniform refinement of the previous mesh, with the exception of mesh B. Here, Δx_1 and Δx_2 represent the element size in the x_1 and x_2

directions, respectively.

Mesh	Vertices	Edges	Faces	Δx_1	Δx_2
A	405	1,044	640	0.250	0.250
B	847	2,286	1,440	0.167	0.167
C	2,211	6,210	4,000	0.100	0.100
D	8,421	24,420	16,000	0.050	0.050
E	32,841	96,840	64,000	0.025	0.025

Table 5.1: Step flow mesh statistics

There are three major factors that contribute to the cost of the finite element simulations discussed here: right hand side (or residual) evaluation, left-hand-side (or tangent and mass) formation, and linear algebra (involving matrix-vector products for iterative solution techniques). The first two of these measures rely on the numerical integration of element level quantities, and the dominant terms are in the left-hand-side calculation (proportional to the number of integration points times the number of element basis functions squared). The matrix-vector products are dominated by the fill pattern of the matrix. The relative size of these different measures depends on the order of the basis being used (as well as the problem). For example, the linear basis is dominated by the linear algebra, since we are using relatively cheap integration rules. Also, the fine meshes needed for linear basis computations lead to greater cost in solving the linear system (more Krylov vectors are needed).

In Table 5.2, we are assuming that the cost is proportional to the number of elements times the number of element shape functions squared times the number of quadrature points, $C_I = n_{el} \times n_{es}^2 \times n_{int}$, and a “-” in the Cost column indicates that this simulation is not included in the study. The cost measured in this way reflects the tangent matrix formulation cost. The symbols in the far-right column indicate the meshes used in the comparison study discussed below. These calculations are in terms of the face data and face quadrature rules in an attempt to level the playing field for the 2-D comparison with the 3-D code.

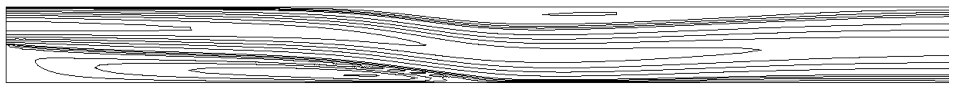
The basic character of this flow is well known. At $Re = 800$, there are two separation regions, one starting at the step corner and continuing downstream approximately 12 step heights, and another on the upper wall of the channel occupying a region

Mesh	k	n_s	n_{int}	$C_I(\times 10^3)$	
	1	405	3	-	
A	2	1,449	6	-	
	3	3,133	12	768	$+$
	1	847	3	-	
B	2	3,133	6	-	
	3	6,859	12	-	
	1	2,211	3	-	
C	2	8,421	6	864	$\circ$
	3	18,631	12	-	
	1	8,421	3	-	
D	2	32,841	6	-	
	3	73,261	12	-	
	1	32,841	3	1,728	$*$
E	2	129,681	6	-	
	3	290,521	12	-	

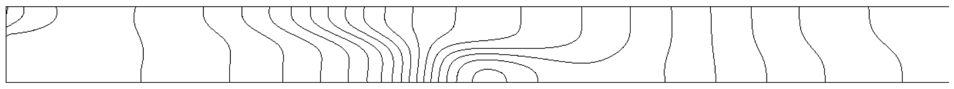
Table 5.2: Step flow simulation cost comparison

from approximately 10 to 20 step heights downstream. These features are shown in Figures 5.4(a)- 5.4(d) which 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 [24].

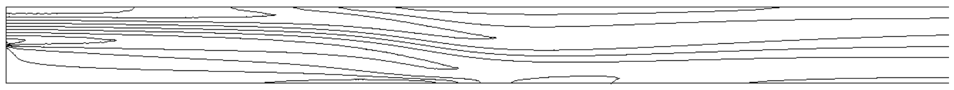
The contour plots look similar for all simulations making it difficult to quantify the benefit of the higher-order methods. We will therefore compare line plots of various flow quantities at different spatial locations. The first of these plots demonstrates that all of the methods are converging to the benchmark result (with linear being the slight exception). Figure 5.5(a) shows the (most refined) cubic, quadratic, and linear simulations on meshes C, D, and E, respectively, as well as the benchmark result of Gartling [24]. For each of these, 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 [24], 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 at the $x_1 = 7.0$ location. This isn't surprising, since the benchmark result is from a quadratic simulation with 41 vertices



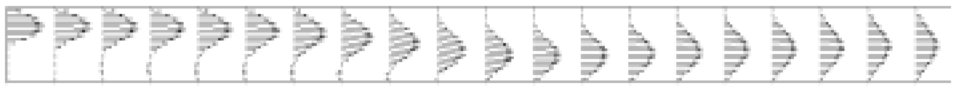
(a) contours of fluid speed



(b) contours of pressure



(c) contours of vorticity

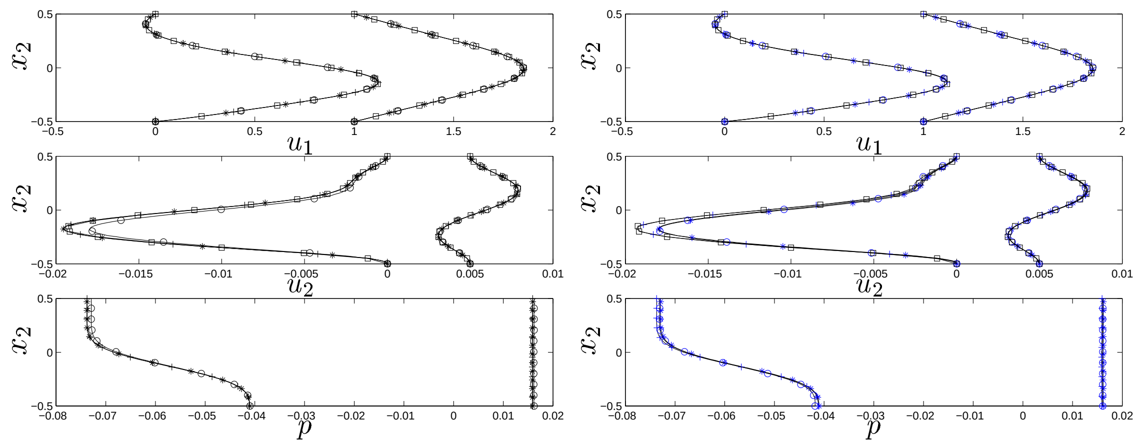


(d) velocity vectors

Figure 5.4: Step flow simulation characteristics: Mesh C, $k = 3$

across the channel, which is a higher resolution than our most refined linear simulation.

Figure 5.5(b) presents a comparison between the cubic simulation on mesh A, the quadratic simulation on mesh C, and the linear on mesh E. These three simulations represent qualitatively similar results. Clearly, the only plots that visibly differ from the benchmark result are the linear and cubic x_2 velocity at $x_1 = 7.0$, which is the most sensitive quantity in the study. Although the quadratic basis simulation is slightly better, here, the cubic and linear basis simulations appear identical. The symbols in the plots



(a) Comparison on finest meshes. $\circ$: $k = 1$, $*$: $k = 2$, $+$: $k = 3$, and $\square$: Gartling.

(b) Comparison of qualitatively similar solutions (see Table 5.2 for symbols), $\square$: Gartling.

Figure 5.5: Backward-facing step. Velocity and pressure plotted versus x_2 at $x_1 = 7$ and $x_1 = 15$. Velocities at $x_1 = 15$ were shifted for plotting.

may be found in the right column of Table 5.2 which also shows the cost index for these three simulations as 7.7×10^5, 8.6×10^5, and 1.73×10^6 for the cubic, quadratic, and linear simulations, respectively. From this, it is clear that the same accuracy can be obtained with the cubic simulation for 40% the cost of the linear, while the quadratic costs about half as much as the linear. These results are also in agreement with the results for the Kovasznay flow using the L^2 error as a comparison measure.

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 to the most refined cubic simulation at the $x_1 = 7$ location. The log of the L^∞ error (defined as the maximum difference from the reference solution) versus Δx_2 shows a slope of 2.1, which is slightly better than optimal for the meshes considered; the L^∞ error for the interpolation being $O(h^2)$ (see Johnson [52]). Quadratic and cubic solutions are too close to the reference solution (the most refined cubic) to obtain useful convergence data.

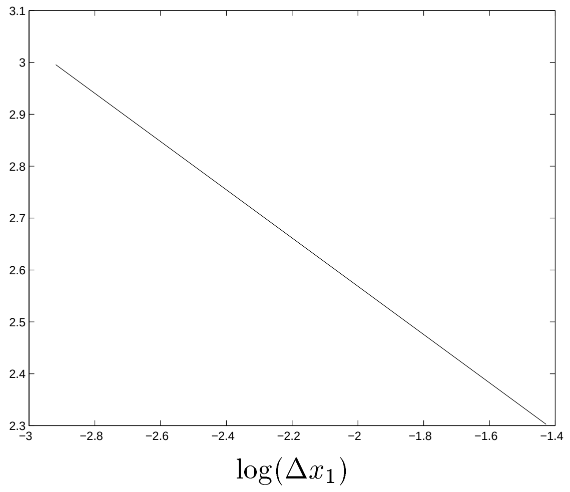


Figure 5.6: log of L^∞ error in pressure vs. log(Δx_1).

5.3 Lid-driven cavity flow

The next problem considered is the steady, two-dimensional and incompressible flow inside a closed container driven by its lid. The lid slides to the right at unit velocity 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 corners (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 exist well-established benchmark results to compare with (see Ghia *et al.* [26]). Since the velocity is discontinuous at both upper corners, singularities will develop in the pressure and stress fields, which must be controlled by the method. In addition, there are singularities also in the lower corners, however, they are well resolved by the uniform meshes.

The geometry and boundary conditions are illustrated in Figure 5.7. 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 adaptivity was used by subdividing 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) and is

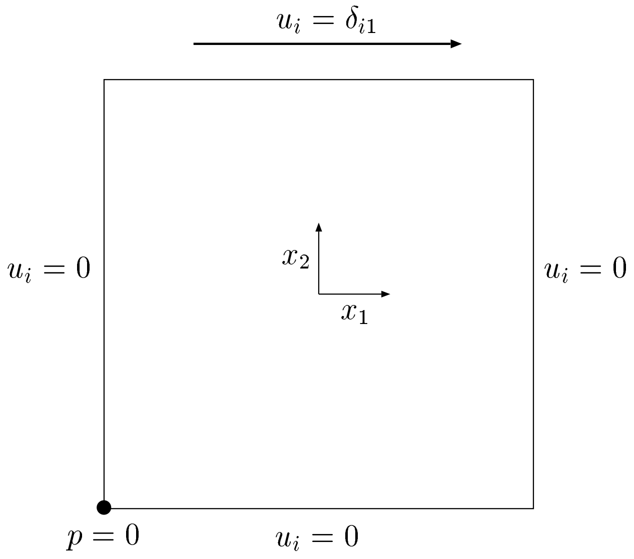


Figure 5.7: Lid-driven cavity geometry and boundary conditions

fixed at the given value for all simulations by changing the number of corner elements. This procedure is crucial to obtaining identical solutions for the different polynomial orders, since the actual mesh size varies dramatically between the least refined cubic and most refined linear simulations. Figure 5.8 shows the 21×21 mesh along with the local refinement in the upper right corner (the upper left corner is adapted the same way).

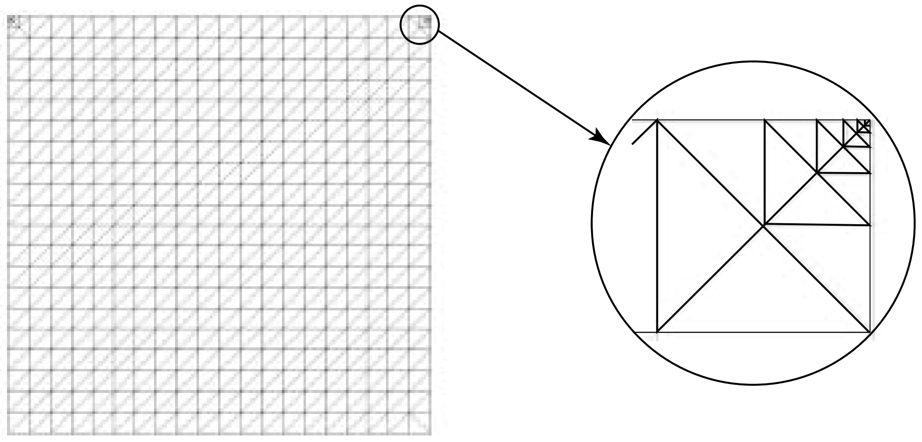


Figure 5.8: Mesh of lid showing corner adaptivity

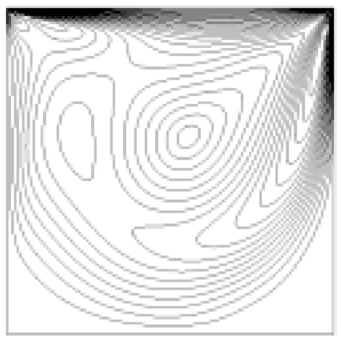
The statistics for these meshes along with the polynomial orders used are shown in Table 5.3. This data does not include the refinement in the upper corners, as this represents a small percentage of the total mesh.

Mesh	Vertices	k
A	11×11	3
B	21×21	2,3
C	41×41	1,2,3
D	81×81	1,2
E	161×161	1

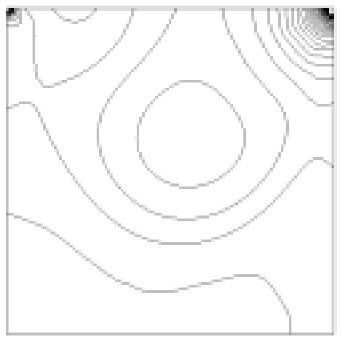
Table 5.3: Lid-driven cavity mesh statistics

The basic solution characteristics of this flow are shown in Figures 5.9(a) - 5.9(d) which display contours of fluid speed, pressure, and vorticity, as well as velocity vectors, respectively. The plots shown here are the quadratic simulation on mesh C, however, all converged simulations look identical. Figure 5.10(a) shows profiles of $u_2(x_1, x_2 = 0)$ and $u_1(x_1 = 0, x_2)$ for the the most refined mesh for each polynomial order. Note that u_1 was scaled by 0.5 to facilitate plotting. Also shown is the benchmark result of Ghia, *et al.*[26] (one stray point was removed from their tabular data). The three plots are virtually indistinguishable.

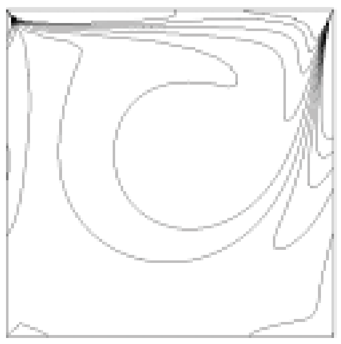
A cost comparison study similar to that for the backward-facing step was carried out for the lid-driven cavity flow (see the velocity compared in Figure 5.10(b)). The simulations, along with their cost index, matrix storage, and mesh size are shown in Table 5.4. In this case, the linear simulation is approximately four times more costly than the cubic and about twice as costly as the quadratic. Here we have also provided information comparing the memory requirements and disk storage required for the simulations. The "Matrix storage" column of Table 5.4 indicates the number of nonzero blocks for the sparse storage of the tangent matrix (the dominant memory requirement), indicating that the memory requirements for the cubic simulation are about one third of the linear, while the quadratic is slightly better than the linear. The "Mesh size" column compares the size in mega-bytes of the files that store both the compact data structure as well as the complete mesh database (on the left and right of the /, respectively). This data indicates that there is also a significant size advantage for the compact data structure for linear elements, however, for cubic elements, the full mesh database is of comparable size (Beall and Shephard [4] found similar results). The final column indicates the CPU time in sec-



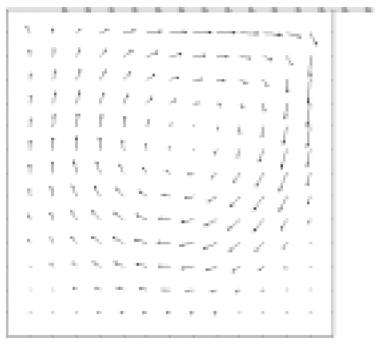
(a) contours of fluid speed



(b) contours of pressure



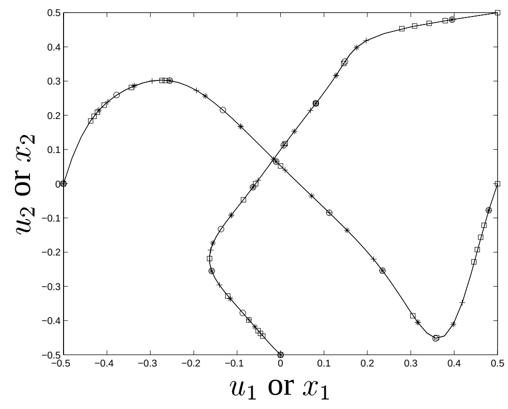
(c) contours of vorticity



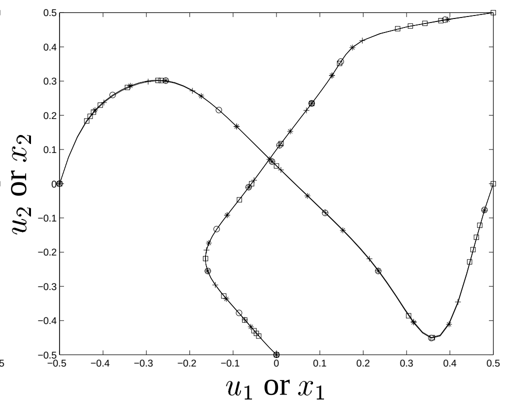
(d) velocity vectors

Figure 5.9: Lid-driven cavity flow characteristics: Mesh D, $k = 2$

onds to run this simulation for a total of 10 time steps. These timings indicate that our cost index, C_I, provides a good measure of the actual computing time. In fact, considering only CPU time, the cubic comes out almost 7 times faster than the linear, while the quadratic is about twice as fast. Some of this may be due to the fact that the linear algebraic system is much tougher to solve (using more than four times as many Krylov vectors per solve) for the linears, due to the extremely fine meshes. In an attempt to level the playing field, the linear and quadratic simulations were limited to 50 Krylov vectors



(a) Comparison on finest meshes. $\circ : k = 1$, $* : k = 2$, $+ : k = 3$, and $\square$: Ghia *et al.*



(b) Qualitatively similar solutions: linear on mesh E, quadratic on mesh C and cubic on mesh A. (see Table 5.4 for symbols), $\square$: Ghia *et al.*

Figure 5.10: Lid-driven cavity flow. Plots of $u_1(x_1 = 0, x_2)$ and $u_2(x_1, x_2 = 0)$

per linear solve, and the convergence was not affected significantly for the 10 time steps (the cubic had no problem reaching the desired linear solver tolerance of 0.1 within the allotted 50 vectors). These timings are with the 50 Krylov vector limitation for the linear and quadratic.

Mesh	k	$C_I(\times 10^3)$	Matrix Storage	Mesh size (MB)	CPU time (sec)	
A	3	316.7	202,804	0.13/0.1	43.2	$+$
C	2	701.6	458,196	0.78/1.4	152.5	$\circ$
E	1	1,383.3	570,193	5.4/24.6	300.7	$*$

Table 5.4: Lid-driven cavity cost comparison

5.4 Vortex shedding behind a square cylinder

The simulation of the flow around a square cylinder at a Reynolds number of 100 (based on the cylinder edge length) is presented as an application of the hierarchical basis to a laminar, time-dependent flow. This flow will also be used to illustrate the techniques of mesh adaptivity based on statistical error indicators for different uniform polynomial orders. Detailed studies (both numerical and experimental) of this flow have been carried out by Davis and Moore [17], and more recently by Sohankar *et al.* [70].

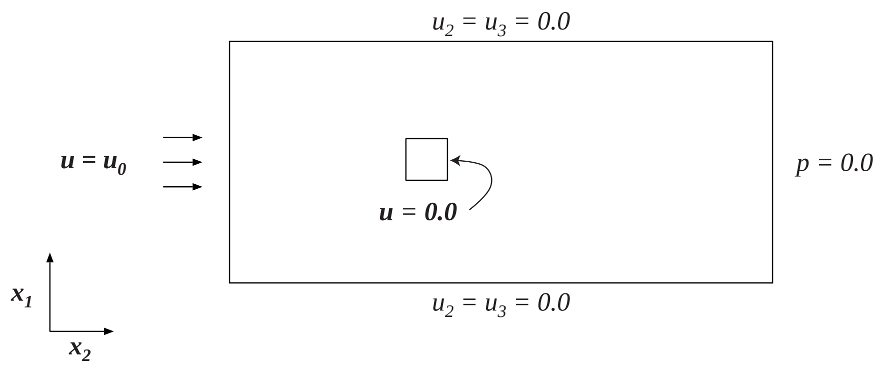
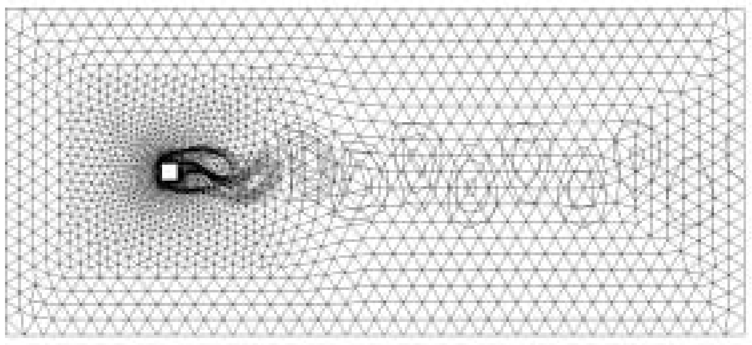


Figure 5.11: Boundary conditions for flow around a square cylinder

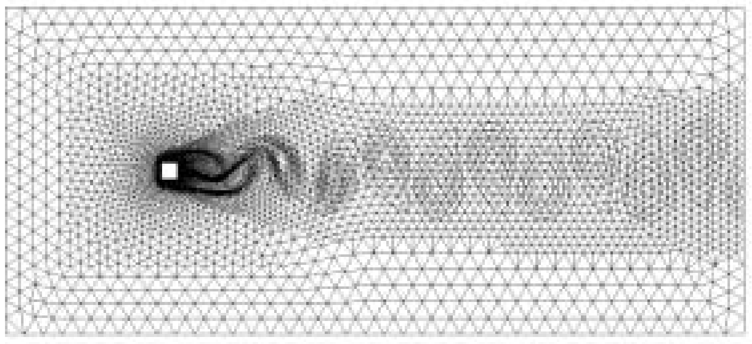
The geometry and boundary conditions for this flow problem are shown in Figure 5.11. The cylinder is centered at the origin, and has an edge length of 1. The distance from the origin to the inflow is 10 units, and it is 25 units from the origin to the outflow of the domain. The lateral walls are placed at $x_2 = \pm 10$. In addition to the boundary conditions shown in Figure 5.11, we have set the tangential traction and normal velocity to zero on the x_3-planes to simulate the two-dimensional flow. The key feature of this flow (at $Re = 100$) is the development of a time-periodic vortex street in the wake of the cylinder, similar to the more common flow about a circular cylinder. This flow, however, is considered a more difficult simulation due to the corners of the box which lead to singularities in the flow field (see Gresho [28]).

The simulation is started from an initial flow field of a uniform velocity of $u_i = u_\infty \delta_{i1}$ and advanced in time using the generalized-α method time integrator introduced in Section 3.3 with the high frequency damping parameter (ρ_∞) set to 0.5. No detailed study of the temporal accuracy *vs.* ρ_∞ was carried out for this flow, since the main goal of this simulation was to perform the h-adaptive calculations and compare the different polynomial orders. The time step was set to 0.1, and 2 Newton iterations per time step were performed.

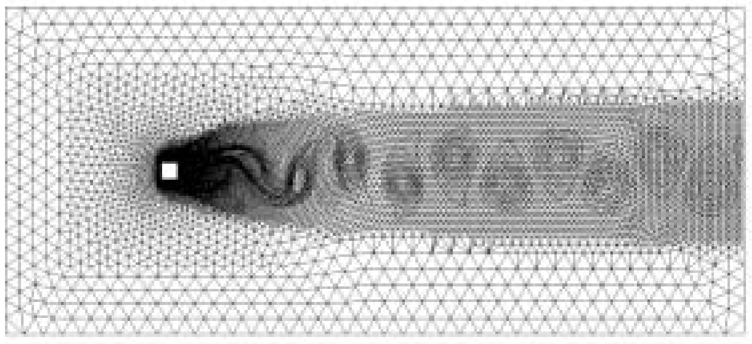
Figures 5.12 through 5.14 show the results of the simulation, along with the h-adaptive meshes. The flow quantity shown in each of the plots is a snapshot in time of the flow vorticity at values of $\pm 0.1 \ldots \pm 1.0$ (at an increment of 0.1), $\pm 1.0 \ldots \pm 7.5$ (at an increment of 0.5), ± 10, and ± 15. In addition to the vorticity, the mesh is shown for each of the simulations. For $k = 1 \ldots 3$, an initial coarse mesh was created and adaptively



Initial Mesh

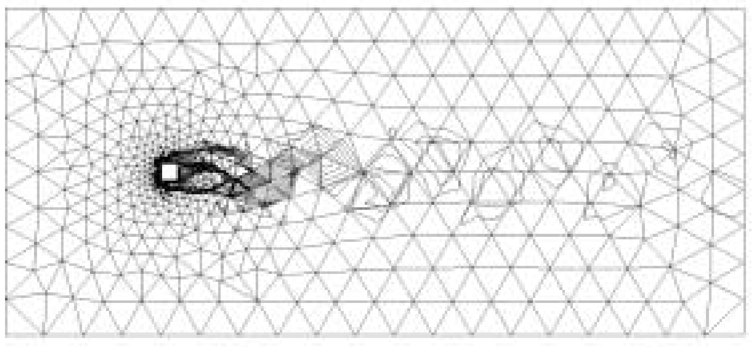


First refinement

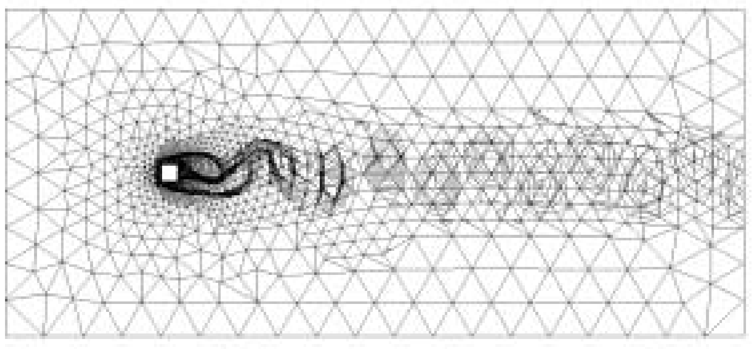


Second refinement

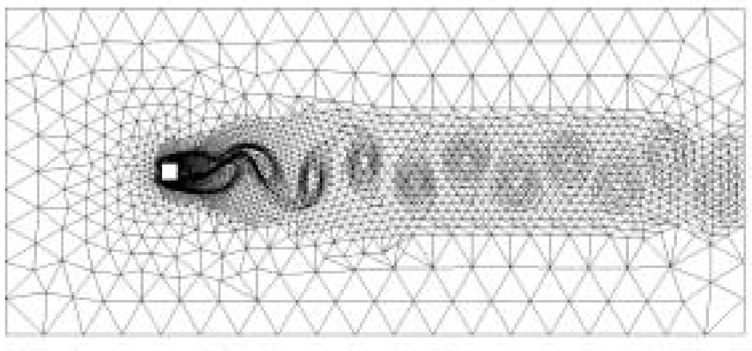
Figure 5.12: Vortex shedding behind a square cylinder: $k = 1$



Initial Mesh

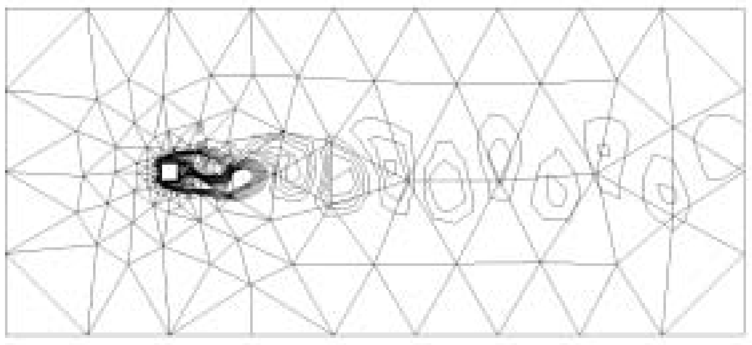


First refinement

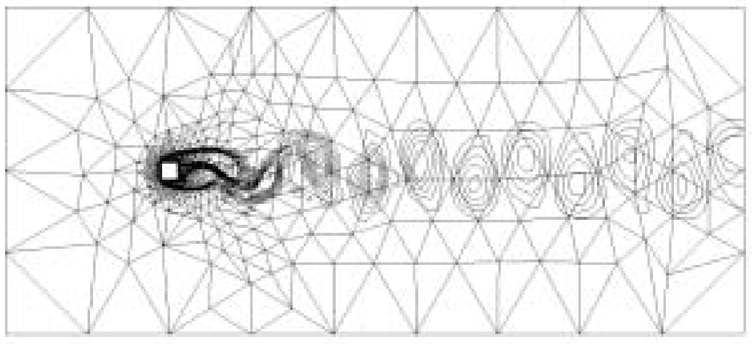


Second refinement

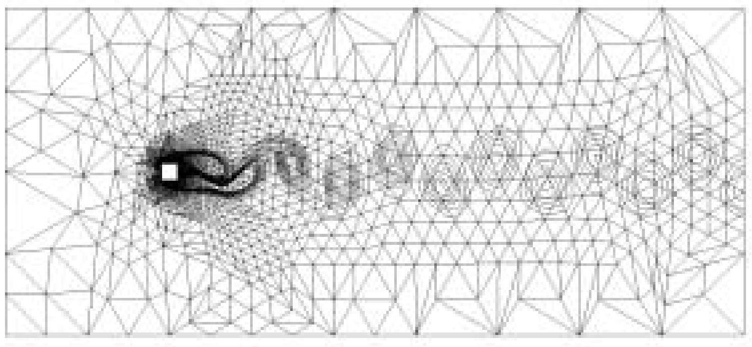
Figure 5.13: Vortex shedding behind a square cylinder: $k = 2$



Initial Mesh



First refinement



Second refinement

Figure 5.14: Vortex shedding behind a square cylinder: $k = 3$

refined based on the error indicators presented in Section 4.6. The data for the series of meshes is presented in Table 5.5. This data includes the mesh information for the entire 3 dimensional mesh, causing the number of degrees of freedom to grow much more rapidly for the higher-order simulations. Although there are only 2 vertices in the x_3-direction, there are additional faces and edges that all get higher-order degrees of freedom (but are effectively wasted).

k	Refinement level	Vertices	n_{sh}	Strouhal #
1	Initial	3658	3658	0.139
1	First	7442	7442	0.145
1	Second	21678	21678	0.146
2	Initial	952	5484	0.127
2	First	1988	11643	0.145
2	Second	4748	28161	0.147
3	Initial	346	5676	0.141
3	First	1032	17760	0.146
3	Second	3780	66600	0.147

Table 5.5: Vortex shedding from square simulation data

The drag and lift profiles are shown in Figures 5.15 through 5.17 for the $k = 1 \ldots 3$ simulations as functions of the non-dimensional time, once the flow has fully developed into its limit cycle. Shown here are profiles on the most refined mesh for each of these polynomial orders. The convergence to the Strouhal number ($St = f_S d/u_0$ where f_S is the shedding frequency and d is the edge length of the square) for each of the simulations is also presented in Table 5.5. These values are in excellent agreement with that presented in the work of Sohankar *et al.* [70] of 0.146, actually our values seem to be converging to 0.147. It can also be observed from the drag profiles that there is a high-frequency oscillation present in the linear solution, and to a lesser extent in the quadratic. This is possibly due to the extremely fine mesh in the near cylinder region for the linear mesh, requiring more corrector passes to sufficiently resolve the flow at each time step. Experience has shown that insufficient nonlinear convergence will lead to such high-frequency oscillations. An alternative approach would be to damp out these oscillations by decreasing the value of ρ_∞, however this would also lead to a slight decrease in accuracy.

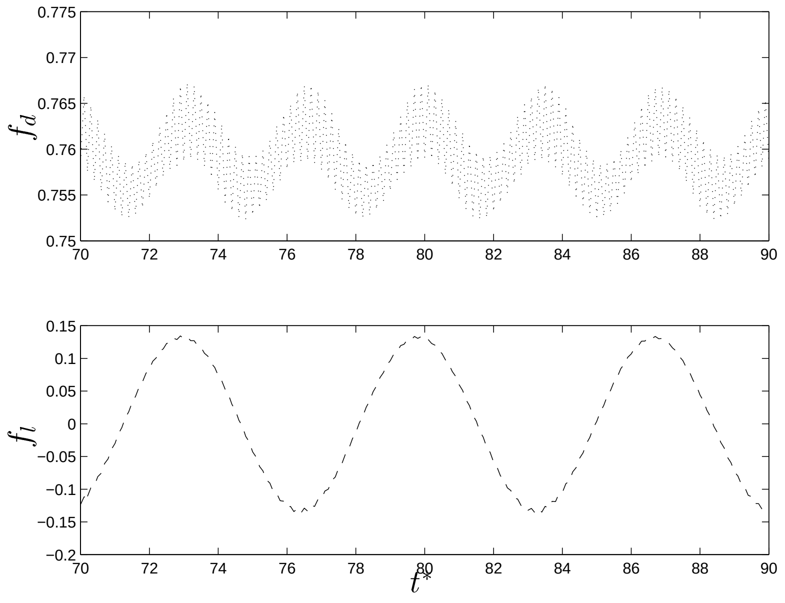


Figure 5.15: Lift and drag profiles for $k = 1$

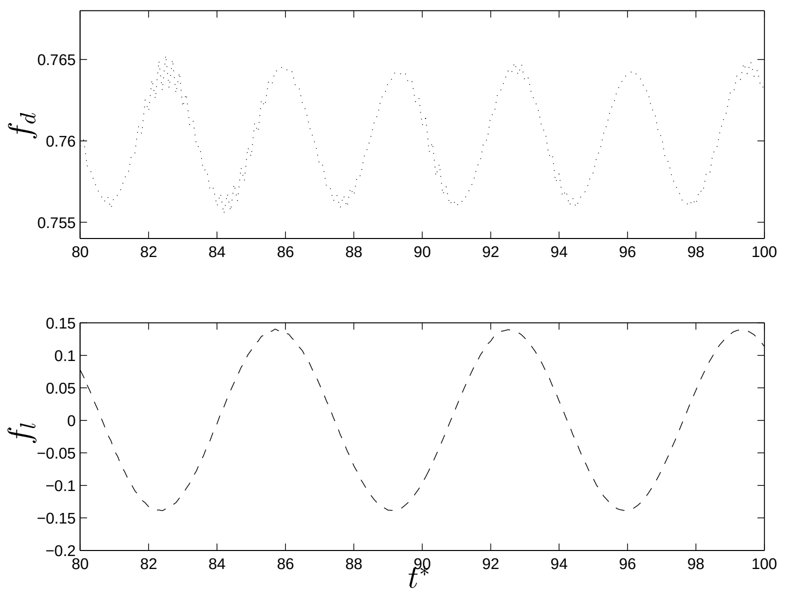


Figure 5.16: Lift and drag profiles for $k = 2$

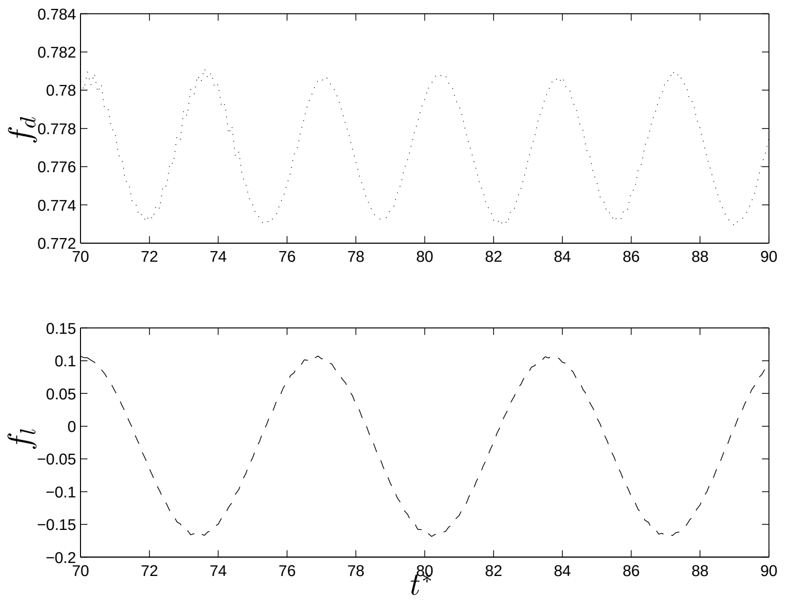


Figure 5.17: Lift and drag profiles for $k = 3$

5.5 Chapter summary

The examples presented in this chapter serve to demonstrate the use of the stabilized finite element formulation with hierarchical basis functions for both steady and unsteady laminar flows. The near optimal convergence rate of the formulation was shown for a problem with a closed form analytical solution, and the cost effectiveness of the higher-order basis was demonstrated for the backward-facing step and the lid-driven cavity. This cost effectiveness has shown the cubic basis to be over seven times cheaper than the linear basis. Also presented in this chapter were simulations of an unsteady problem using statistics based error indicators for creating h-refined meshes.

CHAPTER 6
TURBULENCE COMPUTATIONS

The purpose of this chapter is to provide an introduction to using hierarchical basis finite elements to compute and analyze turbulent flows. The study of turbulent flows typically involves the computation and analysis of statistical quantities derived from the primitive flow variables. Many techniques for analyzing turbulent flow statistics, which work well for linear elements, break down and must be modified when used with the hierarchical basis. Techniques will be described in this chapter to analyze higher-order turbulence simulations which are based on hierarchical basis formulations. Before describing these techniques, a brief introduction to turbulence simulations will be given. The study of turbulence using analytical, experimental, as well as numerical techniques is extremely involved and the reader unfamiliar with the subject should consult a basic text such as Tennekes and Lumley [72], which provides a good introduction to the field.

From a numerical analysis perspective, there are three distinct ways in which turbulent flows can be studied, which essentially relate to how much of the turbulent motion is intended to be resolved, and how much will be modeled. Reynolds averaged Navier-Stokes, or RANS, simulations attempt to resolve only the average flow quantities, leaving all turbulent fluctuations to be modeled. This approach is the most computationally efficient and has proven successful for some steady flows, but has been quite disappointing for unsteady and/or complex flows. RANS simulations have nonetheless become the “workhorse” for industrial type problems (see Wilcox [77] for additional detail on RANS calculations). Large eddy simulation, or LES, attempts to resolve the large eddies in the flow (as the name implies), where most of the energy is known to exist, leaving the subgrid-scale eddies to be modeled (see Jansen [44]). Since it is generally accepted that the smallest eddies, where motion is converted to heat via viscous dissipation, behave similarly for a wide range of flows, it is hoped that modeling only these smallest-scale structures will provide more accurate simulations for a wider variety of flows. However, since more of the flow structures are being resolved, as well as the temporal behavior of the flow, these simulations are necessarily more costly to perform than RANS simulations.

The most complete, and also the most costly, of the methods is direct numerical simulation (DNS), where an attempt is made to resolve all scales of the turbulent flow (both spatial and temporal). While DNS has proven fruitful for studying the basic physics of turbulence (see Kim *et al.* [55] and Le *et al.* [57]), its application to flows of engineering interest is expected to remain out of reach for at least the next several decades.

Simulations of turbulence based on the RANS approach can be analyzed much the same way as laminar, steady flows, allowing the straightforward use of the hierarchical basis function post-processing techniques discussed above. There are, however, expected to be other complicating issues related to using higher-order methods to solve these complicated systems of equations that are beyond the scope of this work. To carefully study LES and DNS computations of turbulence, more work must be done to collect time-averaged statistical quantities which can be compared with theoretical and experimental results. This is in addition to the traditional types of visualization techniques that can be applied to the instantaneous flow fields, but these only provide the qualitative behavior of the flow at a given instant in time, and are difficult or impossible to compare directly with experiments.

6.1 Basic relationships for turbulence simulations

Consider the flow quantities to be decomposed into time-averaged and fluctuating components as (see Tennekes and Lumley [72]),

$$u_i = \bar{u}_i + u_i' \tag{6.1}$$

$$p = \bar{p} + p' \tag{6.2}$$

where the barred quantities are the time-averaged, or mean flow properties, such that,

$$\bar{u}_i = \lim_{T\to\infty} \frac{1}{T} \int_{t_0}^{t_0+T} u_i \;\, dt \tag{6.3}$$

where T is the time over which the average is performed. In computations, this quantity is commonly computed by performing a numerical average over N discrete time steps, i.e.

$$\bar{u}_i = \frac{1}{N}\sum_{k=1}^{N} u_i(t_k) \tag{6.4}$$

where $u_i(t_k)$ is the flow quantity of interest at the k^{th} time step. To gain insight into the dynamics of turbulent flow, various quantities based on the decomposition in (6.1) are collected during a simulation, and post-processed, to generate a statistical "picture" of the turbulent flow-field. In addition to this temporal averaging, spatial averaging over homogeneous directions is also used (for flows that have such character) to increase the statistical sampling, allowing fewer time steps to achieve statistically steady behavior. Several quantities are of particular interest, starting with the mean quantities themselves and properties derived from the mean quantities such as wall forces, followed by the turbulence intensities such as the root-mean-square velocity,

$$u_i^{rms} = \sqrt{\overline{u'_{(i)}u'_{(i)}}} \tag{6.5}$$

(no sum on i) and pressure. Also of interest is the total shear stress,

$$-\overline{u'_i u'_j} + \tau_{ij} \tag{6.6}$$

which is composed of the Reynolds stress (from the fluctuating velocity components) and the viscous stress (from the mean flow quantities). The total shear stress for incompressible turbulent channel flow can be shown analytically to be linear across the channel, and this characteristic is used to indicate that a flow is indeed statistically steady. Finally, the superscript $+$ indicates a non-dimensional quantity scaled by the wall variables; e.g. $x_1^+ = x_1 u_\tau/\nu$, and the friction velocity is defined such that $u_\tau^2 = (\tau_w/\rho)$ where τ_w is the shear stress at the wall (see Tennekes and Lumley [72]).

6.2 Computation of turbulence statistics

As pointed out above, to gain insight into the dynamics of a turbulent flow, statistical quantities are typically used which are generated by collecting several quantities during the flow simulation. Collecting time-averaged statistics is not as simple as it might seem, however. It turns out that the direct method of simply collecting running sums of the quantities, does not always achieve the best results, since the nodal velocity and stress fields that are collected are not necessarily conservative. A method will be presented to reproduce conservative statistics for linear elements, in addition to the standard method which will be used for hierarchical simulations.

To compute the statistics of the turbulent flow from the numerical simulation, the following quantities are collected during the simulation:

$$\bar{u}_i \text{ and } \bar{p} \quad \text{mean flow properties} \tag{6.7}$$

$$\overline{u_i u_j} \quad \text{first-order velocity correlations} \tag{6.8}$$

$$\overline{p^2} \quad \text{first-order pressure correlations} \tag{6.9}$$

$$\overline{\sigma_{ij}} \quad \text{total stress (including pressure)} \tag{6.10}$$

In the flow solver, after the completion of each time step, the quantities in (6.7)-(6.10) (without the bars) are added to running sums, and then divided by the number of time-steps during post-processing to form the time-averages (barred quantities). For example, the Reynolds stress and r.m.s. velocity may be recovered from $\overline{u_i u_j}$ and $\bar{u}_i$ by computing

$$\overline{u'_i u'_j} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j \tag{6.11}$$

which is easily derived from (6.1). The numerical evaluation of (6.7) through (6.10), however, is quite different for linear and higher-order basis simulations.

Some flows, in particular the channel flow considered here, also exhibit spatial directions in which the turbulence is known to be homogeneous. This property allows for additional averaging in these directions, which serves to increase the statistical sample for the averaged quantities, and decreases the number of time steps necessary to achieve a statistically stationary flow. For example, the channel flow problem presented below is averaged over the span-wise and stream-wise directions, in addition to time. This is accomplished as a post-processing operation. Other flows also exhibit homogeneous directions, e.g. a flat plate boundary layer has a single homogeneous direction (span-wise).

6.2.1 Collecting turbulence statistics

The main difficulty in collecting statistics for hierarchical basis coefficients is that the basis coefficients do not directly correspond to solution values at specific spatial locations. Therefore, simply collecting and averaging the coefficients will work for the mean flow properties, but will fail for the second (and higher) order statistics. To correct for this problem, the statistical quantities given above are evaluated and collected, instead, at the element interpolation points. The first step in post-processing is then to use the interpolation algorithm described in Section 4.4 to recover the correct time-averaged basis coefficients for the collected quantities. For example, the velocity correlation coefficients are found such that

$$\overline{u_i u_j} = \sum_{a=1}^{n_{es}} c_{ij}^a N_a \tag{6.12}$$

where c_{ij}^a represent the basis coefficients of the time-averaged field, and N_a are the usual basis functions. When using the linear basis, we can simply collect the statistics at the vertices, however a more accurate approach will be described below for use with linear elements.

After solving for the c_{ij}^a, spatial averaging over homogeneous directions must be accomplished. This is done by creating a structured, two-dimensional sampling grid corresponding to the homogeneous plane. The hierarchical solution is then evaluated at each of the points in the structured grid (using the same search algorithm used to make line plots), and the values are collected and averaged to a single point on a line in the non-homogeneous direction.

6.2.2 Conservative statistics using linear elements

Straightforward evaluation of (6.7)-(6.10) may result in non-conservative nodal fields, causing the statistics to be less accurate than is possible. To remedy this problem, we introduce a method to compute nodal statistics that have the conservation properties restored. The procedure is a generalization of that described by Hughes [42] on page 107, which describes a post-processing technique for calculating consistent boundary flux. This method is also used for computing the boundary flux for all simulations presented in this work. The conservative approach to collecting turbulent statistics, however, does not work for hierarchical basis function, due to the fact that techniques are used for inverting the projection matrices that rely on nodal quadrature which breaks down for hierarchical elements. A potential remedy to this problem is to project the hierarchical basis quantities onto a Lagrange basis for the purposes of collecting the statistics, since Lagrange basis coefficients correspond directly to solution values.

The basic idea is to project the conservative fields to the nodes to recover the conservative nodal velocities and stresses. This involves using the element level residuals from the momentum and continuity equation in a special way to recover these conservative quantities. From a conservation analysis, we have, for any vertex (node),

$$\boldsymbol{m}_a = \hat{\boldsymbol{n}}_a \cdot \boldsymbol{u}_a, \qquad \text{for } a = 1 \ldots n_n \tag{6.13}$$

and

$$\sum_{a=1}^{n_n} \boldsymbol{m}_a = 0, \qquad \sum_{a=1}^{n_n} \hat{\boldsymbol{n}}_a = 0 \tag{6.14}$$

where $\boldsymbol{m}_a$ is the mass flux out of element a, $\hat{\boldsymbol{n}}_a$ is the outward "normal" from element a, for the given node, and n_n is the number of elements incident on the node. The normal here is defined as the average of normals from the faces adjacent to the node. Unfortunately, there is no single velocity $\boldsymbol{u}$ which satisfies $\boldsymbol{m}_a = \hat{\boldsymbol{n}}_a \cdot \boldsymbol{u}$. To remedy this situation, we project, i.e., find $\boldsymbol{u}$ such that

$$\min_{\boldsymbol{u} \in R^N} (\hat{\boldsymbol{n}}_a \cdot \boldsymbol{u} - \boldsymbol{m}_a)(\hat{\boldsymbol{n}}_a \cdot \boldsymbol{u} - \boldsymbol{m}_a) \tag{6.15}$$

which yields the matrix problem

$$\boldsymbol{M}\boldsymbol{u} = \boldsymbol{R} \tag{6.16}$$

where

$$\boldsymbol{M} = \hat{\boldsymbol{n}}_a \hat{\boldsymbol{n}}_a^T \tag{6.17}$$

$$\boldsymbol{R} = \hat{\boldsymbol{n}}_a \boldsymbol{m}_a \tag{6.18}$$

and $\boldsymbol{m}_a$ is the residual of the continuity equation associated with the node and element a. If $\boldsymbol{M}$ is rank deficient, due to the location of the bounding elements, we add $(\boldsymbol{u} \cdot \boldsymbol{u}^{(h,k)})^2$ to the $\min$ function in (6.15), hence

$$\boldsymbol{M} \leftarrow \boldsymbol{M} + \sigma \boldsymbol{I} \tag{6.19}$$

$$\boldsymbol{R} \leftarrow \boldsymbol{R} + \sigma \boldsymbol{u}^{(h,k)} \tag{6.20}$$

where $\sigma = 0$ if $\boldsymbol{M}$ is full rank, and $\sigma = \text{tr}(\boldsymbol{M})$ if $\boldsymbol{M}$ is rank deficient. Similar relationships may be derived from the residuals of the momentum equation and used to compute the second order statistical quantities in a conservative manner, however details will not be given here.

For linear elements, span-wise and stream-wise averaging (for the channel flow) is accomplished using the known structure of the mesh (i, j, k structured numbering) to sum the contributions along these directions, then divide by the number of points included in the sum. This collapses the statistical average to a single line in the non-homogeneous direction (x_2 for the channel).

6.3 Turbulent channel flow at $Re_\tau = 180$ (DNS)

This example presents an initial application of the hierarchical basis functions to a direct numerical simulation of turbulent channel flow at a Reynolds number of 180 based on the friction velocity (u_τ) and the channel half-width, δ, and a Reynolds number of 2800 based on a mean bulk velocity of $U_m = 1.0$. This flow was studied for polynomial orders $k = 1 \ldots 3$ on three different meshes which were selected to give similar resolution for each of the three basis orders. It should be emphasized that the results presented in this section only provide the initial applications of the method to turbulent flows, and more research is needed to fully realize the potential of the higher-order methods. The shortcomings of the present simulation will be described below as they are encountered.

Turbulent channel flow has been studied extensively by many researchers using theoretical, experimental, and numerical methods and is typically used as a test problem for new numerical techniques. Direct numerical simulations were carried out by Kim *et al.* [55] who demonstrated (perhaps for the first time) that turbulent flows could be successfully simulated and statistics collected and analyzed. Their study also enabled the collection of turbulence statistics that were not able to be seen in the experimental results alone.

The flow is considered in a channel with dimensions 4π, 2, and $4/3\pi$ in the x_1, x_2, and x_3 directions, respectively. These physical dimensions were used by Kim *et al.* [55] and were shown to be sufficient to ensure that the flow was statistically uncorrelated in the periodic directions. The boundary conditions were set such that the upper and lower walls were no-slip, and the x_1 and x_3 planes were treated as periodic. The flow is driven by a body force which is adjusted dynamically to maintain the mass-flux through the inflow plane at the value appropriate for a Reynolds number of 2800. The meshes used for each polynomial order are described in Table 6.1 where N_{x_i} represents the number of vertices in the i^{th} direction and $\Delta_1 x_2^+$ represents the placement of the first vertex from the wall in the near wall layer in wall coordinates (indicated by the $+$ superscript). The vertex placement was then graded exponentially in the x_2 direction from this initial spacing.

Polynomial order	$N_{x_1} \times N_{x_2} \times N_{x_3}$	$\Delta_1 x_2^+$
1	$33 \times 65 \times 33$	1.0
2	$17 \times 33 \times 17$	2.0
3	$9 \times 17 \times 9$	4.0

Table 6.1: Turbulent channel flow meshes

Simulations were carried out using the above mesh configurations and polynomial orders. Each simulation ($k = 1 \ldots 3$) was run from an initial condition given by Poiseuille flow (parabolic) with random perturbations of about 10% of the maximum centerline velocity at a time step of 0.1. This initial flow was advanced in time for several thousand time steps, until the flow was determined to have reached a statistically steady state as indicated by a linear profile of the total shear stress across the channel (Reynolds stress plus viscous stress, $-\overline{u_1' u_2'} + (1/Re)\partial \bar{u}_1/\partial x_2$). The flows were then advanced for an additional two thousand time steps and turbulent statistics were collected and post-processed as described in Section 6.2. The structured sub-sampling mesh for the quadratic and cubic simulations, respectively, contain $33 \times 33 \times 33$ and $65 \times 17 \times 65$ points. These values were sufficient to produce good results for the r.m.s. quantities, and adding additional points had no visible effect.

For the results shown here, two Newton corrector passes per time step were used for the linear and quadratic simulations while three passes were needed for the cubic to resolve the flow. The qualitative nature of the flow can be seen in Figure 6.1, which shows velocity contours for the instantaneous flow field for the two periodic planes. All of the contour plots show 10 equi-spaced contours between the maximum and minimum values. Also shown in each of these plots is the finite element mesh on the $x_1 - x_3$ plane. From these figures, it is clear that the qualitative nature of the flow is similarly resolved for all of the simulations. More quantitative comparisons must rely on the statistical properties of the computed turbulence.

Figure 6.2 shows plots of the total shear stress across the channel, normalized by the wall shear stress. It is clear from this plot that all of the simulations have reached their statistically steady state, as indicated by a linear total shear stress distribution across the channel. The turbulence intensities are shown in Figure 6.3. Figure 6.3(a) shows the root-mean-square velocity normalized by the friction velocity (u_τ) for $k = 1 \ldots 3$. This plot displays the basic statistical features of the turbulent flow, and they are qualitatively similar to the results presented by Kim *et al.* [55]. Figure 6.3(b) shows the root-mean-square pressure, normalized by u_τ^2. The r.m.s. pressure for the cubic simulation is considerably higher than that observed for the quadratic and linear.

An additional quantity of interest in the turbulent channel flow is the mean flow velocity, which may be compared to theoretical results for the near wall region, and the log layer (see Tennekes and Lumley [72] for more details). Figure 6.4(a) shows plots of the time averaged x_1-velocity profiles in wall coordinates. This plot indicates that the cubic solution is clearly the closest to the theoretical result in both the inner and outer regions, however it is still quite far from the expected profile. These profiles are normalized by the calculated friction velocities and the resulting profiles are relatively far from the expected ones. Figure 6.4(b) presents the same results normalized by the experimental

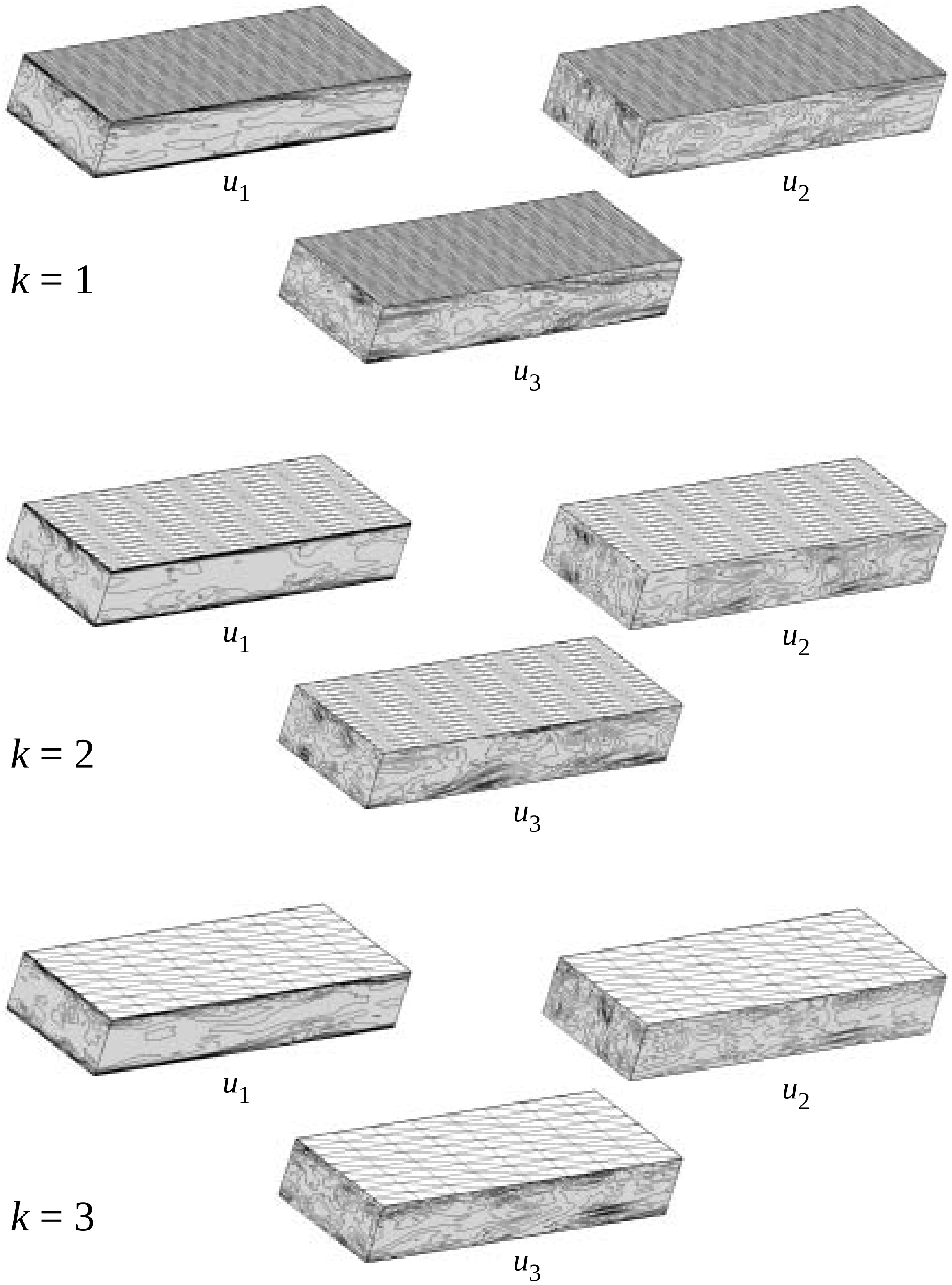


Figure 6.1: Channel flow: instantaneous velocity contours and mesh on $x_1 - x_2$ plane

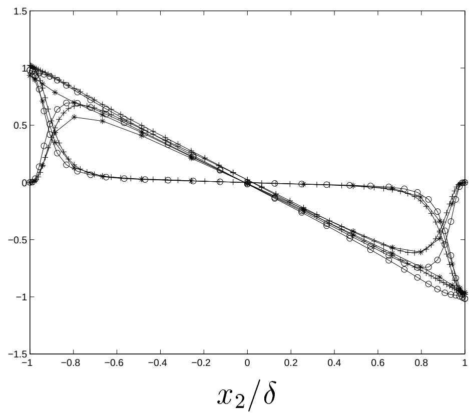
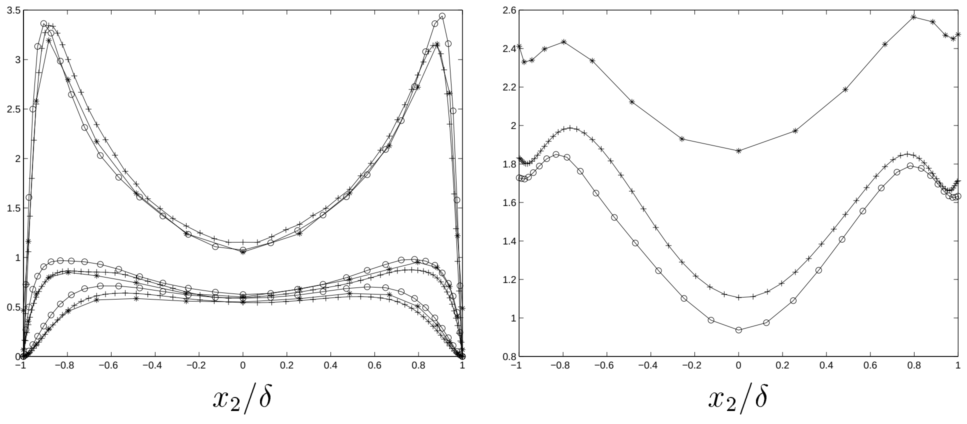


Figure 6.2: Channel flow: Reynolds and viscous stress. $+$: linear, $\circ$: quadratic, and $*$: cubic



(a) Root-mean-square velocity (b) Root-mean-square pressure

Figure 6.3: Channel flow: turbulence intensities. $+$: linear, $\circ$: quadratic, and $*$: cubic

$u_\tau = \sqrt{0.004}$, which are much closer to the matched solution. The discrepancy most likely results from the inability of the method to accurately resolve the near wall structures on the relatively coarse meshes used here (Kim *et al.* used $192 \times 129 \times 160$ grid points in x_1, x_2, and x_3, respectively, using a spectral method). The mean flow velocity can be integrated to yield the bulk mean velocity, defined as

$$U_m = \frac{1}{2}\int_{-1}^{1} \bar{u}_1 d\left(\frac{x_2}{\delta}\right) \tag{6.21}$$

which is shown in Table 6.2, normalized by the calculated u_τ. When normalized by the

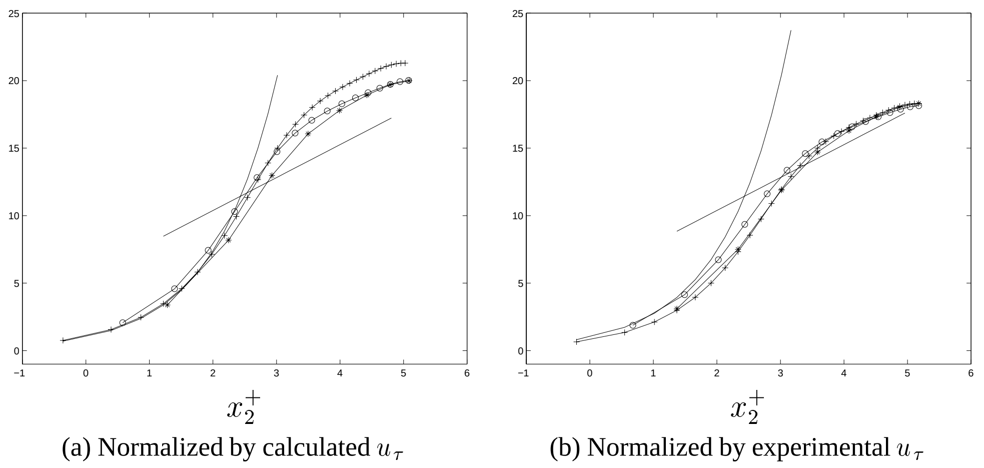


Figure 6.4: Channel flow: wall velocity. $+$: linear, $\circ$: quadratic, and $*$: cubic. Solid line shows theoretical results for wall and log layers

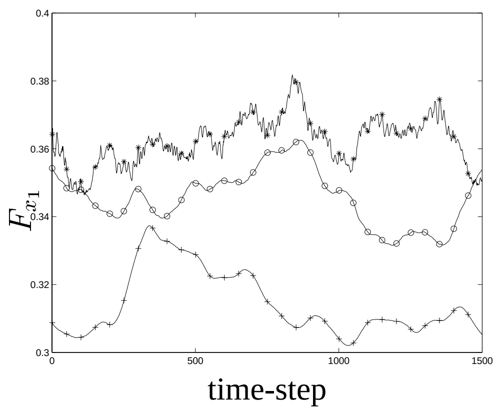


Figure 6.5: Instantaneous total drag force on channel walls. $+$: linear, $\circ$: quadratic, and $*$: cubic. Solid line shows theoretical results for wall and log layers

experimental u_τ, we obtain 15.82, 15.93, and 15.74, which are all in good agreement with the value of 15.63 presented by Kim *et al.* [55], the cubic simulation being best.

Figure 6.5 shows the instantaneous (integrated) drag force on the walls of the channel plotted versus the time step. These profiles can be time averaged to find the average wall shear stress and compared to the expected values. The values obtained for the time averaged coefficient of friction, $C_f = \tau_w/1/2\rho U_m^2$, are included in Table 6.2. The calculations of Kim *et al.* [55] produced a value of $C_f = 8.18 \times 10^{-3}$, giving an error of 17%, 21%, and 28% for the cubic, quadratic, and linear simulations, respectively. It can be

Polynomial order	U_m	C_f
1	18.40	5.91×10^{-3}
2	17.57	6.48×10^{-3}
3	17.18	6.78×10^{-3}

Table 6.2: Mean flow properties

observed from this figure that the cubic simulations appear to have a high-frequency oscillation in the force profiles. This is possibly due to the fact that the consistent calculation of the boundary flux only currently uses the linear modes.

6.4 Chapter summary

The results presented in this chapter represent an initial effort towards studying turbulent flows using hierarchical basis functions, and are not intended to provide a complete study. Before the full potential of the hierarchical basis can be realized, more effort must be dedicated to this area of research. For example the methods of collecting conservative statistics must be generalized for the hierarchical basis, methods of calculating the boundary flux must be more carefully studied, and mesh refinement studies should be carried out. Based on the findings in the present work, the hierarchical basis should be pursued as a means to attain more cost effective simulations of turbulence, as they have been shown to provide for the simpler laminar flows.

CHAPTER 7
DISCUSSION AND CONCLUSIONS

A stabilized finite element method using a hierarchical basis has been applied to the incompressible and compressible Navier-Stokes equations. The implementation is general, allowing three-dimensional simulations on arbitrary, unstructured meshes to be carried out on parallel computers. The stabilized formulation that was introduced has been modified from traditional stabilized formulations to build conservation of momentum back into the discrete solution, in a weak sense, which is typically lacking in formulations based on the advective form of the Navier-Stokes equations. This new formulation has been shown to yield accurate and robust simulations on a variety of problems using both linear and hierarchical basis function alike. Additionally, a new second-order accurate, implicit time integrator has been introduced to advance the semi-discrete weak form in time. This time integrator has been proven to be second-order accurate as well as to have desirable stability properties for a linear model problem. This time integrator has the additional asset of a user controllable amount of numerical dissipation which has been shown to be necessary for damping un-resolvable scales that may appear in a numerical solution.

In addition to the development and implementation of the stabilized formulation and generalized-α method time integrator, significant work has been done to provide a general framework for studying higher-order basis functions for efficient, large-scale finite element simulations of the Navier-Stokes equations. This has been accomplished by basing the pre- and post-processing on a rich, abstract mesh data structure and the use of a compact, streamlined data structure in the analysis code. This enables us to dramatically reduce the memory and computational cost of the rich database, while maintaining its desirable attributes for pre- and post- processing. Techniques have also been developed for parallel processing that enable the communication buffers to be pre-processed and effectively used within the analysis code, and have been shown to yield nearly perfect scalability (98%) for 1 through 8 processors.

The hierarchical basis described in this thesis has been shown to attain near optimal rates of convergence with respect to the interpolation error for both the compressible (channel) and incompressible (Kovasznay) flows, as well as for the linear advection-diffusion equation. These convergence studies are a key step in validating the use of the new stabilized formulations. Not only do they ensure correct implementations, but they also validate the higher-order accuracy property of the formulations. Using the hierarchical basis functions, careful cost *vs.* accuracy studies have been carried out to assess the relative benefits of using higher-order basis functions for stabilized finite element formulations. These studies have been motivated by the desire to simulate more complicated physics problems, where traditional linear elements require too many grid points for current computers. It has been shown here for the first time, through several, relatively simple examples, that for steady problems the cubic basis functions can be up to 7 times more cost effective than linear, while the quadratic basis is up to twice as cost effective. This indicates that the higher-order basis functions may provide a means to attain simulations of physical problems unattainable with linear elements due to computational limitations.

Preliminary application of the methods discussed here to turbulent flows have also been accomplished. A direct numerical simulation (DNS) of turbulent flow in a channel at a Reynolds number of 180 (based on the friction velocity) has been performed and the statistics have been collected and analyzed. New methods to collect and analyze these time averaged turbulence statistics have been developed for use with hierarchical basis functions, which behave differently than Lagrange basis functions. This simulation has been carried out for polynomial orders 1 through 3 on meshes with comparable numbers of degrees of freedom. While the preliminary results look promising, it is still too early to make any conclusive remarks. Currently, the development of new filters is underway to enable large eddy simulation (LES) of turbulent flows making special use of the hierarchical nature of the basis. It is expected that this technology will provide more cost effective LES studies of turbulence as well.

To fully realize the advantage of higher-order basis functions, it is clear that some level of h-refinement will also be necessary to achieve optimal results. Since we are mainly interested in dynamical problems where the solution is changing in time, traditional methods of adaptive h-refinement are not necessarily the most cost effective. New methods of indicating the error based on time-averaged statistical quantities have been developed and tested on a simple problem to demonstrate the ideas. Currently, heuristic reasoning is used in conjunction with the error indicators to adaptively refine the mesh, but it has been shown to provide effective h-adaptivity on the problem of vortex shedding behind a square cylinder which has a stationary wake-type structure when viewed from a statistical standpoint. It is hoped that these methods can be further developed in future work, as well as coupled with more rigorous mathematical analysis.

There are, however, limitations of the present research. It is hoped that they may be addressed in future work, to further enhance the benefit of the hierarchical basis. The present formulation is limited to uniform polynomial order. Although the formulation is not theoretically limited to cubic basis, the current implementation must be slightly generalized if higher than cubic basis functions are desired. At a minimum, a face-based data structure must be added to indicate which shape functions are active on each mesh face. This is not the only problem with higher polynomial order. Efficient symmetric Gauss integration formulas only exist for tetrahedral regions to a high enough order to integrate the cubic basis. To go with k greater than 3 requires the use of inefficient degenerate quadrature rules, which are much more expensive. Some of this cost may be reduced when the method is extended to hexahedral elements, which can use efficient tensor-product integration formulas to any order. The current implementation is also limited to straight-sided elements using a linear mapping to element coordinates.

There are many ways in which the research in the present thesis may be extended and built upon. Although the formulation presented is completely general, the present implementation only allows for tetrahedral meshes and uniform polynomial order. Based on our experience with the excellent properties of hexahedral elements with linear basis functions (for relatively uniform flow configurations which allow hexahedral meshes), the extension to hierarchical hexahedral meshes is likely to provide even better results. This, in conjunction with non-uniform polynomial order will allow meshes for relatively simple configurations such as boundary layers to be more effectively simulated by using a lower polynomial order in the streamwise and span-wise directions while maintaining higher-order for the wall normal direction, where the flow gradients are highest. This type of polynomial order grading is less likely possible for meshes comprised of tetrahedral elements due to the fact that they are often poorly aligned with surfaces.

Theoretical issues related to the present research also need to be addressed in future work. Although the stability of the new (conservation restoring) formulation is relatively straightforward to show, a full convergence proof has yet to be completed. Based on the convergence proofs for closely related stabilized formulations (such as the SUPG formulation without the conservation restoring terms) as well as the numerical demonstration of higher-order accuracy presented here for the Kovasznay flow, it is strongly expected that the new formulation may also be rigorously shown to be higher-order accurate. The development of improved inverse estimates used in the design of the stabilization matrix, τ, are also needed to perhaps further increase the accuracy of the higher-order methods.

Large eddy and direct numerical simulation of turbulent flows present special problems for higher-order basis functions, which have only partially been addressed in the present work. The procedures that are used for computing the dynamic model coefficients, crucial to the performance of LES, are computed in part by averaging over homogeneous directions in the flow. This type of operation is difficult for hierarchical basis functions since the basis coefficients do not directly correspond to solution values at nodes, as they do for the Lagrange basis. Some preliminary methods have been developed to circumvent these problems and provide adequate results, but further research is necessary to most effectively study turbulent flows. This involves extending the consistent calculation of time-averaged statistics to hierarchical basis functions. New filters that take advantage of the hierarchical nature of the basis should also be developed and implemented. This may be achieved, for example, by forming an element based filter from the difference between the cubic and linear solutions.

The design of new stabilized methods that explicitly take advantage of the hierarchical basis should also be explored. These new stabilized methods should build on recent research on multilevel finite element methods and their connection with large eddy simulations of turbulent flows. The hierarchical basis may be used in these new formulations to represent the sub-grid, or “un-resolvable”, scales in the turbulent flow field. This may be accomplished by modeling these “un-resolvable” scales by cubic and higher polynomials and representing the resolvable scales by the linear and quadratic portions of the basis. The hierarchical basis is unique for such applications due to the subset property between successive polynomial orders. These new methods are currently under investigation, and are expected to yield much more accurate results than the current techniques for large eddy simulations.

APPENDIX A
ADVECTION-DIFFUSION EQUATION: TRELLIS IMPLEMENTATION

To implement a new equation within Trellis requires that we derive a new analysis class from the `FEAnalysis` base class (defined in the Trellis library), we call this class `ADAnalysis`, and its definition is as follows:

```
class ADAnalysis : public FEAnalysis {
public:
  ADAnalysis(AttCase *theCase);
  Field<ScalarDof> *scalarField();
  Field<DofVector> *diffusiveFluxField();
  SGModel *getModel() { return gModel; }
  virtual ~ADAnalysis();

protected:
  virtual StiffnessContributor
                           *makeElement(MFace &meshFace);
  virtual DiscreteSystem *system() const = 0;
  virtual void setup();

  virtual void doSource(GFace *gf, AttributeTensorOr0 *at);
  virtual void doFlux(GEdge *ge, AttributeTensorOr1 *at);
  virtual void doPrescribedScalar(GEdge *ge,
                                AttributeTensorOr0 *at);

  Field<ScalarDof> *ScalarField;
  Field<DofVector> *DiffusiveFluxField;
}
```

The member functions defined here carry out necessary tasks for FE analysis that are specific to the advection-diffusion equation. Essential boundary conditions are set up using the function `doPrescribedScalar` and the boundary integral (and natural boundary conditions) are implemented with the function `doFlux`. Finally, the function `doSource` implements the source term (if one is present). This class contains several `public` data members, to return the results of the simulation, and the `private` members `ScalarField` and `DiffusiveFluxField`. The concept of a field is defined in detail in Beall and Shephard [4]. A field consists of a collection of interpolations over the finite element mesh and represents the unknown function we are solving for. `DiffusiveFluxField` is used to store the local reconstruction of the diffusive flux.

The core of the work, however, is accomplished by the `makeElement` member function which is responsible for creating the appropriate system contributors. Its definition is given as follows:

```
StiffnessContributor *ADAnalysis::makeElement(MFace
                                              &meshFace)
{
  Interpolation2d<ScalarDof> *interp;

  // create a new interpolation for this
  // mesh face (element)
  interp = ScalarField->createInterpolation(&meshFace);

  // create (then return) a new stiffness contributor for
  // this element
  StiffnessContributor  *e = new ADSC2d(interp);
  return e;
}
```

There are two key lines here, first,

```
  interp = ScalarField->createInterpolation(&meshFace);
```

which creates a new interpolation within the field `ScalarField`, and second,

```
  StiffnessContributor  *e = new ADSC2d(interp);
```

which defines a new `StiffnessContributor` for representing the element level weak form. This stiffness contributor for the advection-diffusion equation is known as `ADSC2d` (the 2-D version) and its class definition takes the form:

```
class ADSC2d : public Stiffness2d {
public:
  ADSC2d(Interpolation2d<ScalarDof> *interp) ;

  // residual contributor
  void r(VectorAssembler *a, int order);

  // tangent contributor
  void du0(MatrixAssembler *a);

  // time contributor
  void du1(MatrixAssembler *a);

  // compute tangent, residual, and mass at an
  // integration point
  ForceVector residual(const SPoint2 &pt);
  ElementMatrix tangent(const SPoint2 &pt);
  ElementMatrix mass(const SPoint2 &pt);

protected:
  Interpolation2d<ScalarDof> *Interp;

private:
  AttributeTensorOr0 *velocity[2];
  double diffusivity;
  Stabilization tau;
};
```

These class members define all the computations that are necessary for the local computation of the weak form. The first three functions `r, du0`, and `du1` are responsible for setting up the integration rule for element quadrature for the residual, tangent, and mass terms, respectively. Then, the integrator (defined in the Trellis library) calls, respectively, `residual`, `tangent`, and `mass` to evaluate the integrand.

APPENDIX B
VISUALIZATION OF HIERARCHICAL SOLUTION DATA

The tools available in the SCOREC mesh database (see Beall [2]) greatly simplify the process of creating this “visualization” mesh. Operations such as looping over all mesh entities classified on a given model entity, and attaching data to mesh entities are used. The entire procedure is outlined in the following algorithm (written as pseudo-C++ code), given a single model face to be visualized with `nVis` new nodes on each original mesh edge,

```
// get the template for a triangle
TriMesh tri(nVis);

// loop over classified mesh faces
EDListIter<MFace> fIter  = gface->firstMeshFace(mesh);
MFace *face;
while (face = fIter.next()){

  // vertex modes (3 vertices per triangle)
  for (j = 0; j < 3; j++) {
    MVertex *vertex = face->vertex(j);
    double *q = solution( vertex );
    add ( q );
  }

  // edge modes (3 edges per triangle)
  for (k = 0; k < 3; k++) {
    MEdge *edge = face->edge(j);

    // add new nodes along this edge
    for (j = 0; j < tri.num_per_edge(); j++) {
      double *q = solution( face, tri.xi(i) );
      add ( q );
    }
  }

  // add new nodes on the face
  for (j = 0; j < tri.num_per_face(); j++) {
    double *q = solution( face, tri.xi(i) );
    add ( q );
  }
}
```

This algorithm illustrates the key concepts in the implementation of the local refinement. `TriMesh` is a C++ class that provides the uniform mesh of a triangle (nodal coordinates and element connectivity), which is used for each of the mesh faces, and is shown in Figure B.1 for the case `nVis`$= 3$. The function “solution” returns the hierar-

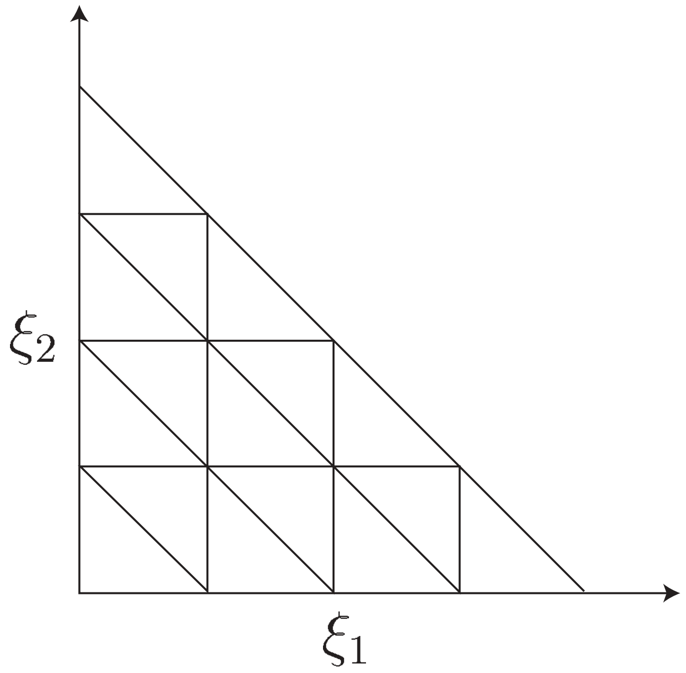


Figure B.1: Trianglular face mesh template

chical solution evaluated at the desired local coordinate for the given mesh entity, and “add” adds the new solution values to our global visualization data structure. The number of new nodes that are added to each mesh edge and face depend on the user-entered parameter `nVis`, and are returned by the `TriMesh` member functions `num_per_edge` and `num_per_face`, respectively.

The new global mesh is output to the visualization package, and consists of nodal coordinates and element connectivity for the new mesh. Note that the “add” function must determine whether or not a node has already been added, so as not to add it multiple times, an operation that is conveniently carried out with the mesh database tools to attach data to an existing mesh entity. Before new data is added, the entity is simply checked to see if the data is already there.

REFERENCES

[1] F. Bastin. Jet noise using large eddy simulation. In *Annual Research Briefs*, pages 115–132, NASA Ames / Stanford University, 1996. Center for Turbulence Research.

[2] M. W. Beall. *SCOREC Mesh Database User Guide: Version 2.4*.

[3] M. W. Beall. *An object-oriented framework for the reliable automated solution of problems in mathematical physics*. PhD thesis, Rensselaer Polytechnic Institute, 1999.

[4] M. W. Beall and M. S. Shephard. A general topology-based mesh data structure. *Int. J. Numer. Meth. Engng.*, 40(9):1573–1596, 1997.

[5] M. W. Beall and M. S. Shephard. An object-oriented framework for reliable numerical simulations. *Engineering with computers*, 15(1):61–72, 1999.

[6] M. Behr, D. Hastreiter, S. Mittal, and T. E. Tezduyar. Incompressible flow past a circular cylinder: dependence of the computed flow field on the location of the lateral boundaries. *Comp. Meth. Appl. Mech. Engng.*, 123:309–316, 1996.

[7] K. S. Bey, A. Patra, and J. T. Oden. hp-version discontinuous galerkin methods for hyperbolic conservation laws: A parallel adaptive strategy. *Int. J. Numer. Meth. Engng.*, 1995.

[8] R. Biswas, K. D. Devine, and J. E. Flaherty. Parallel adaptive finite element methods for conservation laws. *Appl. Numer. Maths.*, 14:255–284, 1994.

[9] D. L. Bonhaus. *A higher order finite element method for viscous compressible flows*. PhD thesis, Virginia Polytechnic Institute and State University, Nov. 1998.

[10] F. Brezzi, L. P. Franca, T. J. R. Hughes, and A. Russo. $b = \int g$. *Comp. Meth. Appl. Mech. Engng.*, 145:329–339, 1997.

[11] A. N. Brooks and T. J. R. Hughes. Streamline upwind / Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 32:199–259, 1982.

[12] P. Carnevali, R. B. Morris, Y. Tsuji, and G. Taylor. New basis functions and computational procedures for p-version finite element analysis. *Int. J. Numer. Meth. Engng.*, 36:3759–3779, 1993.

[13] M. H. Carpenter and David Gottlieb. Spectral methods on arbitrary grids. *Journal of Computational Physics*, 1996.

[14] Qi Chen and Ivo Babuška. The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. *Comp. Meth. Appl. Mech. Engng.*, 137:89–94, 1996.

[15] J. Chung and G. M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. *Journal of Applied Mechanics*, 60:371–75, 1993.

[16] P. G. Ciarlet. *The finite element method for elliptic problems*. North-Holland, Amsterdam, 1978.

[17] R. W. Davis and E. F. Moore. A numerical study of vortex shedding from rectangles. *Journal of Fluid Mechanics*, 1982.

[18] H. L. de Cougny and M. S. Shephard. Parallel mesh adaptation by local mesh modification. Technical Report SCOREC #21-1995, Scientific Computation Research Center, RPI, Troy, NY, 1995.

[19] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward a universal h-p adaptive finite element strategy, part 1: Constrained approximation and data structure. *Comp. Meth. Appl. Mech. Engng.*, 77:79–112, 1989.

[20] Karen M. D. Devine. *An adaptive hp-finite element method with dynamic load balancing for the solution of hyperbolic conservation laws on massively parallel computers*. Ph.D. Thesis, RPI, 1994.

[21] S. Dey. *Geometry-based three-dimensional hp-finite element modelling and computations*. PhD thesis, Rensselaer Polytechnic Institute, 1997.

[22] L. P. Franca and S. Frey. Stabilized finite element methods: II. The incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 99:209–233, 1992.

[23] L. P. Franca, S. Frey, and Thomas J. R. Hughes. Stabilized finite element methods: I. Application to the advective-diffusive model. *Comp. Meth. Appl. Mech. Engng.*, 95:253–276, 1992.

[24] D. K. Gartling. A test problem for outflow boundary conditions – flow over a backward-facing step. *International Journal of Numerical Methods in Fluids*, 11:953–967, 1990.

[25] C.W. Gear. *Numerical initial value problems in ordinary differential equations*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[26] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–441, 1982.

[27] P. M. Gresho. Some current CFD issues relevant to the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 87:201–252, 1991.

[28] P. M. Gresho and S. T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 2: implementation. *International Journal of Numerical Methods in Fluids*, 11:621–659, 1990.

[29] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*. Wiley, New York, NY, 1998.

[30] P. Hansbo and A. Szepessy. Velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 1990.

[31] Isaac Harari and Thomas J. R. Hughes. What are C and h?: Inequalities for the analysis and design of finite element methods. *Comp. Meth. Appl. Mech. Engng.*, 97:157–192, 1992.

[32] G. Hauke. *A unified approach to compressible and incompressible flows and a new entropy-consistent formulation of the k-ϵ model*. PhD thesis, Stanford University, 1995.

[33] G. Hauke and T. J. R. Hughes. A unified approach to compressible and incompressible flows. *Comp. Meth. Appl. Mech. Engng.*, 113:389–396, 1994.

[34] G. Hauke and T. J. R. Hughes. A comparative study of different sets of variables. *Comp. Meth. Appl. Mech. Engng.*, 153:1–44, 1998.

[35] D. Haworth and K. E. Jansen. LES on unstructured deforming meshes: towards reciprocating IC engines. In *Proceedings of the 1996 Summer Program*, pages 329–346, NASA Ames / Stanford University, 1996. Center for Turbulence Research. also accepted for publication in Computers in Fluids.

[36] T. J. R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Comp. Meth. Appl. Mech. Engng.*, 127:387–401, 1995.

[37] T. J. R. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for fluid dynamics: V. A stable petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Comp. Meth. Appl. Mech. Engng.*, 59:85–99, 1986.

[38] T. J. R. Hughes, L. P. Franca, and G. M. Hulbert. A new finite element formulation for fluid dynamics: VIII. The Galerkin / least–squares method for advective–diffusive equations. *Comp. Meth. Appl. Mech. Engng.*, 73:173–189, 1989.

[39] T. J. R. Hughes and K. E. Jansen. A stabilized finite element method for the Reynolds-averaged Navier-Stokes equations. *Surveys on Mathematics for Industry*, 4:279–317, 1995.

[40] T. J. R. Hughes and M. Mallet. A new finite element formulation for fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems. *Comp. Meth. Appl. Mech. Engng.*, 58:305–328, 1986.

[41] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. Large-eddy simulation and the variational multiscale method. *Computing and Visualization in Science*, to appear, 1999.

[42] Thomas J. R. Hughes. *The finite element method: Linear static and dynamic finite element analysis*. Prentice Hall, Englewood Cliffs, NJ, 1987.

[43] K. E. Jansen. Large-eddy simulation of flow around a NACA 4412 airfoil using unstructured grids. In *Annual Research Briefs*, pages 225–232, NASA Ames / Stanford University, 1996. Center for Turbulence Research.

[44] K. E. Jansen. A stabilized finite element method for computing turbulence. *Comp. Meth. Appl. Mech. Engng.*, 174:299–317, 1999.

[45] K. E. Jansen, S. S. Collis, C. H. Whiting, and F. Shakib. A better consistency for low-order stabilized finite element methods. *Comp. Meth. Appl. Mech. Engng.*, 174:153–170, 1999.

[46] K. E. Jansen, C. H. Whiting, and G. M. Hulbert. A generalized-α method for integrating the filtered Navier-Stokes equations with a stabilized finite element method. *Comp. Meth. Appl. Mech. Engng.*, 1999. Contributed to a special volume devoted to the Japan-US Symposium on F.E.M. in Large-Scale C.F.D., SCOREC Report 10-1999.

[47] Z. Johan, T. J. R. Hughes, K. K. Mathur, and S. L. Johnsson. A data parallel finite element method for computational fluid dynamics on the Connection Machine system. *Comp. Meth. Appl. Mech. Engng.*, 99:113, 1992.

[48] Z. Johan, T. J. R. Hughes, and F. Shakib. A globally convergent matrix-free algorithm for implicit time marching schemes arising in finite element analysis. *Comp. Meth. Appl. Mech. Engng.*, 87:281–304, 1991.

[49] A. A. Johnson and T. E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Comp. Meth. Appl. Mech. Engng.*, 119:73–94, 1994.

[50] A. A. Johnson and T. E. Tezduyar. 3D simulation of fluid-particle interactions with the number of particles reaching 100. *Comp. Meth. Appl. Mech. Engng.*, 145:301–321, 1997.

[51] C. Johnson and A. Szepessy. On the convergence of a finite element method for a nonlinear hyperbolic conservation law. *Mathematics of Computation*, 1987.

[52] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, Sweden, 1987.

[53] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report #95-035, U. Minnesota, Dept. of Comp. Sci. and Army HPC Center, 1995.

[54] J. Kennedy, M. Behr, V. Kalro, and T. Tezduyar. Implementation of implicit finite element methods for incompressible flows on the CM-5. *Comp. Meth. Appl. Mech. Engng.*, 1994.

[55] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133, 1987.

[56] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Proc. Cambridge Philos. Soc.*, 44, 1948.

[57] H. Le, P. Moin, and J. Kim. Direct numerical simulation of turbulent flow over a backward-facing step. *Journal of Fluid Mechanics*, 330:349–374, 1997.

[58] J. T. Oden, I. Babuška, and C. E. Baumann. A discontinuous hp finite element method for diffusion problems. *Journal of Computational Physics*, 146:491–519, 1998.

[59] A. Rússo. Bubble stabilization of the finite element methods for the linearized incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 132:335–343, 1996.

[60] Robert Sedgewick. *Algorithms in C*. Addison–Wesley, Reading, Massachusetts, 1990.

[61] F. Shakib. *Finite element analysis of the compressible Euler and Navier-Stokes equations*. PhD thesis, Stanford University, 1989.

[62] F. Shakib and T. J. R. Hughes. A new finite element formulation for computational fluid dynamics: IX. Fourier analysis of space-time Galerkin/least-squares algorithms. *Comp. Meth. Appl. Mech. Engng.*, 87:35–58, 1991.

[63] F. Shakib, T. J. R. Hughes, and Z. Johan. A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 89:141–219, 1991.

[64] Farzin Shakib. http://www.acusim.com.

[65] M. S. Shephard. The specification of physical attribute information for engineering analysis. *Eng. Comput.*, 4:145–155, 1988.

[66] M. S. Shephard, S. Dey, and J. E. Flaherty. A straight forward structure to construct shape functions for variable p-order meshes. *Comp. Meth. Appl. Mech. Engng.*, 147:209–233, 1997.

[67] S. J. Sherwin and G. E. Karniadakis. A new triangular and tetrahedral basis for high-order (hp) finite element methods. *Int. J. Numer. Meth. Engng.*, 38:3775–3802, 1995.

[68] S. J. Sherwin and G. E. Karniadakis. A triangular spectral element method; applications to the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 123:189–229, 1995.

[69] S. J. Sherwin and G. E. Karniadakis. Tetrahedral hp finite elements: algorithms and flow simulations. *Journal of Computational Physics*, 124(1):14, 1996.

[70] A. Sohankar, C. Norberg, and L. Davidson. Low-Reynolds-number flow around a square cylinder at incidence: study of blockage, onset of vortex shedding and outlet boundary condition. *International Journal of Numerical Methods in Fluids*, 1998.

[71] C. A. Taylor, T. J. R. Hughes, and C. K. Zarins. Finite element modeling of blood flow in arteries. *Comp. Meth. Appl. Mech. Engng.*, 158:155–196, 1998.

[72] H. Tennekes and J.L. Lumley. *A First Course in Turbulence*. The MIT Press, Cambridge, MA, 1972.

[73] T. E. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro, and M. Litke. Flow simulation and high performance computing. *Computational Mechanics*, 18:397–412, 1996.

[74] T. E. Tezduyar, M. Behr, and J. Liou. New strategy for finite element computations involving moving boundaries and interfaces. The deforming-spatial-domain/space-time procedure. I. the concept and the preliminary numerical tests. *Comp. Meth. Appl. Mech. Engng.*, 94:339–351, 1992.

[75] T. E. Tezduyar, M. Behr, and J. Liou. New strategy for finite element computations involving moving boundaries and interfaces. The deforming-spatial-domain/space-time procedure. II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Comp. Meth. Appl. Mech. Engng.*, 94:339–351, 1992.

[76] C. H. Whiting, K. E. Jansen, and S. Dey. Hierarchical basis in stabilized finite element methods for compressible flows. *Comp. Meth. Appl. Mech. Engng.*, submitted, 1999. SCOREC Report 11-1999.

[77] David C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, La Canada, CA, 1998.