Nonconforming Sliding Spectral Element Methods for the Unsteady Incompressible Navier-Stokes Equations

by

George Anagnostou

B.S., University of Michigan, Ann Arbor, MI (1985)
M.S., M.I.T., Cambridge, MA (1987)

SUBMITTED TO THE DEPARTMENT OF OCEAN ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Doctor of Philosophy
at the
Massachusetts Institute of Technology

February 1991

©Massachusetts Institute of Technology 1990

Signature of Author

Certified by

Professor Anthony T. Patera
Thesis Supervisor

Accepted by

Professor Douglas A. Carmichael
Chairman, Department Committee on Graduate Studies
Nonconforming Sliding Spectral Element Methods for the Unsteady Incompressible Navier-Stokes Equations

by

GEORGE ANAGNOSTOU

Submitted to the Department of Ocean Engineering, in partial fulfillment of the requirements for the Doctor of Philosophy in Ocean Engineering

Abstract

The aim of this work is to present, in the context of high-order methods, a new class of nonconforming discretizations for the unsteady, incompressible Navier-Stokes equations. In the current approach, the computational domain is broken-up into $K$ macro-elements with no geometric restrictions imposed on adjoining elements for coincidence of edges and corners (nonconformity). Inside each element, dependent and independent variables are approximated by $N^{th}$-order tensor product polynomial expansions, while, at elemental interfaces, a weak matching condition between expansions is required. Variational projection operators and Gauss numerical quadrature are used to generate the set of discrete equations. Convergence to the exact solution is achieved by increasing the degree $N$ of the polynomial approximation.

The decoupling of subdomains in the nonconforming decomposition allows for treatment of sliding grids; the governing equations in the moving region are extended by considering a convection term based on mesh velocity. The fully-discrete equations are solved by a fractional-step method in time, where the decoupled Helmholtz systems are treated by conjugate gradient iteration. Numerical results are presented that demonstrate the optimality of the decomposition. The flexibility and effectiveness of the decomposition are further illustrated by several examples of incompressible Navier-Stokes simulations.

Thesis Supervisor: Professor Anthony T. Patera

Thesis Committee: Professors David C. Gossard and Justin E. Kerwin.
Acknowledgements

My deepest of gratitudes to Prof. Tony Patera for his priceless advice throughout the course of this work and his unending enthusiasm and optimism. Special thanks to Prof. Y. Maday for his unique insight in applied mathematics, and Dr. D. Dewey for the fun and excitement of Geometry-Defining Processors. For productive discussions and suggestions, I wish to thank Drs. E. Ronquist and P. Fischer both of whom I consider my mentors.

Finally, my warmest of feelings to my parents and brother for the best emotional support any family could have ever provided during the course of this work. I couldn’t have done it without you.
3.3 Navier-Stokes Solution Algorithm ........................................ 52

4 Implementation ........................................................................ 57

4.1 Skeleton Composition .............................................................. 58

4.1.1 Detection ........................................................................ 58

4.1.2 Representation ................................................................. 68

4.2 Field Data Structure ............................................................... 71

4.3 Memory Allocation .................................................................. 81

4.4 Inter-element Communication .................................................. 85

5 Examples .................................................................................. 94

5.1 Mixing Processes ................................................................. 94

5.2 Rotor/Stator Interactions ....................................................... 108

5.3 Future Work ......................................................................... 114

A NACA four-digit series of wing sections ..................................... 118
Chapter 1

Introduction

Computational Fluid Dynamics in recent years has become a reliable alternative mean of analysis for many problems of fundamental and practical interest in fluid dynamics. Yet, because of the inherent difficulties associated with the governing equations, the analysis of many commonplace engineering problems has proven an elusive goal. Continued emphasis must be placed on all aspects contributing to Computational Fluid Dynamics research such as geometry input and mesh generation, discretization, subsequent solution of the discrete equations, programming models, parallel computer architectures and implementation.

In particular, many are the desirable features of a state-of-the-art simulation environment. For illustration purposes, we have compiled them in the following "wish" list: easier input of complex geometries and subsequent generation of the mesh; simplified grid refinement with dynamic capabilities; grid tracking of nonstationary boundaries without excessive mesh distortion; high-order accuracy numerical methods; efficient solution algorithms; general-purpose, computational constructs for data management and instruction execution; and, finally, inherently parallel procedures for implementation on the latest generation of concurrent computers. In the remaining of this section, we will outline the major complications related to these issues and summarize existing approaches for addressing them.
The initial step of geometry definition and grid generation is very often the most human-time-consuming task of the analysis. It is usually a manual process requiring much intuition and experience from the part of the designer. Although much progress has been made in geometric and solid modelling [6,12,44,52] for describing the geometry, and semi-automatic mesh generation [19,24,31,35,59,60,61] for subdividing the domain into small, manageable units, the process remains both tedious and complex, in particular for non-specialized complex geometries.

Adaptive grid refinement has been introduced over the last several years to alleviate shortcomings associated with the initially constructed mesh; a posteriori error estimates, integrated in the solution procedure, initiate an improved decomposition of the mesh with no human intervention. The methodology has resulted in improved efficiencies by capturing important aspects of flows that would have gone unnoticed in traditional fixed mesh schemes or would have become apparent at the cost of a whole new calculation. Particular adaptive grid strategies may involve relocation of nodes towards areas of steep gradients [9,46], or may proceed with a non-propagating refinement of the mesh in certain regions [7,46]. Both approaches have been applied mostly in the context of low-order finite volume [43] and finite element methods [8], but have not yet been widely implemented in higher-order methods where the discretization ceases to be local and (flexible) domain decomposition by triangles becomes prohibitively expensive.

Additional techniques in grid evolution have been produced for problems where parts of the boundary move in relation to others (e.g. rotor/stator configurations in turbomachinery equipment). Here, the domain is decomposed in a
set of two grids which can be either overlaid and exchange information via interpolation [14], or patched along an interface where necessary interpolations take place (one spatial dimension less) [51], or even joined by a thin strip which continuously deforms and is occasionally remeshed when distortions are excessive [25]. The flexibility of these methods however, in handling complex topologies, is obtained at the expense of high accuracy and programming simplicity.

As regards numerical methods for solving fluid-flow equations, spectral discretizations [13,29] have been optimal in terms of accuracy (or more precisely, convergence rate), for problems with simple geometries where regularity of the solution is guaranteed (e.g. incompressible fluid mechanics). The limitations due to geometry have been partially offset by the spectral element method [40,49] where additional geometric and resolution generality is introduced by combining domain decomposition and high-order interpolation. In cases however, where high-order regularity is the exception (e.g. compressible fluid mechanics), spectral methods will consistently perform no better than low-order finite-difference and finite element techniques [15,26,56]. The latter can capture complex geometries and are easy to implement, but, are efficient only in situations where coarse resolution is adequate. A hybrid approach, as suggested by Bernardi et al [10,11], which couples low-order finite elements with spectral methods for problems where the solution exhibits irregularities in some regions and is smooth in others, can offset the inherent limitations of individual methods and could provide optimal discretizations of fluid dynamical systems in the future.

Which ever the choice of solution approach may be, it has a direct consequence on the choice of solution algorithm. Traditionally, direct solution techniques, which perform a factorization of the resulting (sparse) matrix system of
discrete equations, have been successful with low-order methods for moderate size problems. High-order methods, resulting to a relatively full matrix, have been used in conjunction with iterative solution techniques. The latter, which avoid the cost of a matrix inversion by iterating upon an initial guess for the solution and performing matrix-vector products at every level, have become increasingly competitive for several reasons. First, in solving large problems, the use of iterative methods is almost always the case due to the stringent memory requirements associated with direct solvers. Second, if optimal (resolution-independent convergence rate) solvers are used, the operation count associated with the matrix vector products will favor an iterative method. Third, in time-dependent problems where the time-step changes dynamically according to stability or accuracy requirements, the multiple matrix inversions of direct methods are prohibitively expensive. Fourth, iterative methods tend to parallelize well due to the natural decoupling of the matrix vector products into smaller, independent substructures [21].

Programming implementation of the aforementioned discretizations, although practical in its nature, is becoming an issue of growing concern. The large development efforts required in writing large, general-purpose Computational Fluid Dynamics software has impeded efforts in the analysis of ever more complex physical systems. Generality and flexibility of programming models is also important when the synergies associated with mixing various solution techniques are to be exploited; more often than not, this results into cumbersome, detail-ridden, and user-unfriendly programs. Development of new data structures and algorithmic constructs that partially or fully standardize (and, therefore, automate) this programming development process should certainly be encouraged.
Finally, speed of computation has been the motivating factor behind recent advancements in computer architectures [34,55]. Large scale parallel processors can achieve many-fold increases in performance by assigning many, relatively simple computers to execute a single task in concert [22]. These great efficiencies are realized by careful balancing of computation to communication loads for the individual processors, exploitation of locality in the underlying solution algorithm, and sparsity of interdependencies in the discretizations [20,22,38].

The quality of research and the maturity of the discipline has led to a high-degree of sophistication in research approaches, as attested in this discussion. It is however, also true that existing open problems in Computational Fluid Dynamics exhibit a high degree of complexity and interdependence. In evaluating the relative merits of new ideas, it is imperative to consider the extent to which they address all the issues presented in this section. It is with this clear goal in mind that we have embarked on the development of the present effort.

1.1 About this thesis

The aim of this work is to draw on the competitive advantages of differing methodologies and present a new class of nonconforming discretizations for the unsteady, incompressible Navier-Stokes equations. The work, which is in the context of high-order methods, builds extensively on the spectral element method, first introduced by Patera [49] and Maday and Patera [40], and further enhanced by Karniadakis, Ronquist, Fischer, and Ho [21,32,36,54]. In the current approach, the computational domain is broken up into $K$ macro-elements with no geometric restrictions imposed on adjoining elements for coincidence.
of edges and corners (nonconformity). Inside each element, dependent and independent variables are approximated by $N^{th}$-order tensor product polynomial expansions, while, at elemental interfaces, a weak matching condition between expansions is required \[3,11\]. The decoupling of subdomains in the nonconforming decomposition allows for treatment of sliding grids; the governing equations in the moving region are extended by an additional convection term based on mesh velocity. Variational projection operators and Gauss numerical quadrature are used to generate the discrete equations, which are then solved by iterative techniques \[13,27,45,53\].

The nonconforming formulation is an important development in light of the issues presented in the previous section. It simplifies the initial mesh generation of complex geometries by transforming it in a two-stage process: a) individual meshing of simple subregions, followed by b) nonconforming patching to recover the original geometry. It allows for nonpropagating mesh refinement \[41\], which constitutes an extension and generalization of classical nonconforming $h$-type finite element methods \[15,17,58\] but which is of critical importance considering the restrictive global property of spectral discretizations. Its built-in ability to handle sliding meshes allows the method to analyze problems of nonstationary geometries without introducing any mesh distortion, expensive interpolation, or change of reference frame. It is developed in a way consistent with the standard properties of the conforming formulation \[40\] as regards optimality and convergence, and can be used with existing iterative solution techniques \[40,54\]. Finally, the weak, localized coupling of the formulation at elemental interfaces is implemented in a manner which minimizes complex data dependencies, provides the bandwidth for high-rate numerical processing, and lends itself well to parallelization.
The outline of the thesis is as follows. In Chapter 2, we present the detailed analysis of the nonconforming domain decomposition for two-dimensional elliptic equations. The discussion evolves on the choice of bases functions, spatial discretizations, and solution techniques for the discrete problem; ample numerical evidence demonstrates the optimality of the formulation. This is followed, in Chapter 3, by a description of the sliding mesh problem, temporal discretizations for the heat equation, and a brief summary of the full Navier-Stokes solution algorithm. Chapter 4 outlines the implementation of the method, presented in the context of a natural, data-flow-like programming model for the computational algorithm. Finally, Chapter 5 presents two case studies of unsteady aerodynamics and mixing, and closes with a summary of recommendations for future work.
Chapter 2
Nonconforming Spectral Element Discretizations

The linear, elliptic Laplace operator represents the “kernel” of our Navier-Stokes algorithm insofar that it involves the highest spatial derivatives and governs the continuity requirements and conditioning of the discrete system. Furthermore, the fully-discretized Navier-Stokes equations are typically solved at each time step by performing a series of elliptic solves. It is for these reasons that we chose to restrict the discussion here to second-order linear partial differential equations as this will demonstrate the essential features of the nonconforming discretizations.

2.1 Problem Formulation

We consider the problem of solving the Poisson equation on a domain $\Omega$ of $\mathbb{R}^2$: Find $u(x, y)$ such that

\begin{align*}
- \nabla^2 u &= f \quad\text{in } \Omega, \quad (2.1) \\
u &= 0 \quad\text{on } \partial\Omega, \quad (2.2)
\end{align*}

where $\partial\Omega$ is the boundary of $\Omega$, and $f$ is the prescribed force.
To define the space of acceptable solutions, we first recall the Lebesgue space \( L^2(\Omega) \) which consists of all functions that are square integrable, and which is equipped with the scalar product
\[
(\phi, \psi) = \int_\Omega \phi \psi \, d\Omega \quad \forall \phi, \psi \in L^2(\Omega)
\] (2.3)
and associated norm \( ||\phi||^2 = (\phi, \phi) \). We also define the space \( H^1(\Omega) \) consisting of all functions that are in \( L^2(\Omega) \) and whose first derivatives are also in \( L^2(\Omega) \). The corresponding norm for \( H^1 \) is defined as
\[
||u||_{H^1} = \left\{ \int_\Omega \left[ u^2 + (\nabla u)^2 \right] \, d\Omega \right\}^{1/2} \quad \forall u \in H^1(\Omega)
\] (2.4)

The variational formulation of problem (2.1, 2.2) can now assume the following form: Find \( u(x, y) \) such that
\[
\int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega, \quad \forall v \in X,
\] (2.5)

or
\[
(\nabla u, \nabla v) = (f, v) \quad \forall v \in X
\] (2.6)
where \( X \) is the space of functions \( H^1(\Omega) \) which are zero on the boundary \( \partial \Omega \), i.e.
\[
X = \left\{ \phi \mid \phi \in L^2(\Omega), \ \nabla \phi \in L^2(\Omega), \ \phi|_{\partial \Omega} = 0 \right\}.
\] (2.7)

**Remark 2.1**: Note that equation (2.6) involves only first derivatives of the function \( u \), meaning that only Dirichlet boundary conditions need to be imposed; boundary conditions on \( u \) of Neumann type are naturally satisfied. Furthermore, from the viewpoint of applied mathematics, the variational form has the additional advantage that existence and uniqueness of the discrete solution can be rigorously proved. □

The Galerkin discretization of the problem (2.6) consists of testing the variational form with respect to a family of discrete finite dimensional spaces \( X_h \).
The discrete problem becomes: Find \( u_h(x, y) \) such that

\[
(\nabla u_h, \nabla v_h)_h = (f, v_h)_h \quad \forall v_h \in X_h
\]  

(2.8)

where \((\cdot, \cdot)_h\) denotes numerical quadrature corresponding to the continuous integrals of (2.5).

To proceed with the spectral element discretization we choose a pair of integers \( h = (K, N) \) and divide the domain \( \Omega \) into rectangular subdomains \( \Omega^k \), \( k = 1, ..., K \) such that

\[
\Omega = \bigcup_{k=1}^{K} \Omega^k, \quad \forall k, l, k \neq l, \quad \Omega^k \cap \Omega^l = \emptyset
\]  

(2.9)

\[
\Omega^k = [\alpha_k, \alpha'_k] \times [\beta_k, \beta'_k].
\]  

(2.10)

We also define the space of all polynomials of degree less than or equal to \( N \)

\[
P_{(N,K)}(\Omega) = \{ \phi \in L^2(\Omega), \; \phi|_{\Omega^k} \in P_N(\Omega^k) \}.
\]  

(2.11)

**Remark 2.2:** In the classical conforming case, where edges and corners of adjacent elements coincide, the space of approximation for the solution \( u \) consists of the subspace \( X_h \) of \( H^1_0(\Omega) \) corresponding to all piecewise high-order polynomials of degree less than or equal to \( N \)

\[
X_h = H^1_0(\Omega) \cap P_{(N,K)}(\Omega).
\]  

(2.12)

Existence and uniqueness of the discrete solution is ensured by the continuity and ellipticity of the bilinear form \((\nabla u_h, \nabla v_h)_h\). Ellipticity implies stability and, coupled with consistency, leads to convergence to the exact solution. The magnitude of the error between discrete and exact solutions is bounded by a
constant times the best polynomial fit in this finite dimensional subspace $X_h$ \cite{[40]}. □

For the nonconforming approximation, $X_h$ is not a subspace of $H_0^1(\Omega)$, thereby introducing additional “consistency” errors. We will now describe the form of $X_h$ for the nonconforming spectral element discretizations in a manner similar to Bernardi et al \cite{[11]}.

To begin, we identify the $K$ rectangular subdomains of (2.9) as spectral elements, with edges denoted $\Gamma_{k,l}$, $l = 1, ..., 4$,

$$\partial \Omega^k = \bigcup_{l=1}^{4} \Gamma_{k,l}.$$

We then define the skeleton $S$ of the problem as

$$S = \bigcup_{k=1}^{K} \partial \Omega^k. \quad (2.13)$$

We next introduce a non-unique set of $M$ non-empty “mortars”, $\gamma_p$, as follows;

$$\gamma^p = \Gamma_{k,l} \text{ for some } k,l \text{ such that } \left\{ \begin{array}{l} S = \bigcup_{p=1}^{M} \gamma^p \\ \gamma^p \cap \gamma^q = 0 \quad p \neq q \end{array} \right. \quad (2.14)$$

where $p$ is an arbitrary enumeration $p = 1, ..., M$. The intersection of the closures of the $\gamma^p$ defines a set of vertices, $\mathcal{V}$, composed of all non-empty elements

$$v^q = (\bar{\gamma}^m \cap \bar{\gamma}^n) \text{ such that } v^q \notin \gamma^p \quad \forall \quad p = 1, ..., M \quad , \quad (2.15)$$

where $q$ is an arbitrary enumeration $q = 1, ..., \mathcal{V}$. Finally, we define the set of virtual vertices, $\hat{\mathcal{V}}$, composed of all non-empty elements

$$\hat{v}^q = \bar{\gamma}^m \cap \gamma^n \quad , \quad (2.16)$$
for \( q = 1, \ldots, \tilde{V} \); the union of the sets \( \mathcal{V} \) and \( \tilde{\mathcal{V}} \) represents the corners of all subdomains \( \Omega^k \). The geometry of a nonconforming domain decomposition into \( K = 4 \) elements is illustrated in Figure 1a with the associated skeleton structure shown in Figure 1b.

**Remark 2.3:** Note that the *conforming* spectral element discretization corresponds to the case where we (can) choose a mortar set in which \( \Gamma^{kl} \) equals \( \gamma^p \) for some \( p \) for all \( k, l \); the earlier "refinement" mortar method, presented in Mavriplis [41], corresponds to the case where there are no virtual vertices. □

In order to define the nonconforming space of functions \( X_h \), we first require an auxiliary mortar space \( W_h \) of continuous functions along the skeleton \( S \)

\[
W_h = \{ \phi \in C^0(S), \, \forall p = 1, \ldots, M, \, \phi_{|\gamma^p} \in P_N(\gamma^p), \, \phi_{|\partial \Omega} = 0 \} . \tag{2.17}
\]

The space \( X_h \) is finally defined as the subspace of \( P_{(N,K)}(\Omega) \) of all functions \( v \) such that

1) \( v \) vanishes on \( \partial \Omega \)
2) there exists a function \( \phi \in W_h \) such that

\[
\forall q \text{ vertex of } \Omega^k, \quad v_{|\Omega^k}(v_q) = \phi(v_q), \quad 1 \leq k \leq K \tag{2.18}
\]

\[
\forall q \text{ virtual vertex of } \Omega^k, \quad v_{|\Omega^k}(\tilde{v}_q) = \phi(\tilde{v}_q), \quad 1 \leq k \leq K \tag{2.19}
\]

\[
\forall l = 1, \ldots, 4, \, \forall k = 1, \ldots, K, \, \forall \psi \in P_{N-2}(\Gamma^{k,l}) , \quad \int_{\Gamma^{k,l}} (v_{|\Omega^k} - \phi) \psi ds = 0. \tag{2.20}
\]

Let us now summarize the properties of the approximation space \( X_h \). First, for a conforming decomposition, \( X_h \) recovers the standard spectral element form of (2.12). In the nonconforming case, the uniqueness of solution follows again from the ellipticity and continuity of the Laplacian operator and is rigorously
**Figure 1:** Nonconforming subdomain decomposition (a) and associated skeleton structure (b) with vertices $v^q$ (○), virtual vertices $\bar{v}^q$ (○), and mortars $\gamma^p$ (□).
demonstrated in [11]. Furthermore, the $L^2$ condition (2.20) ensures that the jump in functions is small in the interior of elemental boundaries, whereas the vertex conditions (2.18, 2.19) assure exact continuity at cross points where the normal derivative has more than one sense; the combination of both is essentially equivalent to an $H^1$ continuity across the interfaces. In addition, the coupling between adjacent elements (2.20) is by construction democratic as the functions on the two subdomains are deduced from the same, external mortar function. The local nature of (2.20), based on explicit construction of the nonconforming function space $X_h$ rather than Lagrange multiplier constraints [17], becomes important in parallel computations as subdomain evaluations can proceed independently with transmission of continuity conditions provided by the auxiliary mortar structure. The method, specifically designed with these goals in mind, allows for simple implementatation of arbitrary element topologies.

The nonconforming spectral element discretization corresponds to numerical quadrature of (2.8). To proceed, we use an affine mapping: $x \in \Omega^k \rightarrow r = (r_1, r_2) \in [-1,1]^2$, and perform tensor-product Gauss-Lobatto quadrature $\times$ Gauss-Lobatto quadrature in $(r_1, r_2)$ to obtain from (2.8)

$$\langle \nabla u_h, \nabla v_h \rangle_{h,GL} = \langle f, v_h \rangle_{h,GL} \quad \forall v_h \in X_h,$$

or

$$\sum_{k=1}^{K} \sum_{p=1}^{N} \sum_{q=0}^{N} \rho_p^1 \rho_q^2 \nabla u_h(\xi_{p,k}, \xi_{q,k}) \cdot \nabla v_h(\xi_{p,k}, \xi_{q,k}) = \sum_{k=1}^{K} \sum_{p=0}^{N} \sum_{q=0}^{N} \rho_p^1 \rho_q^2 \int_{v_h}(\xi_{p,k}, \xi_{q,k}) \quad \forall v_h \in X_h.$$

Here $\xi_{p,k} = \alpha_k + (\alpha'_k - \alpha_k)(\xi_p + 1)/2$ and $\xi_{q,k} = \beta_k + (\beta'_k - \beta_k)(\xi_q + 1)/2$ are the locations of the local nodes, $\rho_p^1 = \rho_p(\alpha'_k - \alpha_k)/2$ and $\rho_q^2 = \rho_q(\beta'_k - \beta_k)/2$ are the integration weights including geometry factors, $\xi_p, \rho_p$ are, respectively,
the Gauss-Lobatto-Legendre quadrature points and weights [57], and, finally, \( \nabla \) is the gradient operator with respect to \( \mathbf{x} \). It is worthwhile to note that \( N^{th} \) order Gauss-Lobatto quadrature is exact for polynomials of degree \( 2N - 1 \) or less [57] (i.e. the left-hand side of equation (2.22) is integrated exactly, while the right-hand side is not).

A detailed discussion of the overall discretization error which consists of approximation, quadrature and consistency errors, is provided in [11]. Here we shall only reiterate the main result which takes the form

\[
\| u - u_h \|_{H^1} \leq C \left( N^{1-\sigma} \| u \|_{H^\sigma} + N^{-\rho} \| f \|_{H^\rho} \right)
\] (2.23)

where \( C \) is a constant, \( u \in H^\sigma(\Omega) \), and \( f \in H^\rho(\Omega) \) (i.e. \( \sigma \) and \( \rho \) reflect the regularity of the exact solution and data, respectively). Relation (2.23) has the same asymptotic behavior as the error estimate in the conforming case [40] and indicates that the spectral element solution converges, for \( K \) fixed, to the exact solution \( u \), as \( N \to \infty \), at a rate faster than any algebraic power (exponential) for problems with infinitely smooth (analytic) data and solutions.

### 2.2 Bases Representation

In order to reduce in practice the discrete problem (2.21) we also need to define bases for the polynomial spaces \( W_h \) and \( X_h \). The particular form of basis does not effect the optimal error estimate (2.23), however it is critical for the conditioning and structure of the matrix system of algebraic equations. The choise of basis is also important as regards parallelism in that it effects inter-elemental couplings and, hence, communications among processors. We
choose the tensor product form of one-dimensional Gauss-Lobatto interpolants to represent \( v_h \in X_h \),

\[
v_h^k(r_1, r_2) = \sum_{i=0}^{N} \sum_{j=0}^{N} v_{ij}^k h_i^n(r_1) h_j^n(r_2) \quad \forall k = 1, \ldots, K, \tag{2.24}
\]

where \( v_{ij}^k = v_h^k(\xi_1^i, \xi_2^j) \), and the \( h_i \)'s are one-dimensional Gauss-Lobatto Lagrangian interpolants. The interpolants, constructed with Legendre polynomials \( (L_N(z)) \) and their derivatives \( (L'_N(z)) \), have the following properties

\[
h_i \in P_N([-1, 1]); \quad h_i(\xi_j) = \delta_{ij} \tag{2.25}
\]

where \( \delta_{ij} \) is the Kronecker-delta symbol. They are expressed as

\[
h_i(z) = -\frac{1}{N(N+1)L_N(z_i)} \frac{(1-z^2)L'_N(z)}{z - z_i} \quad z \in [-1, 1]. \tag{2.26}
\]

\textbf{Remark 2.4:} Our choice of basis has four important properties. First, it reduces the operation count of matrix-vector products from \( O(N^{2d}) \) to \( O(N^{d+1}) \) and storage from \( O(N^{2d}) \) to \( O(N^d) \); this is perhaps, the single most important contribution toward the efficiency of spectral methods, as first recognized by Orszag [48]. Second, by virtue of (2.25), the basis is nodal, meaning that the coefficient \( v_{ij}^k \) is the solution \( v_h^k \) at the point \((i,j)\). Third, the interpolants are orthogonal to each other in the discrete sense, which results to a diagonal mass matrix (the variational equivalent to the identity operator). Fourth, only a few degrees-of-freedom are nonzero along elemental boundaries which substantially favors the ratio of computation to communication in parallel implementations [21].

Similarly, for the auxiliary mortar space \( W_h \) we have chosen \( N^{th} \)-order Lagrangian interpolants through the Gauss-Lobatto-Legendre points as the basis.
The function $\phi$ of (2.17) is defined as

$$\phi_{(t^r)} = \sum_{j=0}^{N} \phi_j^p h_j^N (\hat{s}), \quad \forall p \in \{1, \ldots, M\},$$

(2.27)

where $\hat{s}$ is a mortar-local variable, defined on $[-1, 1]$.

Remark 2.5: The element-internal degrees-of-freedom, $v_{ij}^k$, $i, j \in \{1, \ldots, N - 1\}$ are clearly free, however, the degrees-of-freedom lying on elemental edges are constrained to the mortars and vertices through conditions (2.18-2.20). Likewise, virtual vertices are not free either, since they obtain their value from the corresponding mortar function which is evaluated at their particular location. It is the union of element-internal, mortar internal, and vertex points which forms the set of independent degrees-of-freedom, and, therefore, denotes the finite-dimensional approximation space of the continuous problem (2.5).

Finally, in order to express the matching condition (2.20) in matrix form we also require a basis for $\psi$, which we choose as

$$\psi_{(t^r)} = \sum_{q=1}^{N-1} \beta_q \eta_q^{N-2} (\hat{s})$$

(2.28)

where

$$\eta_q^{N-2}(z) = (-1)^{N-q} \frac{L_N'(z)}{\xi_q - z}, \quad z \in [-1, 1[, \quad q \in \{1, \ldots, N - 1\},$$

(2.29)

and $\hat{s}$ is a local variable for some particular edge $\Gamma^{k,l}$, $\hat{s} \in [-1, 1]$. The basis functions $\eta_q^{N-2}(z)$ are again, by construction, nonzero at only one particular Gauss-Lobatto Legendre quadrature point along an elemental edge; this leads, as shown in the following section, to a diagonal projection matrix in the case of conforming domain decomposition.
2.3 Discrete Equations

2.3.1 Integral Matching Condition

In order to implement the matching condition (2.20), we require further notation. In Figure 2 we illustrate an arbitrary mortar/edge configuration, and introduce the notions of mortar offset $s_0$, mortar $\gamma_p$ of length $|\gamma_p|$, elemental edge $\Gamma^{kl}$ of length $|\Gamma^{kl}|$, and integration strip $\gamma = \Gamma^{kl} \cap \gamma_p$ of length $|\gamma|$. We also show the intersection points $\xi_+^*$ and $\xi_-^*$ of the mortar $\gamma_p$ with the top and bottom corners of the corresponding integration strip $\gamma$, given in mortar-local coordinates (i.e. $\xi_+^*, \xi_-^* \in [-1, 1]$). Note that all other lengths are in absolute $(x, y)$ coordinates. We also introduce the notation

$$r_{+i} = \begin{cases} h_i(\xi_+^*) & \text{if } |\gamma| + |\min(0, -s_0)| = |\Gamma^{kl}| \\ 0 & \text{otherwise} \end{cases}$$

and

$$r_{-i} = \begin{cases} h_i(\xi_-^*) & \text{if } s_0 \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

for some particular edge-mortar combination. During actual implementation of $r_{+i}$ and $r_{-i}$, the two stringent equality conditions need only be met to within a pre-specified tolerance level $\varepsilon$; stability ensures a no-greater-than-commensurate increase in the discretization error.

With the bases defined in the previous section, we perform (here exact) piecewise Gauss-Lobatto quadrature on $N + 1$ points on a particular elemental edge $\Gamma^{kl}$ and integration strip $\gamma$, giving

$$\sum_{j=1}^{N-1} \overline{B}_{ij} \phi_j = \sum_{j=0}^{N} \overline{P}_{ij} \phi_j, \quad \forall i \in \{1, ..., N - 1\}$$

(2.30)
Figure 2: Illustration of a mortar offset $s_0$, edge $\Gamma^{k,l}$, mortar $\gamma^p$, integration strip $\hat{\gamma}$, and intersection points $\xi^+_1$ and $\xi^-_1$ for a particular edge-mortar combination.
where for the destination edge $\Gamma_{k,l}$ we have

$$\int_{\Gamma_{k,l}} \psi \rightarrow B_{ij} = \frac{|\Gamma_{k,l}|}{2} (-1)^{N-i}(-L''_N(\xi_j)) \rho_i \delta_{ij}, \quad \forall i, j \in \{1, \ldots, N-1\}^2.$$  

(2.31)

For $f_\psi \phi ds \rightarrow \overline{P}_{ij}$ we have, for $s_0 \geq 0$,

$$\overline{P}_{ij} = \frac{|\gamma|}{2} \sum_{q=0}^{N} \rho_q \eta_i^{N-2} \left( \frac{2s_0}{|\Gamma_{k,l}|} - 1 + (1 + \xi_j) \frac{|\hat{\gamma}|}{|\Gamma_{k,l}|} \right) h_j \left( -1 + (1 + \xi_j) \frac{|\hat{\gamma}|}{\gamma^p} \right)$$

$$- \frac{|\Gamma_{k,l}|}{2} \eta_i^{N-2} (-1) r_{-i} \rho_0 - \frac{|\Gamma_{k,l}|}{2} \eta_i^{N-2} (1) r_{+i} \rho_N$$

(2.32)

whereas for $s_0 < 0$,

$$\overline{P}_{ij} = \frac{|\gamma|}{2} \sum_{q=0}^{N} \rho_q \eta_i^{N-2} \left( -1 + (1 + \xi_j) \frac{|\hat{\gamma}|}{|\Gamma_{k,l}|} \right) h_j \left( \frac{2s_0}{|\gamma^p|} - 1 + (1 + \xi_j) \frac{|\hat{\gamma}|}{\gamma^p} \right)$$

$$- \frac{|\Gamma_{k,l}|}{2} \eta_i^{N-2} (-1) r_{-i} \rho_0 - \frac{|\Gamma_{k,l}|}{2} \eta_i^{N-2} (1) r_{+i} \rho_N.$$  

(2.33)

The integral matching condition (2.20) may then be expressed in matrix form via the transformation matrix $[Q]$, given as

$$\overline{Q}_{ij} = [Q] = [B]^{-1} [\overline{P}], \quad \forall i \in \{1, \ldots, N-1\}, \forall j \in \{0, \ldots, N\}.$$  

(2.34)

It is worthwhile to note that the formulation of the $[Q]$ matrix is stable as regards virtual vertices: their contribution vanishes uniformly as they approach true vertices from any direction.

Figure 3 is a diagrammatic representation of the integral matching condition, and, in fact, the entire $X_\alpha$ basis for the model problem of Figure 1. The arrows represent the descendance of mortar data to elements with the assumed priority that, in any location, all incoming contributions will be completed before any outgoing contributions can be initiated. The vertex condition is illustrated by
Figure 3: Diagrammatic representation of the basis for $X_h$ on the nonconforming decomposition of Fig. 1(a).
the arrows emanating from the large open boxes or circles which represent real and virtual vertices, respectively; vertex data is assigned equally to the local mortar endpoints (∙) as well as the elemental data vertices (○). The mortar projection (multiplication by the \([Q]\) transformation matrix) is represented by arrows between mortar strips and elemental edges. Note that more than one arrows leading to the same edge location indicates a sum of the contributions of all intersections of \(\gamma^p\) with the particular \(\Gamma^{k,l}\), whereas equations (2.32) and (2.33) refer to just one such intersection (i.e., just one integration strip). The role of the virtual vertex is primarily that of an intermediate storage location holding the value of the function \(\phi\) and descending it to elemental corners.

Although in practice we evaluate \(v^k\) from the diagram of Figure 3 without explicitly forming the global projection operator \(\bar{Q}\), it is nevertheless useful to remark that the diagram is equivalent to

\[
\begin{pmatrix}
    v_{ij}^k|_{\text{interior}} \\
    v_{ij}^k|_{\text{edges}}
\end{pmatrix}
= \begin{pmatrix}
    [I] & 0 \\
    0 & \bar{Q}
\end{pmatrix}
\begin{pmatrix}
    v_{ij}^k|_{\text{interior}} \\
    \phi(v_q), \forall q, \forall p, \forall j \in \{1, \ldots, N - 1\}
\end{pmatrix}
\forall k
\]

(2.35)

By referring as \([Q]\) the combination of the \([\bar{Q}]\) operator for a particular mortar-edge pair along with the two vertex assignments associated with that mortar, we can rewrite (2.35) as

\[
v = Qv
\]

(2.36)

where \(Q\) refers to the global projection operator for all mortar-edge pairs and corner-vertex assignments, \(v\) represents the set of the algebraic degrees-of-freedom, and \(v\) is the set of admissible, elemental unknowns. We note that our basis construction allows us to express admissible elemental degrees-of-freedom in terms of their images via the \(Q\) transformation. This, in turn, permits us to
construct the global discrete equations directly from local structure-preserving elemental equations, which is the essence of the discretization-driven domain decomposition approach.

2.3.2 Fully Discrete Equations

We have, so far, presented the essential features of the domain decomposition with an example involving rectilinear geometry (Figure 1a). We will proceed in the derivation of the matrix system of equations by considering the more general case of a domain with curved geometry decomposed into $K$ disjoint quadrilateral elements. We will make use of isoparametric transformations [13,56] to approximate the geometry of the curved elements with polynomials of the same order as the solution itself.

We will first describe the derivation of the elemental discrete matrices by rewriting the integral in (2.21) as a sum of contributions from individual elements $\Omega^k$,

$$
(\nabla u_h, \nabla v_h)_{h,\Omega LL} = \sum_{k=1}^{K} (\nabla u_h, \nabla v_h)^k
$$

(2.37)

where superscript $k$ refers to the particular subdomain $\Omega^k$. The elemental integrals of (2.37) are mapped, as shown in Figure 4, into a local $(r,s)$-system [28], giving

$$
(\nabla u_h, \nabla v_h)^k = \int_{-1}^{1} \int_{-1}^{1} \tilde{\nabla} u^k(r,s) \cdot \tilde{\nabla} v^k(r,s) |J^k(r,s)| drds
$$

(2.38)

where $\tilde{\nabla}$ now denotes differentiation with respect to the local $(r,s)$ coordinates,

$$
\tilde{\nabla} = \left( \frac{\partial r \partial}{\partial x \partial r} + \frac{\partial s \partial}{\partial x \partial s} \right) \hat{e}_z + \left( \frac{\partial r \partial}{\partial y \partial r} + \frac{\partial s \partial}{\partial y \partial s} \right) \hat{e}_y
$$

(2.39)
Figure 4: Each quadrilateral spectral element is mapped into a local \((r, s)\)-system:
\((x, y) \in \Omega_k \Rightarrow (r, s) \in [-1, 1]^2\).
and $J$ is the Jacobian related to the mapping transformation,

$$J = \left( \frac{\partial x}{\partial r} \frac{\partial y}{\partial s} - \frac{\partial x}{\partial s} \frac{\partial y}{\partial r} \right).$$  \hspace{1cm} (2.40)

Similarly, we write

$$(f, v_h)^k = \int_{-1}^{1} \int_{-1}^{1} |J^k| f^k(r, s) v^k(r, s) dr ds.$$  \hspace{1cm} (2.41)

In (2.39) we note that

$$\frac{\partial r}{\partial x} = \frac{1}{J} \frac{\partial y}{\partial s}, \quad \frac{\partial s}{\partial x} = \frac{-1}{J} \frac{\partial y}{\partial r}, \quad \frac{\partial r}{\partial y} = \frac{-1}{J} \frac{\partial x}{\partial s}, \quad \frac{\partial r}{\partial s} = \frac{1}{J} \frac{\partial x}{\partial r};$$  \hspace{1cm} (2.42)

and we rewrite (2.39) as

$$\hat{\nabla} = \frac{1}{J} \left\{ \left( \frac{\partial y}{\partial s} \frac{\partial}{\partial r} - \frac{\partial y}{\partial r} \frac{\partial}{\partial s} \right) \hat{e}_x + \left( \frac{\partial x}{\partial s} \frac{\partial}{\partial r} + \frac{\partial x}{\partial r} \frac{\partial}{\partial s} \right) \hat{e}_y \right\}. \hspace{1cm} (2.43)

We now express the discrete solution $u_h$, test functions $v_h$, data $f$, as well as the independent variables $x(r, s, t)$ and $y(r, s, t)$ in terms of the elemental bases shown in (2.24). Relations (2.38, 2.41) become

$$(\nabla u_h, \nabla v_h)^k = \sum_{i=0}^{N} \sum_{j=0}^{N} \rho_i \rho_j \frac{1}{J_{ij}} \left\{ g_{ij}^{1,k}(D_{ip} v_{pj}^k)(D_{iq} u_{qj}^k) + g_{ij}^{2,k}(D_{jp} v_{ip}^k)(D_{jq} u_{iq}^k) + g_{ij}^{3,k}(D_{ip} v_{pj}^k)(D_{jq} u_{iq}^k) \right\}, \hspace{1cm} (2.44)$$

and

$$(f, v_h)^k = \sum_{i=0}^{N} \sum_{j=0}^{N} \rho_i \rho_j J_{ij}^k f^k_{ij} v_{ij}^k \hspace{1cm} (2.45)$$

where

$$g_{ij}^{1,k} = (D_{jq} u_{iq}^k)^2 + (D_{jq} x_{iq}^k)^2 \hspace{1cm} (2.46)$$
$$g_{ij}^{2,k} = (D_{ip} y_{pj}^k)^2 + (D_{ip} x_{pj}^k)^2 \hspace{1cm} (2.47)$$
$$g_{ij}^{3,k} = - \left[ (D_{jq} y_{iq}^k)(D_{ip} y_{pj}^k) + (D_{jq} x_{iq}^k)(D_{ip} x_{pj}^k) \right] \hspace{1cm} (2.48)$$
$$J_{ij}^k = (D_{ip} x_{pj}^k)(D_{jq} y_{iq}^k) - (D_{jq} x_{iq}^k)(D_{ip} y_{pj}^k) \hspace{1cm} (2.49)$$
and where
\[ D_{ij} = \frac{d h_j}{d r}(\xi_i). \] (2.50)

Since (2.44, 2.45) hold for arbitrary \( v_h \in X_h \), we set \( v_{ij} \) to be nonzero at only one global collocation point \((i,j)\). By introducing as \( A^k \) and \( B^k \) the elemental Laplacian and mass matrix operators of equations (2.44, 2.45), we obtain
\[
A^k u_{ij}^k = \sum_{p=0}^{N} D_{pi} \left[ \left( D_{pq} u_{ij}^k \right) g_{pj}^1 + \left( D_{jq} u_{pq}^k \right) g_{pq}^3 \right] \frac{\rho_{pj}}{J_{pj}} + \sum_{p=0}^{N} D_{pj} \left[ \left( D_{pq} u_{ij}^k \right) g_{ip}^2 + \left( D_{iq} u_{qp}^k \right) g_{ip}^3 \right] \frac{\rho_{ip}}{J_{ip}} \] (2.51)
\[
B^k f_{ij}^k = \rho_i \rho_j J_{ij} f_{ij}^k \] (2.52)

As a shorthand notation, we will use \( \text{blk}(A^k) \) and \( \text{blk}(B^k) \) to refer to all elemental Laplacian and mass matrix operators, respectively.

To arrive at the global matrix system of discrete equations we need to assemble the evaluations performed at the elemental level back to the form of the algebraic degrees-of-freedom. In particular, elemental contributions to nodes that are shared by more than one element need to be accounted for. This last step is customarily known as the direct-stiffness procedure [20,40] which, in our case, takes the form of the transpose \( Q^T \) of the projection operation (2.36). Indeed, from the conforming formulation where we minimize the inner product \( v^T A v \), the functions \( v \) are no longer admissible but must be replaced by \( Qv \) which leads us to \( v^T Q^T A Q v \). As a consequence, the fully-discrete equations for the Poisson problem of (2.5) are:
\[
Q^T \text{blk}(A^k) Q = Q^T \text{blk}(B^k) Q f \] (2.53)
or
\[
A u = B f \] (2.54)
where $A$ and $B$ are the global Laplacian and mass matrices which include projection and direct-stiffness. In other words, equations (2.53,2.54) treat the global Laplace and mass operators as local operators "mortared" together by the $Q^T, Q$ transformations. In the implementation of iterative procedures the $Q, Q^T$ are, of course, never explicitly formed, but rather are evaluated; diagrammatic evaluation of $Q^T$ (direct stiffness summation) is shown in Figure 5 for the model problem of Figure 1.

We will now demonstrate numerically the rapid convergence rate, implied in (2.23), by considering the problem (2.1) with $f(x) = 2\pi^2 \sin(\pi x) \sin(\pi y)$, exact solution $u(x, y) = \sin(\pi x) \sin(\pi y)$, and the geometry of Figure 1a. A typical mesh ($N = 6$) is shown in Figure 6. As the data and the solution are analytic, we expect exponential convergence to the exact solution. Figure 7 illustrates the behavior of the $H^1$-error as a function of $N$ for $K = 4$ on a log-lin scale. Recalling (2.4), the error norm is defined as

$$||u - u_h||_{H^1} = \left\{ \int_{\Omega^k} [(u - u_h)^2 + (\nabla u - \nabla u_h)^2] \, d\Omega \right\}^{1/2} \quad (2.55)$$

where the integral in (2.55) is approximated by Gauss-Lobatto Legendre quadrature.

We finally compare the current approach with the traditional spectral element method by considering solution to the problem (2.1) with $f(x) = 2\pi^2 \sin(\pi x) \sin(\pi y)$, exact solution $u(x, y) = \sin(\pi x) \sin(\pi y)$, and the three different curved geometries depicted in Figure 8. The problem domain is that of a quarter annulus with center at (-1,-1) and inner and outer radii of one and two, respectively; decompositions (a) and (c) are conforming while (b) is nonconforming. The $H^1$-error, Figure 9, for the nonconforming case remains bounded by the approximations in cases (a) and (c); the aberration is attributed to the differing number of degrees.
Figure 5: Direct stiffness summation $Q^T$ of residuals on the nonconforming decomposition of Figure 1a.
Figure 6: Mesh for the $K = 4$ problem of Figure 1(a) corresponding to polynomial order $N = 6$ in each spatial direction. Each intersection of lines represents an elemental degree-of-freedom.
Figure 7: Plot of the $H^1$-error in the spectral element solution to the differential equation (2.1) for the geometry in Figure 1(a) as a function of polynomial order $N$. The number of elements is fixed to $K = 4$; exponential convergence is achieved as the polynomial degree is increased.
Figure 8: Three different domain decompositions for a two-dimensional quarter annulus; Cases (a) and (c) are conforming, while (b) is nonconforming. The overall number of degrees of freedom is smallest in (a) and largest in (c).
Figure 9: A plot of the $H^1$-error in the spectral element solution of the differential equation (2.2) as a function of polynomial order $N$ for the three geometries shown in Figure 8; (*) corresponds to case (a); (o) to case (b); and (+) to case (c). Exponential convergence is attained in all three cases. The error in the nonconforming decomposition is bounded from above and below by the errors to the conforming decompositions since the latter two have a smaller and larger (total) number of degrees-of-freedom, respectively.
of freedom between geometries (a)-(c). Figure 9, thereby confirms the result of the theoretical error analysis (2.23) that the additional "consistency" errors \( X_h \notin H^1 \) are of the same order as the approximation errors. Note also that although the domain \( \Omega \) is relatively deformed compared to the mapped rectilinear problem, high accuracy is still obtained due to the good interpolation properties of Legendre Lagrangian interpolants.

### 2.4 Iterative solution

Although the emphasis here is on the mortar discretization, the bases and evaluation procedure have been tailored to admit efficient iterative solution, and it is therefore appropriate to briefly indicate how the method is used in conjunction with iterative solution techniques. The conjugate gradient iteration [27] is a well-documented iterative method and relatively easy to implement. To solve (2.53) we write

\[
\begin{align*}
\mathbf{u}_0: & \quad r_0 = Q^T blk(B^k)Q_f - Q^T blk(A^k)Q\mathbf{u}_0; \quad q_0 = r_0 \\
& \quad a_m = (r_m, r_m)/(q_m, Q^T blk(A^k)Q q_m) \\
& \quad \mathbf{u}_{m+1} = \mathbf{u}_m + a_m q_m \\
& \quad r_{m+1} = r_m - a_m Q^T blk(A^k)Q q_m \\
& \quad b_m = (r_{m+1}, r_{m+1})/(r_m, r_m) \\
& \quad q_{m+1} = r_{m+1} + b_m q_m,
\end{align*}
\]

where \( m \) refers to iteration number, \( r_m \) is the residual, \( q_m \) the search direction and \((.,.)\) is the usual discrete inner product. For the model problem of Figure 1, all \( Q, Q^T \) transformations are performed through the diagrams of Figures 3
and 5, while Laplacian operations are performed locally at the elemental level, and, are therefore, naturally parallelizable [20]. The matrix vector products implied in $blk(A^k)u^k$ are the most critical as regards computational efficiency since they involve the most operations; inspection of (2.51) indicates that they can be evaluated in $O(N^3)$ for each element, for a total of $O(KN^3)$ operations for a two-dimensional problem.

The number of iterations required for convergence scales like $\sqrt{\kappa_A}$, where

$$\kappa_A = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

is the condition number of the global Laplacian operator $A$, and $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$ are the maximum and minimum eigenvalues of $A$. For the spectral element equations it is estimated [40] that

$$\kappa_A \sim O(K^2_1N^3)$$

(2.57)

thus implying, that the solution of (2.53) will require $O(K_1N^{3/2})$ iterations to converge. Here $K_1$ is the number of spectral elements in a typical spatial direction.

Convergence in less iterations could be achieved by the use of a preconditioner [13,27]. Although preconditioners based on the diagonal of $A$, or on an incomplete Cholesky factorization of an $h$-type finite element method have performed well in conforming decompositions [20,40,54], they have been ineffective here. The design of an effective preconditioner for nonconforming spectral discretizations is still considered an open problem.
Chapter 3

Sliding Meshes

Many important problems in fluid mechanics involve nonstationary geometries where parts of the boundary are moving in a prescribed manner. These kinds of problems render the task of numerical approximation to the governing partial differential equations even more difficult by introducing new geometric and discretization difficulties associated with the deforming domain. A step toward alleviating this additional complexity is to allow for part of the computational domain to translate (slide) in a fashion complying with the continuously evolving geometry. The advantages of sliding meshes lie in that they eliminate expensive remeshing procedures between consecutive time steps and allow for translations to occur without introducing excessive mesh distortions. Sliding meshes have been implemented in structural mechanics [30] as well as in compressible flow simulations of turbomachinery equipment [14,25,51]. The latter schemes are primarily low-order approximations to the compressible Euler equations and employ a variety of turbulence models to simulate the behavior of the small flow structures (length-scales below the typical spacing between grid points).

In the present approach, the domain decomposition of Chapter 2 is particularly well-suited for simulation of moving boundary problems. The (now, time-dependent) nonconforming discretizations are applied to the governing equations with a scheme which, rather than resorting to (potentially) expensive interpola-
tions between consecutive time steps, is now extended by considering advection due to grid velocity. The emphasis in this chapter is on temporal discretizations; the essential features of the methodology are presented for the model heat equation, and are followed by a description of the algorithm for solving the fully-discrete Navier-Stokes equations.

3.1 Formulation

We consider the solution to the heat equation: Find \( u(x, t) \) such that

\[
\frac{\partial u}{\partial t} - \nabla^2 u = 0 \quad \text{in} \quad \Omega(t) \tag{3.1}
\]

\[
u(x, t) = 0 \quad \text{on} \quad \partial \Omega \tag{3.2}
\]

\[
u(x, 0) = u_0(x) \tag{3.3}
\]

where \( \Omega(t) \) is a domain with nonstationary boundaries.

Remark 3.1: Problem (3.1-3.3) is an idealized physical situation since the evolution, in general, of the nonstationary boundaries of \( \Omega(t) \) induces motion in the conduction medium itself. The study therefore, of the diffusion process should be coupled with the Navier-Stokes equations. On the other hand, if all boundaries of \( \Omega \) are stationary, or rotationally or translationally invariant, there is no need for sliding meshes. Our intent here, is only to present the formulation of sliding meshes separately from the more complicated Navier-Stokes equations; potential physical applications with moving boundaries will certainly involve the equations of motion for the fluid itself. ~

For the sake of simplicity, we will focus on the case where the boundary of \( \Omega(t) \) consists of two rigid bodies with one moving relative to the other. We
identify two subregions within our domain, $\Omega_1$ and $\Omega_2(t)$, such that

$$\begin{cases}
\Omega(t) = \Omega_1 \cup \Omega_2(t) \\
\Gamma = \Omega_1 \cap (\Omega_2(t))
\end{cases}$$

where $\Gamma$ is not a function of time, i.e. the interface is rotationally or translationally invariant. Figure 10 depicts such a domain, a two-dimensional model of a mixing chamber, where an impeller is rotating with speed $\omega$ about its center and inside a square box which is stationary. Another such example would be the motion of a rotor relative to a stator blade in a hydro-turbine. Note in Figure 10 that the non-rotational invariance of the chamber renders inapplicable the possibility of describing the problem in a non-inertial reference frame based on the impeller.

By letting $\Omega_2(t)$ evolve with the translating rigid body, it becomes clear that the grid in $\Omega_2(t)$ is a function of time. This grid motion allows for a versatile description of the fluid domain and is known as the arbitrary-Lagrangian-Eulerian formulation [16,32,33]. The test functions $v$, against which we test the residual of equation (3.1), follow the mesh in $\Omega_2(t)$, and thus their substantial time derivative remains zero; that is,

$$v_{,t} + w_j v_{,j} = 0 \text{ in } \Omega_2(t) \tag{3.4}$$

where temporal and spatial derivatives are expressed in indicial notation and $w_j$ are the two components of the mesh velocity vector $W$. We then write the weak form of the equation (3.1)

$$\int_{\Omega(t)} v u_{,t} dA = \int_{\Omega(t)} v u_{,ij} dA \tag{3.5}$$

and rewrite the left-hand side as

$$\int_{\Omega(t)} v u_{,t} dA = \int_{\Omega(t)} [(uv)_{,t} - uv_{,t}] dA. \tag{3.6}$$

42
Figure 10: Model of a mixing process. Subdomain $\Omega_2(t)$ follows the mixer plate which rotates with angular speed $\omega$. Subdomain $\Omega_1$ is attached to the mixing chamber and remains fixed for all time. The sliding interface is denoted as $\Gamma = \overline{\Omega_1} \cap \overline{\Omega_2(t)}$. 

We make use of Liebnitz’s rule to express the first term in the right-hand side of (3.6) as

\[
\int_{\Omega(t)} (uv)_t dA = \frac{d}{dt} \int_{\Omega(t)} uv dA - \int_{\partial \Omega_1} uw_j n_j ds - \int_{\partial \Omega_2(t)} uvw_j n_j ds .
\] (3.7)

The line integrals in (3.7) vanish since the test functions are zero on the domain boundaries of \( \Omega \), and the term \( w_j n_j \) is zero along the interface \( \Gamma \). By introducing the kinematic condition (3.4) for the test functions in the moving subdomain, we express the second term of (3.6) as

\[
\int_{\Omega(t)} uv_t dA = -\int_{\Omega_2(t)} uw_j v^j dA .
\] (3.8)

After performing an integration by parts, we arrive at the final form for the variational statement of (3.1), which is

\[
\frac{d}{dt} \int_{\Omega(t)} uv dA - \int_{\Omega_2(t)} v(uw_j)_t dA + \int_{\Omega(t)} u_j v^j dA = 0 \quad \forall v \in H^1_0(\Omega(t)) .
\] (3.9)

Note that the second term in (3.9) is due to grid evolution and accounts for “compressibility” of the mesh as well as mesh advection. In the current thesis, we will consider only divergence-free mesh velocity fields, in which case, equation (3.9) can be written as

\[
\frac{d}{dt} \int_{\Omega(t)} uv dA - \int_{\Omega_2(t)} vu_j w^j dA + \int_{\Omega(t)} u_j v^j dA = 0 \quad \forall v \in H^1_0(\Omega(t))
\] (3.10)

with the boundary and initial conditions given in (3.2,3.3), respectively.

**Remark 3.2:** The derivation of the weak form (3.10) for the heat equation can proceed alternatively by replacing the partial time derivative of equation (3.1) with the substantial time derivative less the advection term due to grid velocity. That formulation allows for discretization of the problem by finite differences.
as well, with an interpolation procedure employed along the interface where gridpoints do not align. □

Although sliding meshes allow us to efficiently track the evolving geometry, it is obvious that the grid points between $\Omega_1$ and $\Omega_2(t)$ will not be aligned along the interface $\Gamma$. It is as regards $\Gamma$ that the analysis tools developed in the previous chapter will become of use; the problem can simply be viewed as a time-series of non-conforming discretizations. We will now proceed with the spectral element implementation of (3.10).

### 3.2 Discretization

The spectral element discretization of (3.10) corresponds to domain decomposition of the domain's subregions (here, two) into disjoint quadrilateral elements

\[ \Omega_1 = \bigcup_{k=1}^{K_1} \Omega^k \]  \hspace{1cm} (3.11)

\[ \Omega_2(t) = \bigcup_{k=K_1+1}^{K} \Omega^k(t) \]  \hspace{1cm} (3.12)

\[ \Omega^k \cap \Omega^l = 0 \quad k \neq l, \quad 1 \leq k, l \leq K. \]  \hspace{1cm} (3.13)

In the case of simple translation or rotation of $\Omega_2(t)$, the shape of the elements is constant. Furthermore, the decomposition is, per force, geometrically nonconforming at almost all time steps. As a consequence, the discretization of problem (3.10) involves the use of nonconforming discrete spaces of approximation $W_h$ and $X_h$, defined in Chapter 2. Replacing the integrals of (3.10) with tensor product Gauss-Lobatto quadrature $\times$ Gauss-Lobatto quadrature in $(r_1, r_2)$, the
discrete problem is as follows: Find \( u(x,t) \in X_h \) such that

\[
\frac{d}{dt} (v_h, u_h)_{h, GL} - (v_h, V \cdot \nabla u_h)_{h, GL|_{t_0}} = - (\nabla v_h, \nabla u_h)_{h, GL} \quad \forall v_h \in X_h \quad (3.14)
\]

where the testfunctions \( v_h \) are always evaluated at the current time step.

To implement (3.14) we express, as for the Poisson equation (2.1), the discrete solution \( u_h \), test function \( v_h \), and coordinates \( x(r,s) \) and \( y(r,s) \) in terms of \( N^{th} \)-order Lagrangian interpolants through the Gauss-Lobatto Legendre points. The elemental Laplacian operator in (3.14) is given in (2.51) while the elemental convection operator of (3.14) takes the form

\[
(v_h, \mathbf{W} \cdot \nabla u)^k = \sum_{i=0}^{N} \sum_{j=0}^{N} \rho_i \rho_j \left\{ v_{ij}^4 g_{ij}^4 (D_{i} u_{qj}^k) + v_{ij}^5 g_{ij}^5 (D_{j} u_{iq}^k) \right\} \quad (3.15)
\]

where

\[
g_{ij}^4 = w_{1,ij} (D_{j} y_{iq}^k) - w_{2,ij} (D_{j} x_{iq}^k) \quad (3.16)
\]

\[
g_{ij}^5 = -w_{1,ij} (D_{i} y_{pqj}^k) + w_{2,ij} (D_{i} x_{pqj}^k) \quad (3.17)
\]

and \( D_{ij} \) is given in (2.50). By choosing \( v_{ij} \) to be 1 at a single global collocation point and denoting the elemental convection operator as \( C^k \), we obtain

\[
C^k u^k = \rho_i \rho_j \left[ g_{ij}^4 (D_{i} u_{qj}^k) + g_{ij}^5 (D_{j} u_{iq}^k) \right] . \quad (3.18)
\]

As in Chapter 2, we will use the notation \( blk(C^k) \) to refer to all elemental convection operators, and \( C \) to denote the global convection operator which includes the direct-stiffness and projection transformations (i.e. \( C = Q^T blk(C^k) Q \)).

Remark 3.3: The appearance of a convective term in the sliding mesh formulation should not be considered a drawback since a term of the same order appears in the Navier-Stokes equations due to actual fluid convection. □
In deriving the fully-discrete equations, the presence of the advective terms in (3.10) gives rise to many numerical problems not present in the Poisson equation of (2.8). The advective terms render the equations nonsymmetric and, in the case of the Navier-Stokes equations, nonlinear; two facts that significantly complicate any solution procedure. Our semi-implicit algorithm for equation (2.8) proceeds by an Euler backward difference [13,50] for the temporal derivative, implicit treatment of the diffusion term, and explicit treatment of the advection term,

\[
\frac{1}{\Delta t} Q^{n+1} blk(B^{k_{n+1}}) Q^{n+1} u^{n+1} + Q^{n+1} blk(A^{k_{n+1}}) Q^{n+1} u^{n+1} =
\]

\[
\frac{1}{\Delta t} Q^{n+1} blk(B^{k_{n+1}}) Q^{n} u^{n} + \sum_{q=0}^{2} \alpha_{q} Q^{n+1} blk(B^{k_{n-q}}) Q^{n-q} u^{n-q}
\]

(3.19)

where superscript \( n \) denotes the timestep level, and \( \alpha_{q} \) are the coefficients for the third-order Adams-Bashforth scheme: \( \alpha_{0} = 23/12; \alpha_{1} = -16/12; \) and \( \alpha_{2} = 5/12. \) The implicit/diffusion - explicit/advection splitting is motivated by the fact that the second-order operator results in the most stringent timestep restriction; as the matrix equations associated with the diffusion operator can be efficiently inverted (Section 2.5), it is clear that these terms should be treated implicitly. On the other hand, the first-order advective terms are less restrictive in terms of timestep and more difficult to invert, and are, therefore, more conveniently treated explicitly. We choose third-order Adams-Bashforth [50] for its relatively large stability region [23].

Remark 3.4: Note in equation (3.19) that the direct stiffness \( Q^T \) is always evaluated at time level \((n + 1)\) since we test the discrete variational form in terms of \( v^{n+1}_h. \) The projection \( Q \) however, of the solution at different time levels, is evaluated with the testfunctions \( v_h \) at level \( n \) since it is they that provide an "image" of \( v^{n+1}_h \) for that particular time level. □

Numerical stability of the scheme is shown in [4] provided, of course, that
the time step is small enough. This time restriction is closely connected to the
eigenvalue spectrum of the explicitly treated spatial operator, in this case the
convection operator [54]. For convection problems the stability criterion is the
well-known Courant condition [50] which essentially requires that the domain of
dependence of the discrete convection includes the one of the physical convection,
i.e.
\[ \Delta t \leq \left( \frac{\Delta x}{U} \right)_p \] (3.20)
where \( U \) is the characteristic velocity at some collocation point \( p \), and \( \Delta x \) is
a typical spacing between grid points which, for spectral discretizations, scales
like \( O(1/KN^2) \).

We have verified the correctness of the formulation by solving the heat equa-
tion on the domain of Figure 11, \( \Omega = [-1,1]^2 \), where the exact solution is given
by \( u(x,y,t) = \exp(-2\pi^2 t) \sin(\pi x) \sin(\pi y) \). We have carried out the simulation
to a fixed time \( (T_f \sim .05) \) by varying \( N \) for a fixed (small) time step. Figure
12 illustrates the approximation error for two rotation rates a) \( \omega = 100 \) and b) \( \omega = 0 \) (i.e. non-conforming vs conforming). The agreement between the two is
another evidence of the optimality of the nonconforming formulation as regards
the consistency error. Next, by keeping fixed the final time to \( T_f = .05 \), the
order of the polynomial to \( N = 12 \), and the rotation rate to \( \omega = 1 \), we have
carried out simulations for a varying time step \( \Delta t \). Figure 13 illustrates the
convergence of temporal errors for a first and second order time discretization;
indeed, the slopes of the two lines are one and two, respectively. It is worthwhile
to note that the particular choice of the mortars along the moving interface does
not influence the magnitude of the discretization errors. Mortar strips can be
associated either with the fixed or moving subdomains.
Figure 11: Spectral element discretization for solving the two-dimensional, diffusion equation with exact solution given by $u(x,y,t) = \exp(-2\pi^2 t) \sin(\pi x) \sin(\pi y)$. The elements inside the circle rotate with angular speed $\omega$. 
Figure 12: Plot of discretization error $||u - u_h||_{H^1}$ as a function of polynomial order $N$ for the heat equation on the domain of Figure 11. The exact solution is given by $u(x, y, t) = \exp(-2\pi^2t)\sin(\pi x)\sin(\pi y)$. The results compare two rotation rates a) $\omega = 100$ (o), and b) $\omega = 0$ (* (i.e. conforming vs. non-conforming). The simulation is carried out to a final time of $T_f = .05$ with $\Delta_t = .00002$. 
Figure 13: Plot of discretization error $\|u - u_h\|_{H^1}$ as a function of time step $\Delta t$ for the heat equation on the domain of Figure 11. The exact solution is given by $u(x, y, t) = \exp(-2\pi^2 t) \sin(\pi x) \sin(\pi y)$ and the simulation is carried out to a final time of $T_f = .05$ with $N = 12$, and $\omega = 1$. The results indicate that the schemes are first (o) and second (*) order in time, respectively.
3.3 Navier-Stokes Solution Algorithm

Analysis of many viscous, incompressible fluid flow problems involve the solution to the Navier-Stokes and continuity equations:

\[
\rho \left( \frac{\partial \bar{u}}{\partial t} + \bar{u} \cdot \nabla \bar{u} \right) = -\nabla p + \mu \nabla^2 \bar{u} - \vec{f} \quad \text{in } \Omega \tag{3.21}
\]

\[
\nabla \cdot \bar{u} = 0 \tag{3.22}
\]

\[
\bar{u} = \bar{u}_b \quad \text{on } \partial \Omega. \tag{3.23}
\]

where \( \rho \) is density, \( \mu \) is viscosity, and \( \vec{f} \) is a body force. Many different approaches exist for discretization of equations (3.21-3.23); several of these schemes are discussed in [48]. In this thesis, the time-advancement approach for solution of the Navier-Stokes equation is the well known fractional step method [47] in which the same approximation space is used for the both pressure and velocity. It is efficient since it involves matrix inversions of the symmetric positive-definite Laplacian and Helmholtz operators and sufficiently accurate for high Reynolds numbers flows. The error of the scheme is of order \( O(\nu \Delta t) \) [47] at best. Further work is needed to incorporate more accurate (and more complex) algorithms.

The splitting formulation is comprised of three computational steps. We start with explicit treatment of the advection and forcing terms to compute \( \hat{\bar{u}} \):

\[
\frac{\hat{\bar{u}} - \bar{u}^n}{\Delta t} = \sum_{q=0}^{2} \alpha_q \left[ -(\bar{u} \cdot \nabla)\bar{u} + (\bar{u} \cdot \nabla)\bar{u} + \vec{f} \right]^{n-q}. \tag{3.24}
\]

This is followed by the pressure step:

\[
\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot \hat{\bar{u}} \quad \text{in } \Omega(t) \tag{3.25}
\]

\[
\frac{\hat{\bar{u}} - \bar{u}}{\Delta t} = -\nabla p \quad \text{in } \Omega(t) \tag{3.26}
\]

\[
\nabla p \cdot \hat{n} = \frac{1}{\Delta t} \hat{\bar{u}} \cdot \hat{n} \tag{3.27}
\]

52
and, finally, the viscous correction:

\[
\frac{\bar{u}^{n+1} - \tilde{\bar{u}}}{\Delta t} = \frac{1}{Re} \nabla^2 \bar{u} \quad \text{(3.28)}
\]

\[
\bar{u}^{n+1} = \bar{u}_0(t) \quad \text{on} \partial \Omega(t). \quad \text{(3.29)}
\]

We note that equations (3.25) and (3.28) can be solved based on the constructs developed for (2.8) and (3.10), respectively. The elliptic solve for the pressure (3.25, 3.27) is the most time consuming task in the whole algorithm. Care must be exercised in the formulation to ensure proper treatment of the boundary conditions. If the velocity is specified everywhere on the domain boundary, \( \partial \Omega \), the pressure equation is a pure Neumann problem, i.e. the pressure can be determined up to an arbitrary constant which corresponds to the mean pressure. If an outflow condition is specified for the normal component of the velocity, then Dirichlet boundary conditions (here, zero) are specified for the pressure.

We close this Chapter by illustrating once again, the exponential convergence of the nonconforming spectral element discretization. Moin and Kim give an analytical solution to the time-dependent Navier-Stokes equations in an infinite domain [39]:

\[
u(x, y, t) = -\cos(\pi x) \sin(\pi y) \exp^{-t} \quad \text{(3.30)}
\]

\[
v(x, y, t) = -\sin(\pi x) \cos(\pi y) \exp^{-t} \quad \text{(3.31)}
\]

\[
p(x, y, t) = -[\cos(2\pi x) + \cos(2\pi y)] \exp^{-2t} / 4 \quad \text{(3.32)}
\]

The computational domain is that of Figure 11 broken up into \( K = 9 \) elements; the initial conditions correspond to the exact velocity (3.30, 3.31) at \( t = 0 \). First, we performed error analysis for two mesh rotation rates (\( \omega = 1 \), and 0) by carrying the simulation to a fixed time \( T = 1 \) with different \( N \) and keeping \( \Delta t \) very small so as to insure that temporal errors are small; the results of Figure
14 indicate exponential convergence for both nonconforming and conforming discretizations. Next, we simulated until a fixed time $T = 1$, using different time steps $\Delta t$ while keeping $N$ large enough so that spatial errors are negligible. Figure 15 shows that the scheme is first-order in time.

In general, the greatest challenge posed in discretization of the Navier-Stokes equations is to accurately resolve all the existing scales of the flow. This is in fact the main limitation of direct simulation of fluid flow. Increased resolution implies smaller spacing between adjacent grid points, which, in turn, restricts the maximum allowable time step (3.20) because of the explicitly treated convection operators. Typically, given the constraint of reasonable computer time, the nonconforming spectral element method presented here is capable of accurately resolving complex geometry flows in the laminar regime and the onset of transition; this, corresponds to a range of Reynolds numbers from $O(1)$ to $O(1000)$. 
Figure 14: Plot of discretization error $\|\overline{u} - \overline{u}_h\|_{H^1}$ as a function of polynomial order $N$ for the incompressible Navier-Stokes equations on the domain of Figure 13. The exact solution is given by equations (3.30-3.33). The results compare two rotation rates a) $\omega = 1$ (o), and b) $\omega = 0$ (*) (i.e. conforming vs. non-conforming). The simulation is carried out to a final time of $T_f = 1$ with $\Delta t = .0002$. 

55
Figure 15: Plot of discretization error $\|\bar{u} - \bar{u}_h\|_{H^1}$ as a function of time step $\Delta t$ for the incompressible Navier-Stokes equations on the domain of Figure 13. The exact solution is given by equations (3.30-3.33). The simulation is carried out to a final time of $T_f = 1$ with $N = 12$, and $\omega = 1$. The results indicate that the scheme is first order in time.
Chapter 4

Implementation

The final issue of concern to us regarding the successful development of non-conforming discretizations involves the extent to which the method can be easily implemented in a simple and general algorithmic model that keeps intact the flexibility of the decomposition, preserves the structure of the high order elemental blocks, and supports the required processing load of large-scale fluid flow simulations. Conceptual and actual algorithmic simplification involves the breakdown of the computational procedures into small, easily-replicated functional units while efficiency of execution is ensured by preserving locality and concurrency in the computations. The latter can lead to substantial parallel and vector efficiencies [5,21].

In this Chapter we will examine in detail the major computational tasks that the discretization imposes on the underlying algorithm. These include procedures for the detection of the skeleton structure, establishment of complex elemental interconnections with mortars and vertices, and, dynamic reconfiguration of the domain topology in the case of sliding meshes. During the course of the discussion, we will provide "pseudo-code" descriptions of the decomposition's key algorithms and data structures in a typical high-level programming language, say C [37]; our aim is to emphasize the intuitive treatment of complex nonconforming discretizations by means of symbolic instruction sequences that
accurately illustrate the various concepts but which are, at the same time, free of the actual details of the implementation.

4.1 Skeleton Composition

4.1.1 Detection

The initial step of the execution process involves the loading of information regarding the geometry definition and mesh generation of the problem domain. In addition, the boundary conditions are specified along with relevant problem and control parameters. This information is generated during what is customarily known as the “preprocessing” stage of the simulation and is considered the point of departure for the solution algorithm.

Initially, the skeleton structure $S$ of the decomposition is established by means of an exhaustive search over all spectral elements. The search, which consists of comparing the edges and corners of all the elements in the domain with all the others’, results in a set of non-overlapping mortars with associated elemental edges and a list of unique virtual/real vertices with coinciding elemental corners. These relationships are established either via the geometric proximity of edges and corners, or by virtue of an imposed periodic boundary condition.

**Geometric Proximity:** The search for mortar strips along elemental interfaces proceeds by attempting to establish a non-trivially overlapping segment spanning two overlapping elemental edges; it is completed when all such overlapping
pairs are considered. For decompositions in $\mathbb{R}^2$, there are three possible configurations for potential alignment of two edges, shown in Figure 16. The resulting mortars have been named after the nature of this alignment to conforming, enclosed, and overlapping mortars. A mortar is conforming in the case where the two elemental edges coincide and the discretization follows the traditional spectral element decomposition [40,54]. In the case where an edge is enclosed by another (Figure 16b), the mortar is chosen (rather arbitrarily) to coincide with the smaller edge. An illustration of this case is shown in Figure 17b for the simple domain of Figure 17a; the mortars are aligned with the smaller edges of elements 2 and 3, respectively. The alternative approach complying with definition (2.14) is to chose the mortar along the longer edge, as shown in Figure 17c, and treat the centered vertex as a virtual vertex. Finally, the overlapping mortar, corresponding to a configuration such as in Figure 16c, could also be associated with any of its two edges, as long as it does not coincide with a previously defined mortar. In either case, one of the edges' endpoints will fall inside the overlapping mortar and be considered a virtual vertex. Figure 18 shows two possible mortar structures for the model problem of Figure 1a. The middle vertical interface is spanned by mortars associated with edges of elements 1 and 3 in the first case (18a), and 2 and 4 in the second case (18b); virtual vertices are associated with the corners of elements 1 and 3 in the first case, and 2 and 4 in the second.

As regards the set of unique vertices and their corresponding coincident elemental corners, they are all identified by their geometric proximity. Absolute coincidence can, of course, never be established in a finite-precision machine; the separating distance between corresponding pairs of nodes is compared against a carefully defined tolerance level $\epsilon$ which is well-above the computer's precision.
Figure 16: Illustrations of the three possible alignments between a pair of edges $\Gamma^{k_1, l_1}$ and $\Gamma^{k_2, l_2}$. In case (a), the edges and the resulting mortar are conforming; in case (b), the mortar is associated with the smaller edge $\Gamma^{k_2, l_2}$; in case (c), the edges overlap and the mortar can be chosen with either edge as long it does not coincide with a previously defined mortar.
Figure 17: Nonconforming domain decomposition (a) and two possible skeleton structures (b) and (c). The middle vertical interface can be spanned by two mortars associated with the edges of elements $\Omega_1$ and $\Omega_2$ (b), or by a single mortar along the edge of element $\Omega_1$ (c).
Figure 18: Two possible skeleton structures for the domain decomposition of Figure 1(a). The middle vertical interface can be spanned by two mortars associated with the edges of elements $\Omega_2$ and $\Omega_4$ (a), or by two other mortars along the edges of elements $\Omega_1$ and $\Omega_3$ (b). The virtual vertex coincides with corners of elements $\Omega_1$ and $\Omega_3$ in case (a); $\Omega_2$ and $\Omega_4$ in case (b).
limitations but well-below the required accuracy. In other words, we accomo-
date round-off errors in the precision of geometric coordinates by placing small
circles of radii \( \varepsilon \) around each nodal point.

The limitations of finite-precision arithmetic appear also in the mortar de-
composition since the tests for conformity, enclosure, and overlap must be sat-
isfied up to the tolerance of the computation (Figure 19). For the conformity
check, the tolerance is included in a way that the test is succesful if the two
edges' endpoints fall within the circle of radius \( \varepsilon \), centered on each node (Figure
19a). For the case of an enclosure mortar, the test checks for elemental end-
points that either both fall within the opposite edge by an amount more than \( \varepsilon \),
or exactly one of them is close (from within) to an endpoint from the other edge
by a distance less than \( \varepsilon \) (Figure 19b). Similarly, the overlapping test checks for
nodes that are located one inside and one outside of the opposite edge but both
by an amount more than \( \varepsilon \) (19c).

**Periodicity** A periodic boundary condition is the mathematical statement that
the solution along two parts of the domain boundary is identical for all times;
the separating distance between the boundaries, denoted as \( L_P \), is referenced as
the periodic length. In terms of the skeleton structure, periodicity requires that
the mortars and vertices along these parts of the boundary must coincide. We
achieve this by simply translating the elements with periodic edges by a distance
\( L_P \), and proceed with the regular tests for conformity, enclosure, and overlap
between edges, and coincidence among corners. The approach is illustrated
in Figure 20(b) for the decomposition of Figure 20(a). Here, the domain is
extended by additional elements, as indeed would be the periodic case, and the
mortar/vertex detection search proceeds with this new configuration. The
Figure 19: Conforming (a), enclosure (b), and overlapping (c) mortar detections in finite-precision arithmetic. Round-off errors in the computation are accommodated by small circles of \( \epsilon \) radii around the endpoints of the particular edge-pairs considered.
Figure 20: Domain decomposition (a) with periodic boundary conditions in the x-direction. Detection of mortars and vertices proceeds along the solid lines of the modified domain (b) where elements are introduced on either side of the periodic boundaries.
resulting skeleton structure still spans the original decomposition, i.e. the solid lines of Figure 20b. However, when each one of the elements is being considered, it is searched in the context of the modified domain; that way, element-pairs (1,3) and (2,4) will correctly be sharing two edges each. Another approach to periodicity is to perform all arithmetic mod$L_P$.

The same principle of introducing “phantom” elements in the decomposition, is applied in the particular case of a sliding mesh where elements are translating in time and wrapped around by periodicity, as in Figure 21(a). During each time step, the translating elements that cross the periodic boundary are moved appropriately to its opposite side. The search, again, proceeds by allocating mortars and vertices based on geometric proximity among elements in the modified domain decomposition. In Figure 21(b), for instance, where all elements are duplicated since they all lie on periodic boundaries, each of the original elements is processed as part of the modified domain. In the case where element 3 has moved completely outside the original boundaries, it is brought back in, and the procedure is applied anew.

***

We believe that the above constitutes a consistent way towards mortar construction of the skeleton decomposition $S$. The non-uniqueness of the final mortar structure should not be considered as a weak point of the formulation but rather as an attribute of its flexibility; after all, a particular mortar decomposition of the skeleton can only produce errors in the computed solution that are a constant factor worse than any other decomposition. A good mortar allocation for $S$ should be based on some a priori knowledge of the computed solution’s reso-
Figure 21: Sliding-mesh domain decomposition (a); element $\Omega_3$ translates with velocity $V$ and periodic boundary conditions are applied in the $x$-direction. Modified domain (b) at a particular instant in time, where all elements are duplicated by virtue of the periodic boundary condition. Skeleton detection proceeds along the solid lines of the modified domain (b).
solution requirements; it should involve small sized mortars to refine regions with steep gradients and large sized mortars to discretize areas where the solution does not exhibit sharp discontinuities.

As far as the precision of the arithmetic is concerned, the tolerance $\epsilon$ acts as a safety net for small errors in the geometry; treating these errors consistently, avoids the assignment of mortars in areas spanned by other mortars. Despite its drawbacks, this geometric approach (and its periodic extension) is the only way of establishing elemental interconnections in the case of sliding geometries where such relationships evolve continuously in time. It is only for static elemental configurations that a symbolic approach could alleviate the uncertainty of the machine precision by describing elemental proximity in integer or character format.

4.1.2 Representation

The skeleton detection forms the basis for subsequent management of the field data since it provides in effect the required data exchanges among spectral elements. The decomposition, as a whole, is represented as a list of sublists where each one of the sublists identifies a unique mortar-edge, vertex-corner, or virtual vertex-corner combination.

The mortar sublist holds all the information associated with a mortar-edge combination, as shown in Figure 2, and can be conveniently grouped together in the following compound data type "mortar_list"

```
    structure mortar_list {
```
By virtue of the skeleton search, all elemental edge lists related to the same mortar configuration (unique \textit{mortar\_number}) appear one after the other. In the special case of an overlap mortar the logical variable \textit{tag} in the mortar list is activated to indicate the existence of a virtual vertex related to that particular mortar strip.

The virtual vertex sublist has its own associated data structure \textit{"v\_vertex\_list"}

```c
struct v_vertex_list {
    int ielement;
    int icorner;
    float mortar_position;
};
```

where the identity of the associated elemental corner, and its geometric location
(in normalized $[-1,1]$ coordinates) with respect to the mortar, are stored.

Each occurrence of coincident corners is also referenced by a particular data structure "vertex_list"

```c
structure vertex_list {
    int vertex_number ;
    int ielement ;
    int icorner ;
    int jelement ;
    int jcorner ;
} ;
```

with elemental corners matched in pairs. Again, corners related to the same vertex appear in sequence and identified by a common vertex_number.

We will reference the decomposition $S$, as a whole, by the variable *skeleton which addresses the first member of the group of sublists. Symbolically, this member can be described as follows:

```c
structure member {
    structure mortar_list  *mortar_group ;
    structure vertex_list  *vertex_group ;
    structure v_vertex_list  *v_vertex_group ;
    structure member  *next_member ;
} *skeleton ;
```

The symbol * in our definition indicates that the particular variable contains
the address to another variable as opposed to the value of the variable itself; this way, a large group of data can be referenced by a single address to the first element in the group. In the present case, where a particular entry in $S$ may be of one of three sublist types, each pointer variable of data type "member" is the address to a group of three variables of type "mortar.list", "vertex.list", and "v.vertex.list". At any one time, only one these variables addresses a non-trivial sublist, while the other two are set by default to the constant value NULL. In addition, since the length of the decomposition list $S$ is not known a priori, each variable of type "member" points to the next, by means of *next_member. The first member, which we have called *skeleton, serves as the port of entry for the whole mortar/vertex decomposition while the last one signals the end by containing the value NULL. In general, this self-referential nature of the data type declaration "member" is advantageous in that it allows for the decomposition to be cast together in arbitrarily long lists and be constructed dynamically in the course of the actual detection algorithm.

4.2 Field Data Structure

The apparent inhomogeneities in the decomposition, where mortars and vertices may involve a widely differing number of associated elemental edges and corners, must be accounted for in the design of the overall field data structure and reflected in its operation. Across-the-board uniformity, a distinct property of classical conforming discretizations, is no longer present in the mortar-element method where elements vary considerably in size, topology, and, hence, complexity. In the current approach, the data is first grouped together in entities that represent the fundamental blocks of the discretization; that is, mortars, vertices,
and spectral elements with their edges and corners. These compound units are
put together into a larger data set, built as a network of inter-connections which
replicate the geometric interrelations of the decomposition. During actual imple-
mentation in, say C, the building blocks are again, user-defined data structures,
while the inter-connections of the assembled data set are pointers indicating the
memory addresses of the decomposition’s skeleton and mesh.

The mortars and vertices play a crucial role in the layout of the data structure
since they will be used as common storage locations where elemental edges and
corners are addressed. Vertex structures consist of the grouping of the data field
value, multiplicity, and coordinate location of a particular vertex in the domain:

```
structure skeleton_vertex {
    int    number ;
    int    on_off ;
    int    mult ;
    float  data ;
    float  x_coordinates ;
    float  y_coordinates ;
} ;
```

The variable `mult` reflects the number of coincident elemental corners, and the
logical `on_off` acts as a switch which controls access to the data structure itself.
Its functionality will become apparent during evaluation of the direct stiffness
and inner product algorithms, described later in this chapter.

Constructed in a similar fashion, the mortar structure consists of the follow-
ing variables:
Here, \textit{mult} is the number of coincident elemental edges while the variables \textit{*x.coordinates}, \textit{*y.coordinates}, and \textit{*data} are pointing to the first location of vectors of length \textit{size} that represent all the discretization points along the mortar. Furthermore, the "mortar" data type points to the two "vertex" structures representing the endpoints associated with each mortar. In the case of overlapping mortars, the variable \textit{*virtual.vertex} contains the address to the first, of potentially many, virtual vertex structures shown below:

```c
structure skeleton_v_vertex {
    int mult ;
    int on_off ;
    float *interpolants ;
    structure skeleton_v_vertex *next_v_vertex ;
};
```
The variable \*interpolants points to a vector containing the values of the Lagrangian interpolants evaluated at the location of the virtual vertex, while \*vertex is linked to the vertex structure itself. Similarly to mortars and vertices, \textit{mult} and \textit{on\_off} are used in vector reduction and direct stiffness operations with \textit{mult} now being essentially the same as the multiplicity of the mortar as opposed to that of the corresponding vertex.

To provide easy access to the skeleton structures from within spectral elements, we have constructed a generic data type \texttt{"skeleton\_link"} which can hold pointers to \texttt{"skeleton\_mortar"}, \texttt{"skeleton\_vertex"}, and similar \texttt{"skeleton\_link"} data structures and can be described as follows:

\begin{verbatim}
structure skeleton_link {
    structure skeleton_vertex *vertex ;
    structure skeleton_mortar *mortar ;
    int direction ;
    float *Q_matrix ;
    structure skeleton_link *next_link ;
}
\end{verbatim}

The variables \texttt{direction} and \*\texttt{Q\_matrix} are related to particular pairs of mortar-edge combinations: \texttt{direction} is a logical variable being true or false depending on whether the grid values on the mortar and edge are stored in the same or opposite order; \*\texttt{Q\_matrix} is the pointer to the first entry of the projection operator $[Q]$. Again, the self-referential nature of the \texttt{"skeleton\_link"} allows in the case of edges, which can be associated with more than one mortars, for \texttt{"skeleton\_mortar"} structures to be accessed locally from that edge, one after
Unlike their counterparts of the skeleton decomposition, edges and corners do not hold their own field values but contain integers that can uniquely identify and retrieve these values from the elemental array blocks. In particular, the edge structure is described as follows:

```c
structure elemental_edge {
    int id;
    int start;
    int skip;
    structure skeleton_link *link;
    structure elemental_edge *next_edge;
};
```

Integers `start` and `skip` are the variables that can locate the edge data out of a two-dimensional array of computed values. The variable `*link` contains the memory address to the corresponding mortar; in the case of more than one mortars associated with that edge, sequential "link" variables provide appropriate access to them. The projection matrix $[Q]$, as mentioned before, can be retrieved locally and referenced by the variable `*Q_matrix`.

Similarly to the edges, the corners are defined in the following manner:

```c
structure elemental_corner {
    int id;
    int start;
    structure skeleton_link *link;
};
```
where now the variable *link leads to the corner’s corresponding vertex.

Finally, the elemental data structures turn out to be, by far, the most complex since they hold all information relevant to an individual element. They can be illustrated in the following table

```
structure element {
  int id;
  int type;
  int polynomial_order;
  float *data;
  float *x_coordinates;
  float *y_coordinates;
  float *g1;
  float *g2;
  float *g3;
  float *g4;
  float *g5;
  float *weights;
  float *jacobian;
  struct skeleton_edge *first_edge;
  struct skeleton_corner *first_corner;
  struct element *next_element;
};
```
where, besides the spectral element number and type, each data structure contains the memory addresses to the first location of large array blocks (of size \textit{polynomial\_order}²) for storage of the data, grid coordinates, geometric factors, integration weights and Jacobian. The field and all other data necessary for evaluating the Laplacian and convection terms of (2.51,3.18) hold contiguous memory locations in order to ensure faster execution in computers with vector capabilities (a factor of 10 faster on a CRAY-2 supercomputer). In addition, each element contains the address to the first of its edge and corner structures. Finally, each variable of "element" type points to the next one; this way, all elements in the decomposition can be referenced in a sequential fashion.

The most distinguishing feature of the overall approach is its dataflow-like nature which provides the necessary paths and allows for evaluations of inter-element data dependencies by linking each data structure to its "geometrically neighboring" one. We will illustrate this concept by considering the model problem of Figure 22a where the domain is partitioned into two elements with the top one sliding in the x-direction by a velocity \( V \) and wrapping around by virtue of a periodic boundary condition. Figure 22b shows the skeleton structure at a particular instant in time where the vertices and mortars are alphabetized and local edges and corners are enumerated. The data structure is shown in Figure 23 where each one of the rectangles corresponds to a particular kind of sub-structure, labeled in accordance to Figure 22b. The arrows represent pointer variables of data type "skeleton\_link" that associate mortars and vertices with elemental edges and corners. For instance, edge 4' of element \( \Omega_1 \) is pointing to mortar g, which is also addressed by edge 3' of element \( \Omega_2 \); the coincidence of the two edges automatically ensures the correct sharing of edge values during the computation. Similarly, corners 1 and 2 of element \( \Omega_1 \) share the same vertex
Figure 22: Two-element decomposition (a) and associated skeleton structure (b) for a sliding-mesh example problem. The model in (a) consists of a top wall that is translating in the x-direction with velocity $V$, a fixed bottom wall, and periodic boundary conditions along the two vertical sides. Element $\Omega_2$ translates with velocity $V$ and wraps around, while $\Omega_1$ stays fixed. In (b), the mortar along the sliding interface is along edge $4'$ of element $\Omega_1$; vertex $c$ is a virtual vertex.
Figure 23: Illustration of the data structure for the example problem and skeleton decomposition of Figure 22. Elemental edge and corner sub-structures are enumerated while mortar and vertex sub-structures are alphabetized, all according to Figure 22(b). The links (arrows) between edge, mortar, corner, and vertex substructures are based on geometric proximity and interrelations. Each element, corner, and edge structure points to the next, or to NULL. The data field as a whole is referenced by a single pointer variable field which is of data type “element”.

79
due to the periodicity condition. Vertex c is also pointed, in addition to the elemental corners, by the mortar g via the intermediate virtual vertex structure c'. The multitude of coincident elemental edges and corners is reflected by the number of arrows leading to the shared mortars and vertices. Note also, that the first element, is linked to the second, and each one in turn, linked to its local edges and corners; therefore, only one pointer variable, in this case *field, is necessary to be stored explicitly since all other data and substructures are directly or indirectly connected to that.

The general properties of the nonconforming domain decomposition method, that is, numerically intensive computations at the local level, and irregular memory accesses and data exchange operations at the inter-element level, are represented in a simple manner by the partitioning of the data storage into locally-structured / globally unstructured blocks. As will be shown in the next section, the data management can be handled by small, yet general-purpose, routines. The local nature of the elemental objects condenses algorithmic development to the optimal treatment of simple, functional units that provide the high throughput rate for intensive floating-point computations and can execute largely independent of others when distributed onto individual processors of a parallel computer network. The communication among elements, reflected by the edge and corner structures, provides an intuitive way of representing the very complex topologies of nonconforming discretizations. Such a direct mapping between geometrical relationships and data dependencies can also be likened to the unrolling of the graph of, say, Figure 22b, into the underlying, tree-like computer architecture.
4.3 Memory Allocation

Storage for the computed solution and connections in the data structure are handled in two steps at the beginning of each simulation. The first one involves the creation of the elemental data structures along with the edge and corner substructures, while the second one establishes the mortars and vertices along with the links to elemental edges and corners.

To begin with, the first procedure accepts as input $K$, the total number of elements in the domain, and $N$, the order of the approximating polynomial which we assume constant for all elements. In what follows, we use the symbol "$->$" to refer to a particular member of a data structure. The routine, which returns the pointer to the first "element" data structure, can be described in the following sequence of instructions:

Algorithm Data Field Storage

\begin{verbatim}
input($K, N)$
for ($iel = 1, K$)
    $temp = make_storage_for_element_structure(iel, N);$  
    for ($iedge = 1, 4$)
        $edge = make_storage_for_edge_structure(iedge, N);$  
        if ($iedge$ equal 1) then
            $temp-> edge = edge;$
            $current_edge  = edge;$  
        else
            $current_edge-> next_edge = edge;$  
\end{verbatim}
current_edge = edge;

for (icorner = 1, 4)
    corner = make_storage_for_corner_structure(icorner, N);
    if (icorner equal 1) then
        temp->corner = corner;
        current_corner = corner;
    else
        current_corner->next_corner = corner;
        current_corner = corner;
    if (iel equal 1) then
        first_element = temp;
        current_element = first_element;
    else
        current_element->next_element = temp;
        current_element = temp;
return first_element;

In the above, the procedures prefixed by the key phrase “make_storage” are basically calls to system level routines requesting memory blocks of specific data type. The pointer variables temp, edge, and corner are used to store temporarily the allocated memory which is subsequently linked to the previously defined element, edge, and corner structures. Elemental edges and corners can be reached one after another from the particular element that they belong, as shown in Figure 23. In the final step, each “element” pointer variable is linked to the previous one and the first is returned as the mean to access the whole domain for the computed solution.
The second routine creates storage for mortars and vertices and at the same time connects them to the appropriate elemental edge and corner data structures. The procedure accepts as input the variable *skeleton which is the pointer to the list of mortar and vertex sublists, and *field, the pointer to the data structure generated by a call to Data Field Storage. The procedure for the data interconnections can be described in the following form:

Algorithm Skeleton Connections

input(*field, *skeleton)

list = skeleton ;
vertex_number = 0 ;
mortar_number = 0 ;
while (*list ≠ NULL)
  if (list-> vertex_group ≠ NULL) then
    v_group = list-> vertex_group ;
    ielement = v_group-> ielement ;
    icorner = v_group-> icorner ;
    jelement = v_group-> jelement ;
    jcorner = v_group-> jcorner ;
    if (v_group-> vertex_number > vertex_number) then
      vertex = make_storage_for_vertex_structure() ;
      vertex_number+ = 1 ;
      connect_vertex_to_corner(field, vertex, ielement, icorner) ;
    else
      vertex = find_vertex_structure(field, ielement, icorner) ;
      connect_vertex_to_corner(field, vertex, jelement, jcorner) ;

 83
if (list->mortargroup != NULL) then
    m_group = list->mortargroup;

if (m_group->mortarnumber > mortar_number) then
    mortar = make_storage_for_mortar_structure();
    mortar_number = mortar_number + 1;

for (i = 1, 2)
    mortar->endpoints[i] = link_vertices_to_mortar(field, mortar);
connect_mortar_to_edge(field, mortar, m_group);
make_Q_matrix(field, mortar, m_group);

if (list->vvertexgroup != NULL) then
    vv_group = list->vvertexgroup;
    vvertex = make_storage_for_virtual_vertex();
    link_vvertex_to_present Morton(mortar, vvertex);
    evaluate_interpolants(vv_group, mortar);
} (list = list->next_member);

The procedure consists basically, of a loop which goes through the list of the skeleton decomposition in order to generate the vertex and mortar structures. Storage is created dynamically at the occurrence of each new case by the "make" routines. The routines prefixed by "connect" are used to generate variables of type "skeleton_link" to connect mortars with edges, and vertices with corners. Similarly, the two "link" procedures relate, a) vertex structures to the endpoints[2] of every mortar, and b) virtual vertices to overlap mortars. In the case of edge-mortar pairs, the particular projection matrix [Q] is also computed and stored locally with the elemental edge.
The two algorithms mentioned above, can basically handle the dynamic memory allocation and management of the data field structures. They are relatively simple in context, but general enough to handle arbitrary, two-dimensional domain decompositions. During the simulation, related support variables needed for storing intermediate results are constructed in a similar fashion, with the only exception being, that arrays for, say, elemental geometric factors, or mortar-edge projection matrices, are not generated anew but related to the ones in the original data structure. In the case of sliding meshes where the topology is changing at every time step, the mortar and vertex connections from previous time steps are destroyed while new connections are constructed based on an updated list of vertex and mortar coincidences with edges and corners. However, the basic field arrays created by Field Data Storage remain intact.

4.4 Inter-element Communication

In this section we will examine the three operations of the conjugate gradient solution algorithm of Section 2.4 as regards the requirements for inter-element coordination and data exchange. They are a) the projection (via $Q$ transformation) of mortar-vertex data onto edges and corners, b) its transpose, direct stiffness accumulation ($Q^T$ interpolation) of elemental values onto mortars and vertices, and c) the inner product vector reduction of the algebraic degrees-of-freedom.

**Projection:** It consists of a sequence of two passes through the data structure, each one of them following a different path of pre-constructed pointers. First, the vertices are assigned to the mortar endpoints which are used along with the
mortar interior to evaluate the values at the virtual vertices; the latter, as discussed in Section 2.4.1, derive their value from evaluating the basis polynomials of the mortar function at their particular location. Next, the mortar values are projected onto elemental edges and accumulated in the case of more than one such projections on a particular edge; at the same time, vertex and virtual vertex values are assigned to the elemental corners. The projection algorithm can be summarized in the following instruction sequence:

Algorithm Projection

input(field)

\[ temp = field; \]

while (temp \( \neq \) NULL)

\[ edge = temp\rightarrow edge; \]

while (edge \( \neq \) NULL)

\[ link = edge\rightarrow link; \]

while (link\(\rightarrow\) mortar \( \neq \) NULL)

\[ mortar = link\rightarrow mortar; \]

\[ size = mortar\rightarrow size; \]

for (i = 0, 1)

\[ vertex = mortar\rightarrow endpoints[i]; \]

\[ mortar\rightarrow data[i \ast size] := vertex\rightarrow data; \]

\[ v\_vertex = mortar\rightarrow virtual\_vertex; \]

while (v\_vertex \( \neq \) NULL)

\[ vertex = v\_vertex\rightarrow vertex; \]

\[ vertex\rightarrow data := interpolate(mortar, v\_vertex); \]

\[ v\_vertex = v\_vertex\rightarrow next\_v\_vertex; \]
\[
link = link\rightarrow next\_link;
\]
\[
edge = edge\rightarrow next\_edge;
\]
\[
temp = temp\rightarrow next\_element;
\]
\[
temp = field;
\]
\[
\text{while (temp} \neq \text{NULL)}
\]
\[
corner = temp\rightarrow corner;
\]
\[
\text{while (corner} \neq \text{NULL)}
\]
\[
link = corner\rightarrow link;
\]
\[
temp\rightarrow data[corner\rightarrow start] = link\rightarrow vertex\rightarrow data;
\]
\[
corner = corner\rightarrow next\_corner;
\]
\[
\text{while (edge} \neq \text{NULL)}
\]
\[
link = edge\rightarrow link;
\]
\[
\text{while (link} \rightarrow \text{mortar} \neq \text{NULL)}
\]
\[
mortar = link\rightarrow mortar;
\]
\[
\text{accumulate\_mortars\_to\_edge(temp, edge, mortar)};
\]
\[
link = link\rightarrow next\_link;
\]
\[
edge = edge\rightarrow next\_edge;
\]
\[
temp = temp\rightarrow next\_element;
\]

The algorithm presented here is, in effect, the actual implementation of the diagrammatic representation of Figure 3 for the example problem in Figure 1a. Complying with the assumed precedence of incoming over outgoing arrows in the individual substructures of Figure 3, the aforementioned procedure ensures that mortar endpoints will be assigned before a mortar projection to an elemental edge occurs and that virtual vertices will be evaluated before they are descended to corners.
**Direct Stiffness:** Similarly to projection, the $Q^T$ transformation consists of two passes through the data structure. However, the roles of edges and mortars on the one hand, and corners, vertices and virtual vertices on the other, are reversed. During the first pass, elemental values are interpolated by $Q^T$ and accumulated on the mortars in case of multiple edge contributions. Corner values are also accumulated on common vertices. In the second pass, the influence of virtual vertices is accounted on the mortars, and mortar endpoints are accumulated on the vertices. The pseudo-coded procedure is as follows:

**Algorithm Direct Stiffness**

```plaintext
temp = field;
while (temp ≠ NULL)
    corner = temp->corner;
    while (corner ≠ NULL)
        link = corner->link;
        link->vertex->data+ = temp->data[corner->start];
        corner = corner->next_corner;
    while (edge ≠ NULL)
        link = edge->link;
        while (link->mortar ≠ NULL)
            mortar = link->mortar;
            accumulate_edges_to_mortars(temp, edge, mortar);
            link = link->next_link;
        edge = edge->next_edge;
    temp = temp->next_element;
```

88
temp = field;

while (temp \neq NULL)
    edge = temp->edge;
    while (edge \neq NULL)
        link = edge->link;
        while (link->mortar \neq NULL)
            mortar = link->mortar;
            onoff = mortar->onoff;
            mult = mortar->mult;
            size = mortar->size;
            vvertex = mortar->virtual_vertex;
            if (onoff equal mult) then
                while (vvertex \neq NULL)
                    value = vvertex->vertex->data;
                    accumulate_vvertex_values(mortar->data, value, vvertex);
                    vvertex = vvertex->next_vvertex;
            for (i = 0, 1)
                vertex->data += mortar->endpoints[i]->data;
            mortar->onoff = 1;
            if (mortar->onoff equal 0) then
                mortar->onoff = mutt;
            link = link->next_link;
            edge = edge->next_edge;
            temp = temp->next_element;

Analogous to projection, the Direct Stiffness algorithm represents the actual
evaluation of Figure 5 for the example problem of Figure 1a. During the second
pass in the data structure, we can notice the action of the mortar on/off switch which correctly allows the values of the mortar endpoints to be accumulated on vertices exactly once.

**Inner Product:** Between two data structures, the inner product sum calls for the accumulation of all values related to the algebraic degrees-of-freedom; that is, element interior, mortar interior, and vertex nodes. Since mortars and vertices are shared by many edges and corners, a "lock" mechanism is provided within each mortar and vertex structure to ensure that the values are accounted for just once. The lock mechanism is activated after the first time that a particular structure is reached; the remaining times a local counter is updated. During the last reach, when the counter is equal to the multiplicity of the structure, the lock is reset. Overall, the data structure is traversed twice; the first time to accumulate the result of the inner product, and the second time to subtract the value of the virtual vertices since they are not part of the independent degrees-of-freedom. The inner product procedure reads as follows:

**Algorithm Inner Product**

```
input(field)
sum = 0.;
temp = field;
while (temp ≠ NULL)
    corner = corner->structure;
    edge = edge->structure;
    N = temp->polynomial.order;
    for (i = 1, N - 1)
```
for \( j = 1, N - 1 \)
\[
\text{sum} += \text{element} \rightarrow \text{data}[i][j] ;
\]
while \((\text{edge} \neq \text{NULL})\)
\[
\text{link} = \text{edge} \rightarrow \text{link} ;
\]
while \((\text{link} \rightarrow \text{mortar} \neq \text{NULL})\)
\[
\text{mortar} = \text{link} \rightarrow \text{mortar} ;
\]
\[
\text{on}_off = \text{mortar} \rightarrow \text{on}_off ;
\]
\[
\text{mult} = \text{mortar} \rightarrow \text{mult} ;
\]
if \((\text{on}_off \text{ equal } \text{mult})\) then
\[
\text{for } (i = 1, N - 1)
\]
\[
\text{sum} += \text{mortar} \rightarrow \text{data}[i] ;
\]
\[
\text{mortar} \rightarrow \text{on}_off = 1 ;
\]
if \((\text{mortar} \rightarrow \text{on}_off \text{ equal } 0)\) then
\[
\text{mortar} \rightarrow \text{on}_off = \text{mult} ;
\]
\[
\text{link} = \text{link} \rightarrow \text{next}_link ;
\]
\[
\text{edge} = \text{edge} \rightarrow \text{next}_edge ;
\]
while \((\text{corner} \neq \text{NULL})\)
\[
\text{vertex} = \text{corner} \rightarrow \text{vertex} ;
\]
\[
\text{on}_off = \text{vertex} \rightarrow \text{on}_off ;
\]
\[
\text{mult} = \text{vertex} \rightarrow \text{mult} ;
\]
if \((\text{on}_off \text{ equal } \text{mult})\) then
\[
\text{sum} += \text{vertex} \rightarrow \text{data} ;
\]
\[
\text{vertex} \rightarrow \text{on}_off = 1 ;
\]
if \((\text{vertex} \rightarrow \text{on}_off \text{ equal } 0)\) then
\[
\text{vertex} \rightarrow \text{on}_off = \text{mult} ;
\]
\[
\text{corner} = \text{corner} \rightarrow \text{next}_corner ;
\]
while (temp $\neq$ NULL)
    while (edge $\neq$ NULL)
        link = edge->link;
        while (link->mortar $\neq$ NULL)
            mortar = link->mortar;
            while (mortar->virtual_vertex $\neq$ NULL)
                v_vertex = mortar->virtual_vertex;
                on_off = v_vertex->on_off;
                mult = v_vertex->mult;
                if (on_off equal mult) then
                    sum -= v_vertex->vertex->data;
                    v_vertex->on_off = 1;
                if (v_vertex->on_off equal 0) then
                    v_vertex->on_off = mult;
                    v_vertex = v_vertex->next_v_vertex;
                link = link->next_link;
        edge = edge->next_edge;
    temp = temp->next_element;
return sum;

This concludes the discussion on the most important new issues related to implementation of nonconforming discretizations: complex topologies, and evolving geometries. Despite the widely differing nature of the three inter-element communication functions (projection, direct stiffness, and inner products), the computational procedures are very similar in construction since all three involve several passages throughout the data structure; it is only the actual operation
that is unique to the particular procedure. In general, we consider advantageous
the development of such user-defined data structures since they preserve the ar-
rangement of elemental data into contiguous memory blocks for vectorization of
tensor-products, they can simplify the representation of data inter-dependencies
in complex domain decompositions, and, they can shift the focus of the algo-
rithmic development into small, general-purpose routines.
Chapter 5
Examples

We conclude with the presentation of two example calculations which illustrate the generality of nonconforming sliding spectral element discretizations in the analysis of fluid flows. The first one is a model of an industrial mixer and the second one is a simulation of the rotor/stator interaction in hydromachinery equipment. Both problems are two-dimensional approximations to the physical situations and the Reynolds number lies in the range of a few hundreds to a thousand.

The cases presented here, can only be considered as preliminary investigations and by no means may they be taken as exhaustive studies of the physical processes involved. However, they are instructive in providing insight to the structure of the flow.

5.1 Mixing Processes

Many applications in the industrial, food, and pharmaceutical industries are carried out in mixing vessels where heat removal or addition, power consumption, and mixedness of the flow are of major importance. When high viscosity liquids are processed, large, low-speed impellers are used with small clearances
from the wall. Nondimensional analysis [18] shows that the fluid flow is primarily a function of Reynolds number and geometric ratios.

A model of such a mixing process is shown in Figure 24. The mixing vessel is of length \( L \). The impeller has a rectangular shape of aspect ratio five and the tips have been smoothed out by placing semicircles of radius equal to the semi-width of the impeller. The ratio of the chamber size to the total impeller length is fixed at two. The impeller and vertical chamber walls are thermally insulated while the top and bottom are held fixed at a constant temperature of \( T_1 \) and \( T_0 \), respectively.

The speed of the impeller \( \omega \), the length of the mixing chamber \( L \), and the fluid’s kinematic viscosity \( \nu \), define the Reynolds number as \( Re = \frac{\omega L^2}{\nu} \). The Peclet number is similarly defined as \( Pe = \frac{\omega L^2}{\alpha} \) where \( \alpha \) is the thermal diffusivity of the fluid. The ratio of momentum to thermal diffusivities is the Prandtl number, defined as \( Pr = \frac{\nu}{\alpha} \) and assumed constant to one. We solve in the problem domain \( \Omega = [0,1]^2 \) the mass, momentum, and energy equations

\[
\nabla \cdot \mathbf{V} = 0 \tag{5.1}
\]

\[
\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla)V = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{V} \tag{5.2}
\]

\[
\frac{\partial \theta}{\partial t} + (\mathbf{V} \cdot \nabla)\theta = \frac{1}{Pe} \nabla^2 \theta \tag{5.3}
\]

with boundary conditions

\[
\mathbf{V} = 0 \quad \text{on} \quad \partial \Omega \cap \partial \Omega^{(1)} \tag{5.4}
\]

\[
\mathbf{V} = \hat{k} \times (x - x_c) \quad \text{on} \quad \partial \Omega \cap \partial \Omega^{(2)} \tag{5.5}
\]

\[
\theta = \begin{cases} 
0 & \text{top} \\
1 & \text{bottom}
\end{cases} \quad \text{wall} \tag{5.6}
\]
\[
\frac{\partial \theta}{\partial n} = 0 \quad \text{on} \begin{cases} \text{side walls} \\ \text{impeller} \end{cases}
\] (5.7)

and initial conditions

\[
\mathbf{V}(x,0) = 0 \quad \text{in } \Omega
\] (5.8)

\[
\theta(x,0) = (1 - y)
\] (5.9)

The symbol \( \hat{k} \) signifies the unit normal vector in the third dimension, and \( x_c \) is the position vector of the impeller center which is also its center of rotation. All dependent and independent variables appearing in the equations have been non-dimensionalized by the length \( L \) of the chamber, the rotational speed \( \omega \) of the mixer, and the temperature difference \((T_1 - T_0)\) between the top and bottom walls. The two quantities of interest are the heat transfer rates at the two horizontal walls (Nusselt number)

\[
Nu_{BT} = - \int_0^1 \frac{\partial \theta}{\partial y}(x, y = 0, 1) \, dx
\] (5.10)

and dissipation (power consumption)

\[
\Phi = \frac{1}{Re} \int_0^1 \int_0^1 \tau_{ij} \frac{\partial u_i}{\partial x_j} \, dx
\] (5.11)

where \( \tau_{ij} \) is the stress tensor \( \tau = [\nabla \mathbf{V} + (\nabla \mathbf{V})^T] \). The time averaged values (in the steady-periodic state) for the Nusselt number and dissipation are denoted as \(< Nu >\) and \(< \Phi >\), respectively.

Figure 25 shows the discretization of the problem in Figure 24. The elements surrounding the impeller and inside the interface \( I \) are moving with rotational speed one, and hence, are able to track the impeller in the course of time, while the elements outside the interface remain fixed to the mixing chamber for all
Figure 24: Geometry of an impeller mixer, comprising a square mixing chamber of length $L$ and a semi-circular-tipped impeller which rotates with speed $\omega$. The domain is divided into two regions: $\Omega^{(2)}$ which rotates along with the impeller and $\Omega^{(1)}$ which remains fixed for all times. Interface $I$ is defined as $I = \overline{\Omega^{(1)}} \cap \overline{\Omega^{(2)}}$. 
Figure 25: Nonconforming spectral mortar-element discretization of the impeller mixer (Figure 24); the chamber and impeller flow regions can be meshed independently, and subsequently brought together.
times. An illustration of the actual mesh is given in Figure 26 where the polynomial in each direction is set to $N = 8$. Each intersection of lines in Figure 26 corresponds to a grid point representing four unknowns (two velocities, temperature, and pressure) which brings the total to approximately five thousand degrees-of-freedom.

We have carried out simulations up to a time when a steady-periodic state is achieved. Figure 27 through 29 show the time history of the dissipation function and Nusselt numbers at the top and bottom walls, for Reynolds number one thousand. The period of the dissipation function (and all other fluid quantities, in general) is, as expected, a quarter of an impeller's revolution. For the thermal problem however, where the insulated boundary conditions on the side walls break the four-fold symmetry, the period increases to half a revolution.

We can verify the correctness of the computation by checking the balance between the energy producing and consuming terms:

$$\frac{d(K.E.)}{dt} = -P - \Phi \quad (5.12)$$

where $K.E.$ is the kinetic energy of the fluid and $P$ is the work done by the fluid on the impeller. They are defined as:

$$K.E. = \int_0^1 \int_0^1 \frac{1}{2} |\nabla|^2 dx \quad (5.13)$$

and

$$P = \hat{k} \cdot \left( \int_{\text{impeller}} (x - x_c) \times F ds \right) \quad (5.14)$$

where $F$ is the resultant force vector due to pressure and shearing forces; its components defined as:

$$F_i = (-p \delta_{ij} + \frac{1}{Re} \partial u_i) \quad (5.15)$$
Figure 26: Mesh for the domain decomposition of Figure 25. The polynomial order is $N = 8$ in each spatial direction. Increased resolution is achieved by raising $N$ and keeping the number of elements $K$ fixed.
Figure 27: Dissipation $\Phi$ as a function of time for the discretization of Figure 25 and Reynolds number $Re = 1000$. The simulation is carried out to a steady periodic state with a time step of $\Delta t = .003$. 
Figure 28: Heat transfer on the top wall as a function of time for the discretization of Figure 25, Reynolds number $Re = 1000$ and $Pr = 1$. The simulation is carried out to a steady periodic state with a time step of $\Delta t = .003$. 

102
Figure 29: Heat transfer on the bottom wall as a function of time for the discretization of Figure 25, Reynolds number $Re = 1000$ and $Pr = 1$. The simulation is carried out to a steady periodic state with a time step of $\Delta t = .003$. After steady periodic conditions are established, the curve shown here is equal and opposite to the one in Figure 28.
where $\delta_{ij}$ is the Kronecker delta. Likewise for the thermal problem, we obtain

$$\frac{d(T.E.)}{dt} = N_u_T - N_u_B$$

(5.16)

where $T.E.$ is thermal energy defined as

$$T.E. = \int_0^1 \int_0^1 \theta dx$$

(5.17)

and, $N_u_T$, $N_u_B$ are the heat transfer rates at the top and bottom walls. Agreement in equations (5.12,5.16) was obtained with a relative error of less than one percent.

At very low Reynolds numbers, for example ($Re = 1$) and ($Pr = 1$), the problem reduces to a quasi-steady Stokes/conduction problem, for which we find $< Nu >= .85$ and $< \Phi >= .47$; the Nusselt number being less than unity due to the insulating effect of the impeller. The velocity vector field is shown in Figure 30. At higher Reynolds numbers, $Re = 1000$, we notice that, at least for this two-dimensional calculation, the velocity vector field, Figure 31, is quite similar to the Stokes solution. This might be expected given the relatively small departure from azimuthal symmetry of the square chamber. However, the temperature field, Figure 32, departs significantly from the conduction solution due to the fact that the temperature boundary conditions (5.6, 5.7) break this symmetry. The result is higher heat transfer rates, as is well known from studies of enhancement by “interrupted” boundary layers [2,36]. We find, at $Re = 1000$, the mean dissipation and Nusselt numbers to be $< \Phi >= .0073$ and $< Nu >= 3.$, respectively.

104
Figure 30: Velocity vectors at $T_f = 37.7$ (six impeller revolutions) for the steady-periodic solution at $Re = 1$. 
Figure 31: Velocity vectors at $T_f = 37.7$ (six impeller revolutions) for the steady-periodic solution at $Re = 1000$. The solution resembles the Stokes solution of Figure 30.
Figure 32: Temperature distribution at $T_f = 37.7$ for $Re = 1000$ and $Pr = 1$. The temperature deviates appreciably from the conduction solution (essentially $T = 1 - \tilde{y}$) due to the non-azimuthally symmetric boundary conditions on the temperature.
5.2 Rotor/Stator Interactions

Our second simulation is a model of rotor/stator configurations in hydro-machinery equipment. Viscous effects occurring at different time scales play a large part in the unsteady operational behavior of such devices. Wake interference due to the periodic passage of the rotor blades and vortex shedding at the blades' trailing edges are two major sources of such behavior. Considerable improvements in hydromachinery design (shorter rotor/stator clearances), thermodynamic efficiency (minimal losses), and reliability (reduced blade fatigue) can be achieved with greater understanding of the fluid dynamical processes involved.

The problem has been studied extensively in the context of turbomachinery equipment for aerospace applications [14,25,51], but, to the best of our knowledge, this is a first attempt as regards high-order discretizations for the incompressible Navier-Stokes equations. The Reynolds numbers considered here are far below the ones encountered in actual situations; a more realistic calculation can be achieved either with a turbulence model or, a, probably prohibitively expensive, high resolution, direct simulation.

A model for the geometric configuration of axial flow hydroturbines is shown in Figure 33 and consists of a single pair of rotor/stator blades. The multitude of rotor and stator blades around the hub is implicit in the imposition of the transverse periodicity condition which, in effect, corresponds to an infinity of blade pairs in that direction. The periodic length in the transverse direction (i.e. the distance between consecutive rotor/stator blades) is denoted as $W$, $L$ is the spacing between the rotor and stator blades, and $\omega$ is the speed of
Figure 33: Model of rotor/stator configuration in a water turbine. Both rotor and stator blade sections are NACA-6512 with chord length c. The cross-section is assumed a distance r from the hub, and the rotor is moving with velocity $\omega r$. Periodic boundary conditions are assumed in the transverse direction.
the rotor. Both blade sections are the NACA-6512 [1] series of wings with chord length $c$ (Appendix A). The wing sections were chosen for their good loading characteristics and not in an effort to match actual blade sections used in hydromachinery.

We denote as $\frac{dp}{dz}$ the pressure head driving the flow in the axial direction and as $U_0 = \sqrt{\frac{1}{\rho} \frac{d\phi}{dz} W}$ a characteristic velocity of the main flow. The mean radial distance away from the hub is defined as $r$, and $M$ is the total number of blade passages corresponding to a single revolution of the hub. The (here, dimensional) equation of motion for the rotor section is

$$\frac{dy}{dt} = \omega r = \frac{\omega MW}{2\pi} \tag{5.18}$$

which, by virtue of the no slip condition at the blade wall, becomes the boundary condition for the fluid velocity as well. After nondimensionalizing the terms in the governing equations by the length $W$, velocity $U_0$, and time $\frac{W}{U_0}$, we obtain

$$\nabla \cdot \mathbf{V} = 0 \tag{5.19}$$

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{V} + \hat{c} \tag{5.20}$$

with boundary conditions

$$\mathbf{V} = \begin{cases} A_j & \text{on the rotor} \\ 0 & \text{on the stator} \end{cases} \tag{5.21}$$

$$\mathbf{V}(0,y,t) = \mathbf{V}(2L,y,t) \tag{5.22}$$

$$\mathbf{V}(x,0,t) = \mathbf{V}(x,W,t) \tag{5.23}$$

and zero initial conditions. In the above, the Reynolds number is defined as $Re = \frac{U_0 W}{\nu}$, and $A$ is the rotor advance coefficient denoted as $A = \frac{\omega MW}{2\pi U_0}$. The boundary conditions are periodic in both the transverse and axial directions; periodicity
in the \( x \)-direction refers to a multitude of rotor/stator stages. Quantities of interest include the flow rate across any vertical plane, (say \( x = 0 \))

\[
Q = \int_0^1 u(x=0, y) dy,
\]

the useful power out (due to the lift force on the rotor)

\[
P = A \int_{\text{rotor}} (\mathbf{F} \cdot \mathbf{j}) ds
\]

where \( \mathbf{F} \) is defined in (5.15), and dissipation \( \Phi \), as defined in (5.11). The energy balance in this case takes the form

\[
\frac{d(K.E.)}{dt} = -P - \Phi + \frac{2L}{W} Q
\]

where the input power is taken to be the axial pressure times the flow rate. Again, the symbol \( <> \) signifies average quantities during steady-periodic conditions.

The spectral element domain decomposition of the problem in Figure 33 is shown in Figure 34. The number of elements is \( K = 32 \) which brings the total number of degrees-of-freedom in the order of 7000 for a polynomial order of \( N = 8 \). The decomposition, although coarse, illustrates the geometric flexibility of the isoparametric spectral element formulation in approximating the curved geometry of the blade sections.

We have performed simulations of the startup flow up to a final time \( T_f = .12 \) for the conditions \( A = 1 \) and \( Re = 50 \). The solution time was in the order of five hours on an Ardent Titan desktop supercomputer running at a speed of \( \sim 1 \) MFLOPS. Longer time simulations and detailed analysis of the physics can be found in [42]. Figure 35 is an illustration of the mesh at the final time. We can
Figure 34: Domain decomposition in $K = 32$ spectral elements for the problem in Figure 33.
Figure 35: Domain configuration at time $T = .12$. The rotor mesh has moved by an amount equal to the rotor displacement.
clearly notice the gap of the two subdomains along the vertical interface separating the rotor and stator blades. Figure 36 shows the corresponding velocity vector field and Figure 37 is an illustration of the instantaneous streamlines for the same final time. The most noticeable feature of the flow is the two vortices around the leading and trailing edges of the rotor blade. The flow field around the rotor blade is similar to the one of a moving flat plate. It is worthwhile to notice the smooth transition of the flow streamlines across the sliding interface; another example of the generality and good approximation properties of the nonconforming discretization presented in this thesis.

5.3 Future Work

The geometric flexibility of the nonconforming formulation in two-dimensional domain decompositions can be readily extended to the case of sliding, compressible meshes. In many physical problems, i.e. wing flaps behind airfoils, the geometric configuration imposes restrictions that prohibit the construction of a mesh which can undergo motion without deformation. The natural choice would be to define a sector around the moving boundary, within whose bounds elements are extended or compressed, and consider in the governing equations the additional term due to the "compressibility" of the mesh.

To complete the two-dimensional implementation of nonconforming decompositions the issue of faster and better solvers must certainly be addressed. Preconditioned conjugate gradient iterations could substantially accelerate the convergence of the solver; furthermore, multigrid iteration holds the potential of decoupling convergence from problem resolution altogether, and therefore, must
Figure 36: Velocity vectors at time $T = .12$ for $Re = 50$ and $A = 1$. At this initial time, the rotor blade generates a flow field similar to that of a moving flat plate.
Figure 37: Instantaneous streamlines showing the two vortices forming at the leading and trailing edges of the rotor blade. The time is $T = .12$, Reynolds number $Re = 50$, and advance coefficient $A = 1$. 
be considered. Other schemes for solution to the Navier-Stokes equations, used in conforming discretizations [40,54], can also improve accuracy of the overall approximation and avoid the occurrence of spurious modes in the pressure solution.

Extension of the formulation to three dimensions would be the natural next step in development of the nonconforming decomposition. A matching condition, in a form similar to equation (2.20), is required to combine the solution over adjacent faces of neighboring elements. The geometric complexity of nonconforming decompositions, which would be significantly increased in three dimensions, will need to be addressed with automatic mesh generation and with robust procedural methods for detection of geometric proximity. At this point, it would also be fruitful to incorporate the work done in the area of error estimation [41] so as to allow for dynamic mesh refinement of the initial grid.

Finally, we suggest the implementation of the "mortar-vertex" constructs to the coupling of finite element and spectral element methods. The coupling of different numerical schemes is a promising approach in that it builds upon the competitive advantages of individual methods. For instance, in a coupled method, we would expect to use spectral elements in the regions of high gradients where good resolution and high accuracy is needed and finite elements in areas where the geometry is irregular and smooth solutions are not expected. The theory has been developed in [11] and numerical applications are underway [10].
Appendix A

NACA four-digit series of wing sections

Most airfoil shapes are defined by giving the coordinates of 50 or more points on its surface. One rather famous series of airfoils, the NACA four-digit series, is defined completely by formulas. The thickness distribution is given by

\[ T(x) = 5rc \left[ .2969 \sqrt{\frac{x}{c}} - .126 \frac{x}{c} - .3537 \left( \frac{x}{c} \right)^2 + .2843 \left( \frac{x}{c} \right)^3 - .1095 \left( \frac{x}{c} \right)^4 \right] \]  

(A.1)

where \( c \) is the airfoil chord length, \( x \) is the distance along the chord from the leading edge, and \( r \) the thickness ratio of the airfoil (maximum thickness/chord).

The camber line consists of two parabolas that meet at the maximum camber point. If \( EC \) is the maximum camber, and \( pc \) is the distance between the leading edge and the maximum camber point, we have

\[
C(x) = \begin{cases} 
\frac{\xi}{p} (2p - \xi) & \text{for } 0 < \frac{\xi}{c} < p \\
\frac{\xi(c-\xi)}{(1-p)^2} (1 + \frac{\xi}{c} - 2p) & \text{for } p < \frac{\xi}{c} < 1.
\end{cases}
\]

(A.2)

In the four-digit designation, the first digit is the maximum camber ratio \( \epsilon \) times 100, the second digit is the chordwise position of the maximum camber \( p \) times 10, and the last two are 100 times the thickness ratio \( r \). The thickness of the NACA four-digit airfoil is defined in the direction perpendicular to the camber line.
Bibliography


