Spectral Element Methods: Algorithms and Architectures

PAUL FISCHER*, EINAR M. RØNQUIST*,
DANIEL DEWEY*, AND ANTHONY T. PATERA*

Abstract. Spectral element methods are high-order weighted residual techniques for partial differential equations that combine the geometric flexibility of finite element methods with the rapid convergence of spectral techniques. In this paper we describe spectral element methods for the simulation of incompressible fluid flows, with special emphasis on implementation of spectral element techniques on medium-grained parallel processors. Two parallel architectures are considered: the first, a commercially available message-passing hypercube system; the second, a developmental reconfigurable architecture based on Geometry-Defining Processors. High parallel efficiency is obtained in hypercube spectral element computations, indicating that load balancing and commumication issues can be successfully addressed by a high-order technique/medium-grained processor algorithm-architecture coupling.

1. Introduction. Spectral element methods are high-order weighted-residual techniques for partial differential equations that exploit both the common foundations and competitive advantages of h-type finite element methods (Strang and Fix, 1973) and p-type spectral techniques (Gottlieb and Orszag, 1977). In the spectral element discretization (Patera, 1984; Maday and Patera, 1987), the computational domain is broken up into macro-spectral elements, and the dependent and independent variables are approximated by Nth order tensor-product polynomial expansions within the individual subdomains. Variational projection operators and Gauss numerical quadrature are used to generate the discrete equations, which are then solved by

*Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

direct or iterative procedures based on tensor-product sum-factorization techniques. Convergence to the exact solution is achieved by increasing the degree, N, of the polynomial approximation, while keeping fixed the number and identity of the underlying spectral elements.

The spectral element discretization is an example of *domain decomposition* in that an individual spectral element is the "basic unit" as regards strong coupling between degrees-of-freedom. In the spectral element discretization the elements serve as the basic building blocks for mesh generation and physics, as well as the basic units of locally structured mesh. The latter is critical as regards the computational efficiency of the method, as it allows for the sum-factorized tensor-product matrix-vector products that render high-order methods competitive with low-order techniques.

Although spectral element methods have proven competitive with h-type finite element methods for incompressible flow simulations on serial and vector computers (Ghaddar, Magen, Mikic and Patera, 1986), it is clear that in the future techniques must be judged on the basis of their performance on parallel machines. Fortunately, whereas *global* spectral methods are not obviously parallelizable due to the strong coupling of all points in the computational domain, spectral *element* methods have an intrinsic "domain-decomposition" granularity that in turn leads to natural and efficient implementation on medium-grained parallel processors.

In this paper we focus on the parallel implementation of spectral element algorithms. In Section 2 we describe the basic spectral element discretization as applied to elliptic problems. In Section 3 we discuss the parallel implementation of spectral element elliptic algorithms on a message-passing hypercube system, and present results for parallel efficiency on a commercially available Intel machine. The viability of the high-order method/medium-grained approach to parallel computing is discussed, and the issues of communication latency and load-balancing are addressed. Lastly, in Section 4 we present a developmental special-purpose reconfigurable architecture, Geometry-Defining Processors, and discuss the advantages of parallel implementation of spectral element and substructured finite element algorithms on this novel configuration.

2. Spectral Element Methods.

Elliptic Equations. To illustrate the basic spectral element concepts we first consider the following one-dimensional elliptic Helmholtz problem: Find u(x) defined over $\Lambda =]-1,1[$ such that

$$-(pu')' + \lambda^2 u = f \qquad x \in \Lambda \quad , \tag{2.1a}$$
$$u(-1) = u(1) = 0 \quad . \tag{2.1b}$$

Here prime denotes differentiation, and we assume that $p(x) > p_0 > 0$, and $\lambda \in \mathbb{R}$. The spectral element discretization for solving (2.1) is based on the variational formulation: Find $u \in H_0^1(\Lambda)$ such that

$$(2.2) \qquad a(u,v) = (f,v) \qquad \forall v \in H_0^1(\Lambda) \quad ,$$

where

$$(2.3a) \qquad a(\phi,\psi) = \int_\Lambda p(x)\phi'(x)\psi'(x)dx + \lambda^2 \int_\Lambda \phi(x)\psi(x)dx \quad ,$$

and,

$$(2.3b) \qquad (\phi,\psi) = \int_\Lambda \phi(x)\psi(x)dx \quad .$$

Here H_0^1 is the space of all functions which satisfy the homogeneous boundary conditions (2.1b), which are square integrable, and whose derivatives are also square integrable.

The spectral element discretization proceeds by breaking up the interval Λ into K subintervals (spectral elements) $\Lambda_1,\ldots,\Lambda_K$, where the length of the k^{th} interval is L_k. The space of approximation for the solution u is then taken to be a high-order polynomial subspace of $H_0^1(\Lambda)$, $X_h = H_0^1(\Lambda) \cap P_{N,K}(\Lambda)$, where $P_{N,K}(\Lambda) = \cup_k P_N(\Lambda_k)$, and $P_N(\Lambda_k)$ is the space of all polynomials of degree $\leq N$ on the interval Λ_k. To accurately evaluate the integrals associated with the bilinear forms in (2.3) we use Gauss-Lobatto Legendre numerical quadrature with quadrature points and weights given by (ξ_i, ρ_i), $i=0,\ldots,N$.

The abstract formulation of the resulting discrete problem is: Find $u_h \in X_h$ such that

$$(2.4) \qquad a_{h,GL}(u_h,v_h) = (f,v_h)_{h,GL} \qquad \forall v_h \in X_h ,$$

where subscript h,GL refers to Gauss-Lobatto numerical quadrature of the bilinear forms in (2.3). Convergence of the spectral element solution u_h to the exact solution u is then achieved by increasing the *degree* of the approximation, N, for *fixed* K. It is shown in Maday and Patera (1987) for this spectral (p-type) convergence strategy that the error goes to zero exponentially fast with N for sufficiently smooth solutions u. This should be contrasted to finite element h-type methods (N fixed, $K \Rightarrow \infty$) for which algebraic convergence results.

To proceed further, we represent our polynomial space with a nodal basis, that is, any function w_h in X_h is written in terms of elemental Lagrangian interpolants $h_i(r)$ through the Gauss-Lobatto Legendre points,

$$(2.5a) \qquad w_h^k(r) = \sum_{i=0}^{N} w_i^k h_i(r) \qquad x \in \Lambda_k \Rightarrow r \in \Lambda \quad ,$$

$$(2.5b) \qquad h_i \in P_N(\Lambda), \quad h_i(\xi_j) = \delta_{ij} \quad \forall i,j \in \{0,\ldots,N\}^2 \quad ,$$

where w_i^k is the value of w_h at the local point ξ_i in the interval Λ_k. In order to ensure C^0-continuity and satisfy the boundary conditions we further require that

$$(2.6a)\qquad w_N^k = w_0^{k+1} \quad \forall k\in\{1,\ldots,K-1\},$$

and

$$(2.6b)\qquad w_0^1 = w_N^K = 0 \quad .$$

Expressing u_h and v_h in terms of the nodal basis (2.6), and choosing each test function to be nonzero at only one global collocation point, we arrive at the following set of discrete equations:

$$(2.7)\qquad \sum_{k=1}^{K,} \Big[\frac{2}{L_k}\sum_{j=0}^{N}\sum_{q=0}^{N} \rho_q p_q^k D_{qi} D_{qj} u_j^k + \lambda^2 \frac{L_k}{2}\rho_i u_i\Big] = \sum_{k=1}^{K,} \frac{L_k}{2}\rho_i f_i^k ,$$

where p_q^k and f_q^k are the values of $p(x)$ and $f(x)$ at the local point $r=\xi_q$ in Λ_k, $D_{ij}=dh_j(\xi_i)/dr$, and Σ' denotes the direct stiffness summation which incorporates (2.6).

As a numerical example we consider the problem (2.1) with $p(x)=e^x$, $f(x)=e^x(\cos x-\sin x)$, $\lambda=0$ on $x=]0,\pi[$, for which the solution is $u(x)=-\sin x$. The particular spectral element discretization corresponds to division of the domain into K=2 similar spectral elements, each of order N. Fig. 1 shows a plot of the maximum nodal error $||u-u_h||_{GL,L^\infty}$ as a function of the total number of degrees-of-freedom $N_t=2N+1$. Exponential convergence is achieved due to the fact that the solution is analytic.

The spectral element discretization of multi-dimensional elliptic equations corresponds to a tensor-product extension of the one-dimensional method. We consider here the two-dimensional Poisson equation on a domain Ω with homogeneous boundary conditions on the domain boundary $\partial\Omega$,

$$(2.8a)\qquad -\Delta u = f \quad \text{in } \Omega$$
$$(2.8b)\qquad u = 0 \quad \text{on } \partial\Omega.$$

In this case the domain is broken up into K quadrilateral elements $\Omega_1,\ldots,\Omega_K$, and the corresponding discrete formulation is given by: Find $u_h \in X_h$ such that

$$(2.9)\qquad (\nabla u_h,\nabla v_h)_{h,GL} = (f,v_h)_{h,GL} \quad \forall v_h\in X_h,$$

where $X_h=H_0^1(\Omega)\cap P_{N,K}(\Omega)$. Here $P_{N,K}(\Omega)=\cup_k P_N(\Omega_k)$, and $P_N(\Omega_k)$ is the space of all piecewise polynomials of degree $\leq N$ with

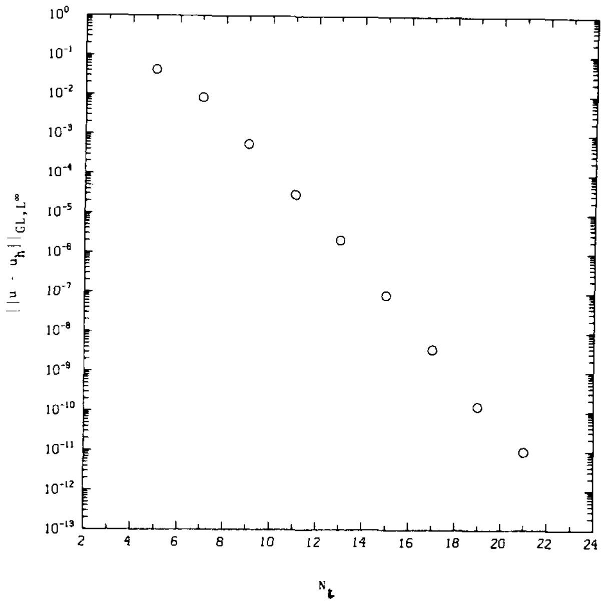

Figure 1. A plot of the maximum nodal error in the spectral element solution to the differntial equation (2.1) as a function of the total number of degrees-of-freedom, N_t. In this problem $\lambda=0$, $p(x)=e^x$, $f(x)=e^x(\cos x-\sin x)$, for which the solution is $u=-\sin x$ on $x \in]0,\pi[$. The domain is divided into K=2 spectral elements of equal length. Exponential convergence is achieved as the polynomial degree of the fixed elements is increased.

respect to each spatial variable x and y. For an error analysis of the discretization (2.9) we refer to Funaro (1986) where it is shown that spectral convergence obtains as $N \Rightarrow \infty$ for fixed K.

A basis for the space X_h is constructed by expressing any function w_h in X_h in tensor-product form within each element Ω_k ($(x,y) \in {}^h\Omega_k \Rightarrow (r,s) \in \Lambda\times\Lambda$),

$$(2.10) \qquad w_k^k(r,s) = \sum_{i=0}^{N} \sum_{j=0}^{N} w_{ij}^k h_i(r) h_j(s)$$

where h_i are the one-dimensional Gauss-Lobatto Lagrangian interpolants defined in (2.5b). The solution u_h, the testfunctions v_h, and the (isoparametric) geometry (x_h,y_h) are all expressed in terms of the nodal basis (2.10), and the integrals in (2.9) are evaluated using Gauss-Lobatto Legendre $\times$ Gauss-Lobatto Legendre quadrature in (r,s). Choosing appropriate "delta" testfunctions v_h we arrive at the set of linear equations,

$$(2.11) \qquad \underline{A}\,\underline{u} = \underline{B}\,\underline{f} \ ,$$

where $\underline{A}$ is the discrete Laplace operator, $\underline{B}$ is the (diagonal) mass matrix, and underscore denotes nodal unknowns.

As a two-dimensional example we consider the problem (2.8a) on the domain $\Omega = (x \in]0,1[, y \in]0, 1+\frac{1}{4}\sin \pi x[)$ with $f=0$ and $u = \sin x \cdot e^{-y}$. Fig. 2 shows the isoparametric spectral element discretization (K=4), while in Figure 3 we plot the maximum nodal error $||u-u_h||_{GL,L^\infty}$ as a function of the number of degrees-of-freedom in one spatial direction. Note that although the domain is relatively deformed compared to the mapped rectilinear problem, exponential convergence to the analytic solution is obtained (Ronquist and Patera, 1987).

<u>Elliptic Solvers</u>. In this section we address the problem of solution of the linear positive-definite symmetric systems resulting from spectral element discretization of self-adjoint elliptic equations. For large three-dimensional problems the most attractive approach is iterative solvers, both in terms of operation count and storage requirements. In order for any solution method to be efficient in this context, fast matrix-vector evaluation schemes are needed; for high-order methods this is effected by tensor-product sum factorization, which we now discuss.

Consider the solution of the two-dimensional Poisson equation (2.8) in a rectilinear domain Ω, resulting in the spectral element discretization (2.11). The matrix-vector product $\underline{A}\,\underline{u}$ can be written in terms of an elemental sum

$$(2.12) \qquad \underline{A}\,\underline{u} \equiv \sum_{k=1}^{K,} \sum_{m,n}^{N} A^k_{ijmn} u^k_{mn} \qquad \forall i,j \in \{0,\ldots\ldots,N\}^2 .$$

Naive evaluation of (2.12) requires $O(N^4)$ operations per element, and also requires $O(N^4)$ storage as the elemental matrix A^k_{ijmn} is, in general, full. Fortunately, due to the tensor-product representation (2.10), a typical elemental term in (2.12) can be factored as (consider here only the term corresponding to $\partial^2/\partial x^2$)

$$(2.13) \qquad [\sum_{q=0}^{N} D_{qi} [\ \rho_q \rho_j [\sum_{m=0}^{N} D_{qm} u^k_{mj}\ ]]] \quad \forall i,j \in \{0,\ldots,N\}^2 \ ,$$

yielding an operation count of $O(N^3)$ and a storage requirement of only $O(N^2)$.

For three-dimensional problems the naive approach of first forming the elemental matrices A^k and then evaluating the elemental matrix-vector products $A_k \underline{u}^k$ results in an operation count and storage of $O(N^6)$ per element, while the tensor-product sum factorization yields an operation count of $O(N^4)$ and storage requirement of $O(N^3)$. It should be noted that for a general d-dimensional problem, the operation count $O(N^{d+1})$ persists even for isoparametric discretization of non-separable problems.

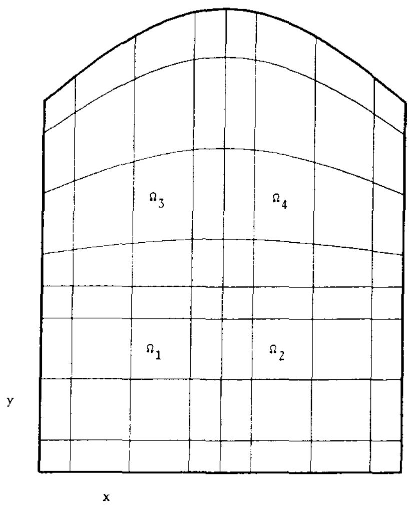

Figure 2. The domain and spatial discretization for solution of the Poisson equation (2.8a) with f=0. Dirichlet boundary conditions are imposed such that the solution is $u = \sin x \cdot e^{-y}$. The domain is divided into K=4 spectral elements.

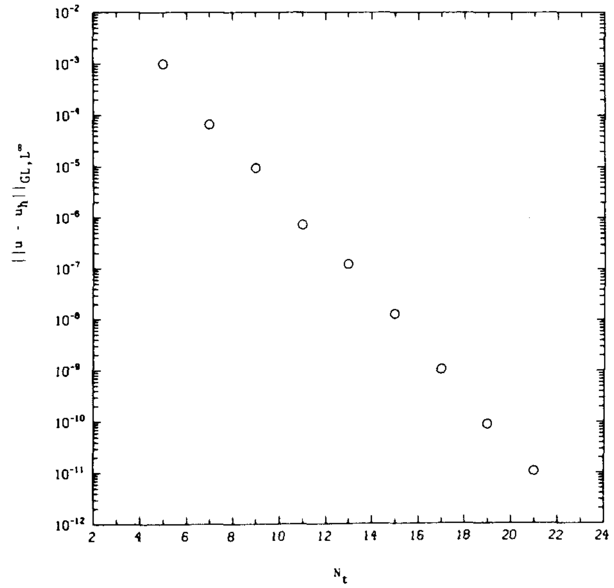

Figure 3. A plot of the maximum nodal error in the Legendre spectral element solution of the Poisson equation (2.8a) as a function of the total number of degrees-of-freedom in one spatial direction, N_t. Exponential convergence is achieved as the degree of the the elements is increased.

Given the efficient tensor-product evaluation, the system (2.11) is solved by preconditioned conjugate gradient iteration (Golub and Van Loan, 1983). First, we construct $\tilde{A}$, an h-type finite element approximation defined on the spectral element quadrature points (Orszag, 1980; Deville and Mund, 1985). The matrix $\tilde{A}$ is then further approximated by its incomplete Cholesky decomposition, $\tilde{A}_I$, which in turn is used as a preconditioner for A (Meijerink and Van der Vorst, 1977; Kershaw, 1978). As a test problem to demonstrate the preconditioned conjugate gradient algorithm, we consider solution of the two-dimensional Poisson equation (2.8a), with $f=e^{x+y}$, $u=\frac{1}{2}e^{x+y}$, and Ω defined by $(x\in]0,1[,y\in]0,1[)$. The domain is discretized by an array of $K=K_1^2$ spectral elements, each of order N. In Figure 4 we plot the convergence history as a function of number of iterations for $K_1=4$ and N=5,7,9, and 11. The results demonstrate the significant savings due to effective preconditioning.

<u>Stokes Discretizations</u>. To illustrate how the spectral element discretization of the Poisson equation extends to the full Stokes equations we consider here the following problem: Find a velocity $u=(u,v)$ and a pressure p in the domain $\Omega=]-1,1[x]0,2\pi[$ such that

$$-\nu\Delta u + \nabla p = f \quad , \tag{2.14a}$$
$$-\mathrm{div}\, u = 0 \quad , \tag{2.14b}$$

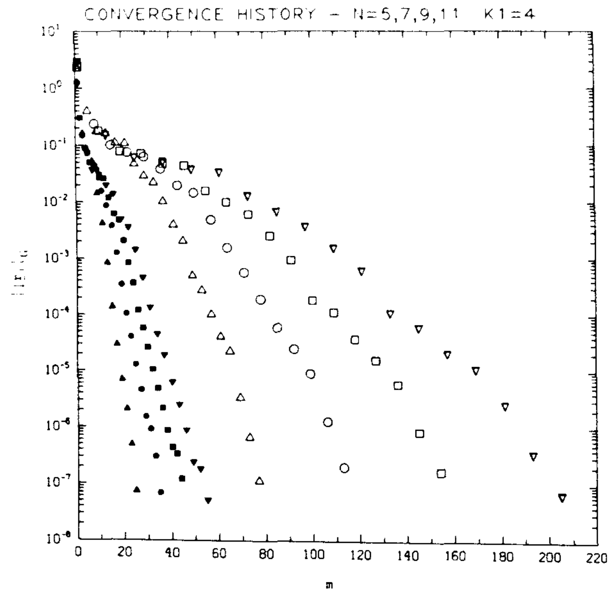

Figure 4. Convergence history for preconditioned (closed symbols) and unpreconditioned (open symbols) conjugate gradient solution of a spectral element Poisson discretization. In all cases $K_1=4$, with N=5(Δ), 7(O), 9($\square$), and 11(∇).

subject to the semi-periodic boundary conditions

(2.15a) $\forall y \in]0,2\pi[,\ u(-1,y) = u(1,y) = 0$,
(2.15b) $\forall x \in]-1,1[,\ u(x,0) = u(x,2\pi)$.

Here ν is the kinematic viscosity of the fluid, and f is the prescribed force.

Due to periodicity the dependent variables can be written in terms of Fourier series in the y-direction,

(2.16a) $$u(x,y) = \sum_{n=-\infty}^{\infty} \hat{u}^n(x) \exp(\underline{i}ny) \quad ,$$

(2.16b) $$p(x,y) = \sum_{n=-\infty}^{\infty} \hat{p}^n(x) \exp(\underline{i}ny) \quad ,$$

where $\underline{i} = \sqrt{-1}$. The Fourier representation (2.16) results in a set of decoupled equations for each Fourier mode n, the variational statement of which is given by (dropping the Fourier superscript and carat notation): For $n \in N^*$ find $\mathbf{u}=(u,v)$ in $X=[H_0^1(\Lambda)]^2$ and p in $M=L^2(\Lambda)$ such that

(2.17a) $$\nu[(u_x,w_x)+n^2(u,w)]-(p,w_x+\underline{i}nz)=(f,w) \quad ,\forall w=(w,z)\in X$$

(2.17b) $$(q,u_x+\underline{i}nv)^* = 0 \quad ,\forall q\in M \ ,$$

where $\Lambda=]-1,1[$, L^2 is the space of square-integrable functions, * refers to complex conjugate, and $(\phi,\psi) = \int_\Lambda \phi\psi^* dx$. For simplicity, we do not consider the case of n=0.

In the same fashion as for the elliptic problem (2.1), the spectral element discretization proceeds by breaking up the interval Λ into K subintervals and searching for a solution in polynomial subspaces of X and M. However, choosing the polynomial degree N to be the same for the velocity and the pressure leads to spurious modes in the pressure. It can be shown that an optimal strategy is to use a staggered mesh for the pressure for which the associated polynomial degree is N-2 (Bernardi and Maday, 1987; Maday, Patera and Rønquist, 1987a; Maday, Patera and Rønquist, 1987b). The discrete formulation corresponding to (2.17) can then be stated as: Find $\mathbf{u}_N$ in $X_N=[H_0^1(\Lambda)\cap \mathbf{P}_{N,K}(\Lambda)]^2$ and p_N in $M_N=\mathbf{P}_{N-2,K}(\Lambda)$ such that

(2.18a) $$\nu[(u_{Nx},w_x)_{N,GL}+n^2(u_N,w)_{N,GL}]-(p_N,w_x+\underline{i}nz)_{N,G}$$
$$= (f,w)_{N,GL}, \quad \forall w = (w,z)\in X_N$$

(2.18b) $$(q,u_{Nx} + \underline{i}nv_N)^*_{N,G} = 0 \ , \quad \forall q\in M_N \quad ,$$

where the subscripts N,G and N,GL denote numerical quadrature based on the Gauss and Gauss-Lobatto points, respectively. Theoretical error estimates for the approximation error due to the discretization (2.18) can be found in (Maday and Patera, 1987; Maday, Patera and Rønquist, 1987a), in which spectral convergence is demonstrated as $N \Rightarrow \infty$ for fixed K.

We now define the bases for the space $X_N \times M_N$. In each element Λ_k the velocity u_N is expanded in terms of N^{th} order Lagrangian interpolants h_i through the Gauss-Lobatto Legendre points ξ_i, $i=0,\ldots,N$,

$$(2.19)\quad u_N^k(r) = \sum_{i=0}^{N} u_i^k h_i(r) \qquad x \in \Lambda_k \Rightarrow r \in \Lambda ,$$

while the pressure is expanded in terms of $(N-2)^{th}$ order Lagrangian interpolants $\tilde{h}_i$ through the Gauss Legendre points ζ_i, $i=1,\ldots,N-1$,

$$(2.20)\quad p_N^k(r) = \sum_{i=1}^{N-1} p_i^k \tilde{h}_i(r) \qquad x \in \Lambda_k \Rightarrow r \in \Lambda .$$

Note the Gauss points are naturally suited for the pressure, which need not be continuous across elemental boundaries. The expansions for the the velocity and the pressure are inserted into (2.18), appropriate "delta" testfunctions $w_N \in X_N$ and $q_N \in M_N$ are chosen, and we arrive at the discrete saddle problem,

$$(2.21a)\quad \underline{A}\,\underline{u} - \underline{D}^T\underline{p} = \underline{B}\,\underline{f} \quad ,$$
$$(2.21b)\quad -\underline{D}\,\underline{u} = \underline{0} \quad .$$

Here the matrix $\underline{A}$ is the discrete Laplacian, $\underline{B}$ is the mass matrix, $\underline{D}$ is the discrete gradient operator, and underscore refers to nodal unknowns.

As a numerical example we consider the following test problem with $\nu=1.0$, $n=1$,

$$(2.22a)\quad (u,v) = (-(1+\cos\pi x),\ \underline{i}\sin\pi x)$$
$$(2.22b)\quad p = \sin\pi x$$
$$(2.22c)\quad (f,g) = (-(1+\cos\pi x)/\pi,\ \underline{i}(2+\pi^2)\sin\pi x) ,$$

on a domain Λ which is divided into K=2 similar spectral elements. We show in Figure 5 the error in the velocity and the pressure as the order N of the polynomial expansions is increased. As expected, exponential convergence is achieved due to the fact that the solution is analytic. In the case when the solution is less regular we obtain an algebraic convergence rate reflecting all the regularity of the solution (Maday, Patera and Rønquist, 1987a). Note that we obtain spectral convergence with only limited continuity between elements due to proper choice of the weak form.

<u>Stokes Solvers</u>. In this section we consider solution of the algebraic system of equations (2.21) resulting from the spectral element discretization of the Stokes problem (2.14). It should be noted that the algorithms presented here are equally appropriate for other types of variational discretizations, and are in fact extensions of the classical

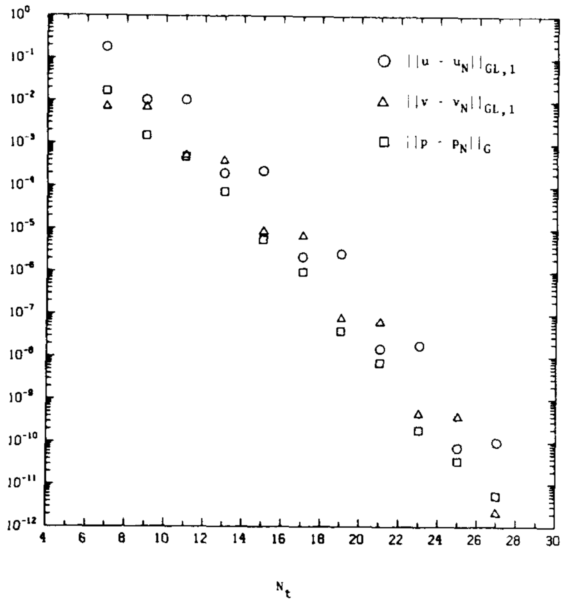

Figure 5. The error in the velocity $u_N=(u_N,v_N)$ and the pressure p_N as a function of the total number of degree-of-freedom (Gauss-Lobatto Legendre points) in the x-direction, N_t, when solving the test problem (2.22). The total interval $\Lambda=]-1,1[$ is divided into K=2 spectral elements $\Lambda_1=]-1,0[$ and $\Lambda_2=]0,1[$. Exponential convergence is obtained.

Uzawa algorithm used in finite element analysis (Girault and Raviart, 1986; Bristeau, Glowinski and Periaux, 1987).

Our approach (Maday, Meiron, Rønquist and Patera, 1987) to solving (2.21) is a global iterative procedure in which the original coupled saddle problem is decoupled into two positive-definite symmetric forms,

$$-\underline{\mathbf{A}}\,\underline{\mathbf{u}} + \underline{\mathbf{D}}^T\underline{\mathbf{p}} = \underline{\mathbf{B}}\,\underline{\mathbf{f}} \quad , \tag{2.23a}$$

$$\underline{\mathbf{S}}\,\underline{\mathbf{p}} = \underline{\mathbf{D}}\,\underline{\mathbf{A}}^{-1}\underline{\mathbf{B}}\,\underline{\mathbf{f}} \quad , \tag{2.23b}$$

where

$$\underline{\mathbf{S}} = \underline{\mathbf{D}}\,\underline{\mathbf{A}}^{-1}\underline{\mathbf{D}}^T \quad . \tag{2.24}$$

In the solution process (2.23b) is first solved for $\underline{\mathbf{p}}$ and then (2.23a) is solved for $\underline{\mathbf{u}}$ with $\underline{\mathbf{p}}$ known.

The equation for the velocities $\underline{\mathbf{u}}$ correspond to standard Laplacian solves, and the inversion of the matrix $\underline{\mathbf{A}}$ is done by conjugate gradient iteration. Although the matrix $\underline{\mathbf{S}}$ in the equation for the pressure is full due to the presence of $\underline{\mathbf{A}}^{-1}$, the matrix is extremely well-conditioned.

In particular, it can be shown that the matrix $\underset{\sim}{S}$ is spectrally close to the variational equivalent of the identity matrix $\underset{\sim}{I}$, namely the mass matrix $\underset{\sim}{B}$ defined on the pressure mesh. This suggests an inner/outer conjugate gradient iteration procedure in which the outer iteration corresponds to inverting the matrix $\underset{\sim}{S}$ preconditioned with the diagonal mass matrix $\underset{\sim}{B}$, and the inner iteration corresponds to inverting the discrete Laplacian, $\underset{\sim}{A}$. As the condition number for the matrix $\underset{\sim}{B}^{-1}\underset{\sim}{S}$ is of order unity, the above algorithm requires only order unity elliptic solves, and is therefore an ideal decoupling of the Stokes problem.

We now present numerical results demonstrating the good conditioning of the matrix $\underset{\sim}{S}$ for the semi-periodic model problem (2.14) and (2.15). Figure 6 shows the spectrum $\lambda_k^S(n)$ of $\underset{\sim}{B}^{-1}\underset{\sim}{S}$ for the spectral element discretization with K=4, N=7, and wave number n=1. The clustering of the spectrum around unity is apparent, and a comparison with the continuous operator spectrum is seen to be virtually exact (Maday, Meiron, Rønquist and Patera, 1987). For multi-dimensional Stokes problem it can be shown that the conditioning κ^S of $\underset{\sim}{B}^{-1}\underset{\sim}{S}$ is still of order unity, allowing for rapid convergence of the outer iteration. It should be noted that the good conditioning of $\underset{\sim}{B}^{-1}\underset{\sim}{S}$ is directly related to good approximation of the pressure (Maday and Patera, 1987).

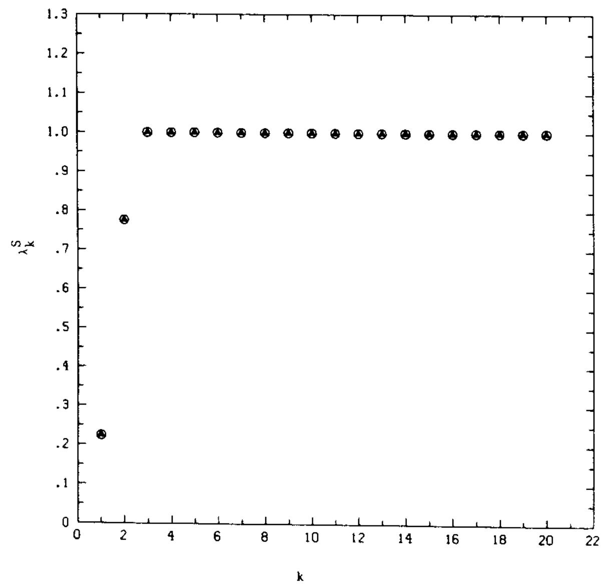

Figure 6. A plot of the spectrum $\lambda_k^S(n)$ of the preconditioned matrix $\tilde{\underset{\sim}{B}}^{-1}\underset{\sim}{S}$, where $\underset{\sim}{S}$ is the pressure matrix given in (2.24) and $\tilde{\underset{\sim}{B}}$ is the mass matrix defined on the Gauss pressure mesh. The spectrum (Δ) corresponds to a spectral element discretization K=4, N=7 for a wavenumber n=1; the agreement with the continuous operator spectrum (O) is very good.

The discretization and solution of the steady Stokes equations (2.14) can be readily extended to solve the unsteady Stokes equations. The implicit Stokes algorithm can be further extended to solve the unsteady Navier-Stokes equations by explicit treatment of the nonlinear term using a third-order Adams-Bashforth scheme. The resulting Navier-Stokes algorithm can be viewed as an implicit Stokes scheme with an "augmented" force which includes the explicit convective contributions. Numerous spectral element calculations of unsteady flows have been performed to date (Ghaddar, Korczak, Mikic and Patera, 1986; Ghaddar, Magen, Mikic and Patera, 1986).

3. Implementation on a Hypercube.

Introduction. It is clear that the new direction of high performance architectures will entail large scale parallelism, and that successful algorithms must be able to exploit these new technological advances. The spectral element discretization and solution algorithms described in the previous section have been designed with parallel architecture in mind; in this section, we outline some parallel algorithm concepts, and describe the implementation of the spectral element method on the Intel iPSC Hypercube parallel processor.

The Intel iPSC Hypercube is typical of the general class of parallel architectures for which our algorithms are appropriate. The Intel machine is a true multiple instruction-multiple data parallel processor, with $M=2^D$ processors arranged in a classical hypercube fashion (Saad and Schultz,1985). Each processor has local source code operating on local data, with execution proceeding asynchronously until data is needed from another processor in the system. Information is exchanged between processors via a message passing scheme, in which data_send and data_receive statements are issued by the participating processors. Efficient use of all the processors can be achieved only if the ratio of communication time to computation time is small, and if all the processors have the same amount of work, that is, are balanced as regards their computational load.

The key aspects of the spectral element formulation of the Navier-Stokes problem which allow for successful parallelization are the following:

a) use of a domain decomposition approach;
b) limited coupling between domains;
c) high order discretization within domains;
d) iterative solution techniques.

Point (a) addresses the central issue of how one divides the problem among the available processors. Following the domain decomposition approach, one or more spectral elements are assigned to each processor, with all the coefficients, geometry, and data associated with each element being locally resident. For our problem, the most natural decomposition is geometric, that is, adjacent elements reside on the same processor so that inter-processor communication can be minimized. An alternative approach is to use a random element placement such that incremental work associated with local mesh refinement is distributed among several processors (Fox and Otto, 1985).

Points (b) and (c) address the requirement that there be minimal communication/computation if high parallel efficiencies are to be attained. The C^0 continuity condition at elemental interfaces implies that the inter-element communication will be $O(N^2)$ in $\mathbb{R}^3$, while the high-order discretization implies that the computational work will be $O(N^4)$ (see Section 2). In essence, high-order methods imply a large amount of computation per point, thus resulting in a low ratio of communication to computation. Point (d) essentially addresses the load balance issue; iterative procedures such as conjugate gradient iteration are more readily amenable to node-simultaneous calculation than direct methods such as Gaussian elimination. By the same argument standard conjugate gradient preconditioning algorithms, such as incomplete Cholesky factorization, are not readily parallelizable; work is underway to find suitable alternatives.

<u>Algorithm Description</u>. At the heart of the parallel Navier-Stokes algorithm are the elliptic solvers which employ (unpreconditioned) conjugate gradient iteration (Golub and Van Loan, 1983). To illustrate the parallel implementation we describe the central step in this procedure, which is formation and evaluation of matrix vector products of the type:

$$(3.1) \qquad \underline{r} = A\underline{p} \quad ,$$

where $\underline{r}$ is the desired global result vector, $\underline{p}$ is an intermediate search direction, and A is the global Laplacian operator defined in Section 2. By definition of the variational statement (2.2-2.3), (3.1) can be written as:

$$(3.2) \qquad \underline{r} = \Sigma' A^k \underline{p}^k \quad ,$$

where A^k is the local Laplace operator coupling the nodal unknowns within the kth element, and $\underline{p}^k$ represents the values of $\underline{p}$ in element k.

To compute $\underline{r}$ in parallel, we first calculate

$$(3.3) \qquad \tilde{\underline{r}}^k = A^k \underline{p}^k \qquad k=1,2,\ldots,K \quad ,$$

and then compute

$$(3.4) \qquad \underline{r} = \Sigma' \tilde{\underline{r}}^k \; .$$

Here direct stiffness corresponds to summation of all elemental nodal values corresponding to the same global degree-of-freedom. Note that in practice (3.4) is not a global operation, but corresponds to local exchange of face data between neighboring spectral elements. The sequence of steps (3.3-3.4) is depicted graphically in Figure 7.

The only other computational step in the conjugate gradient algorithm requiring inter-processor communication is the evaluation of inner products of the intermediate result vectors. This is readily handled using a substructuring technique, whereby local contributions to the inner product are computed on individual processors and then summed via a binary-tree-like gather and sum. The final

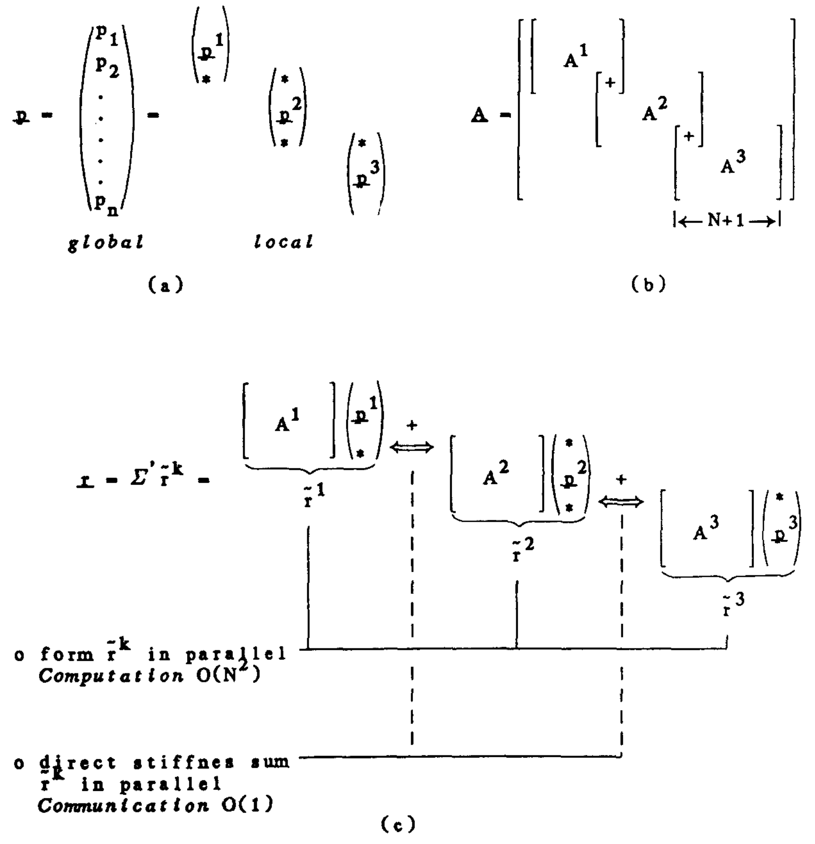

Figure 7. One dimensional example of parallel matrix-vector multiply, $\underline{r}=\underline{A}\underline{p}$, for three spectral elements. (a) Global and local representations of intermediate search direction $\underline{p}$, where * indicates nodal values at element interfaces which must be equal by the C^0 continuity requirement. (b) Global form of the discrete Laplace operator, $\underline{A}$, where + indicates summation of overlapping contributions from local matrices. (c) Expansion of global product into three local products, $\tilde{r}^k$, evaluated in parallel. The $\tilde{r}^k$ are then direct stiffness summed to form the final result vector, $\underline{r}$. Final ratio of computation to communication is $O(N^2)$ in all space dimensions.

result is then redistributed to all the processors via the same binary tree, resulting in a total communication count of 2D, where D is the dimension of the hypercube network (Saad and Schultz, 1985). The communication associated with the inner products is therefore small compared to that associated with direct stiffness summation.

Since the direct stiffness summation (3.4) involves only the faces of the spectral elements it is an $O(N^2)$ operation (per element) in $\mathbb{R}^3$, whereas the matrix-vector product (3.3) is an $O(N^4)$ operation. The ratio of calculation to communication is therefore $O(N^2)$, which will typically be quite large and will thus result in non-communication bound simulations. Note also that the direct stiffness sum message length of N^2 will be relatively large, and thus startup costs associated with each data_send will be amortized over a large number of words, particularly on pipelined configurations. Thus, in theory, the spectral element/conjugate gradient algorithm suffers little communication overhead when implemented on a distributed memory parallel processor.

As an example of the communication overhead incurred in practice, we plot in Figure 8 parallel efficiency η vs. D for solution to a Poisson equation in $\mathbb{R}^3$. Parallel efficiency is defined here as:

$$\eta = \frac{T_1}{M\,T_M}, \tag{3.5}$$

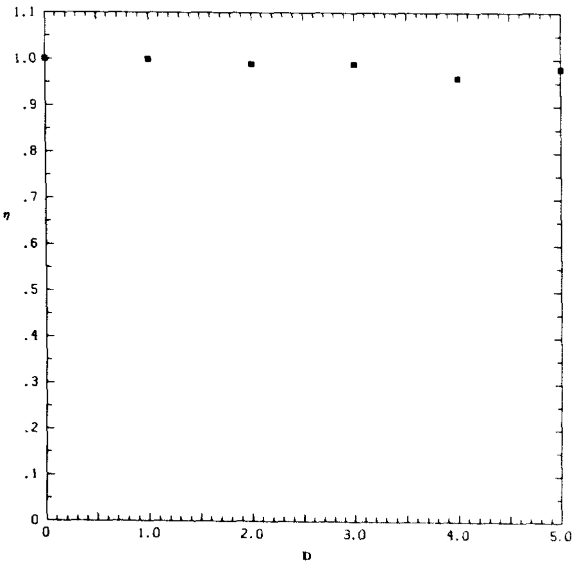

Figure 8. Parallel efficiency vs. hypercube dimension for solution to three-dimensional Poisson equation using sixteen P_6 spectral elements.

where $M=2^D$ is the number of processors, and T_m is the time required to solve a given problem on m processors. The efficiencies range from 0.96 to 0.99 for D ranging from 0 to 5. The K=32 spectral element discretization used in this example is of moderate order, N=6; higher order solutions will show a further increase in parallel efficiency.

The actual efficiency obtained will of course be a strong function of the particular hardware used. The results of Figure 8 were obtained on a D=5 iPSC Hypercube without high performance vector processors. Increases in floating point performance will increase the absolute calculation speed, but will at the same time make the parallel efficiency worse. The question remains as to whether communications capability will continue to keep pace with the rapid advances in floating point hardware.

Virtual Parallel Processor. Central to the development of the parallel spectral element algorithm is the treatment of each element as a separate entity, that is, as a virtual processor. All data structures are set up on an element by element basis, and all operations are executed element by element on this element-local data. In this way, parallelism is inherent in the algorithm, and not overly dependent upon the particular machine on which the code is run.

The practical implementation of this virtual parallel processor concept is relatively simple. The critical feature is that all outer loops in the code run over the number of elements; on a serial machine, loops extend over the entire domain, while on a parallel machine, loops extend over the subset of elements associated with a given processor. The source code on each of the processors is therefore identical. The spectral element-virtual processor association further insures that the few operations requiring communication between processors (direct stiffness summation, inner product evaluation) can be easily effected by device drivers which translate send and receive subroutines into hardware-compatible data_send and data receive commands.

Alternative Algorithm/Architecture Couplings. We now consider our particular choice of architecture and algorithm in light of other available alternatives. Our algorithm is well suited for what we define as medium-grained parallel processing. Defining N_t to be the total number of degrees-of-freedom, M to be the number of processors, and $\rho = N_t/M$, we classify systems as

$$\begin{aligned} \text{coarse-grained:} &\quad N_t \Rightarrow \infty,\ M \text{ fixed},\ \rho \Rightarrow \infty &&; \\ \text{medium-grained:} &\quad N_t \Rightarrow \infty,\ M \Rightarrow \infty,\ \rho >> 1,\ (\rho \not\Rightarrow \infty) &&; \\ \text{fine-grained:} &\quad N_t \Rightarrow \infty,\ M \Rightarrow \infty,\ \rho \Rightarrow 1 &&. \end{aligned}$$

Coarse-grained machines do not exploit the full parallel potential of our algorithm and are consequently not of interest. It is also fairly clear, *a priori*, that high-order methods will not readily parallelize on fine-grained systems due to communication considerations. We therefore restrict our comparison to high-order/medium-grained couplings *versus* low-order/fine-grained couplings.

We first note that a low-order approach is necessarily going to require many more points to achieve the same level of accuracy as the spectral element method. Hence, the fine-grained system will need many more processors than the number of degrees-of-freedom in the spectral element solution. Second, for each of the fine-grained processors the ratio of communication time to computation time will be relatively large, as the number of words transmitted during direct stiffness summation will be of the same order as the number of calculations required to evaluate the local residual. Third, the messages will of necessity be short on a fine-grained processor, and the advantages of pipelined communication may not be realized.

Next, we consider the issue of mesh refinement. Adaptive resolution is readily implemented in the spectral element/medium-grained algorithm. This is effected by increasing the order of the polynomial approximation in the vicinity of the desired high resolution area (Mavripilis); due to spectral accuracy, an order of magnitude improvement can be obtained at the cost of only a few additional points in each spatial direction. Because of this exponential gain in accuracy, *and* because the work per processor is already large, the incremental work incurred due to refinement will be a fraction of the total work on the processor. Therefore, the processor load imbalance resulting from this localized refinement will not be significant. In contrast, mesh refinement on a fine-grained/low-order implementation will probably require at least a doubling of the number of points on the relevant processors to improve local resolution. The incremental work increase will then be of the same order as the initial work, and the parallel efficiency for an initially well balanced problem will be reduced by a significant factor unless a large-scale load redistribution is carried out.

Ultimately, the choice of parallel architecture and associated algorithm must be coupled to economic considerations which are difficult to predict. The above discussion should only be considered as plausibility arguments for a high-order method/medium-grained coupling.

4. Geometry Defining Processors (GDPs).

Motivation. The use of the hypercube for solution of finite or spectral element equations as described in Section 3 results in efficient parallel solution of partial differential equations (PDE) in three dimensional geometries. However, when applied to this class of problems, the general-purpose hypercube communications network is too flexible, and, at the same time, non-optimal. In particular, as more processors are applied to the solution of a problem, the hypercube's log(M) connections will grow, and eventually exceed, the number required for nearest neighbor communication; these unnecessary connections represent a significant increase in per processor cost and complexity. Furthermore, even with the available log(M) connections, computational problems in complex geometries will require some inter-processor communication to be routed through intermediate nodes, potentially resulting in increased communication latency.

The issue of inter-processor communication raises a parallel processing issue that has not yet been explicitly considered in this paper. This is the mapping problem (Bokhari, 1979), which addresses how spectral (or finite) elements should be distributed onto a particular processor configuration. This mapping is typically performed by a (serial) front-end, which is also responsible for the related activity of mesh generation. A number of computer-aided design systems have been designed to facilitate the transfer of a problem geometry from the physical domain to a computer representation, however the input and display of three dimensional objects and domains via essentially two dimensional tools is often cumbersome, and contributes little to solution of the associated parallel processing mapping problem.

Although the problems of efficient parallel solution and convenient mesh generation appear superficially unrelated, it can be seen that they are, in fact, intimately connected. In essense, mesh generation is related to the problem of specifying domain and connection topology, which is, in turn, a central issue in the subsequent parallel solution of the governing PDEs. In this section we briefly describe a special purpose parallel architecture that simulateously addresses the mesh generation and parallel solution problems by exploiting their underlying similarities (Dewey and Patera, 1987).

<u>Geometry-Defining Processors</u>. Geometry-Defining Processors (GDPs) are microprocessors housed in *physical geometric packages* which can be easily *manually* assembled or reconfigured to define any particular system geometry. An individual GDP is aware of the parameters of its physical package, is able to communicate with neighboring GDPs as well as with a host processor, and is capable of performing independent numerical computations. To solve a PDE, that is, to simulate a physical system, GDPs are assembled in a "scale" model of the actual geometry, as shown in Figure 9. The GDP assembly provides a means for geometry input, and subsequently functions as an optimally-connected dedicated parallel processor for the particular problem at hand.

The basic processor/communication structure of a single GDP is shown schematically in Figure 10 for the simple case of a rectangular two-dimensional GDP. A communication port on each face of the GDP allows for two types of communication within the GDP assembly. First, each GDP is able to communicate with neighboring GDPs through *local* busses. The local bus is reconfigurable, in that the lines emanating from different faces can be internally connected to route messages through the GDP; this allows high-speed, parallel, non-nearest-neighbor communication (e.g., for vector reduction operations (Lin and Sips, 1986)). The second type of communication supported by the GDP faces is a *global* bus which all GDPs and the host computer can access.

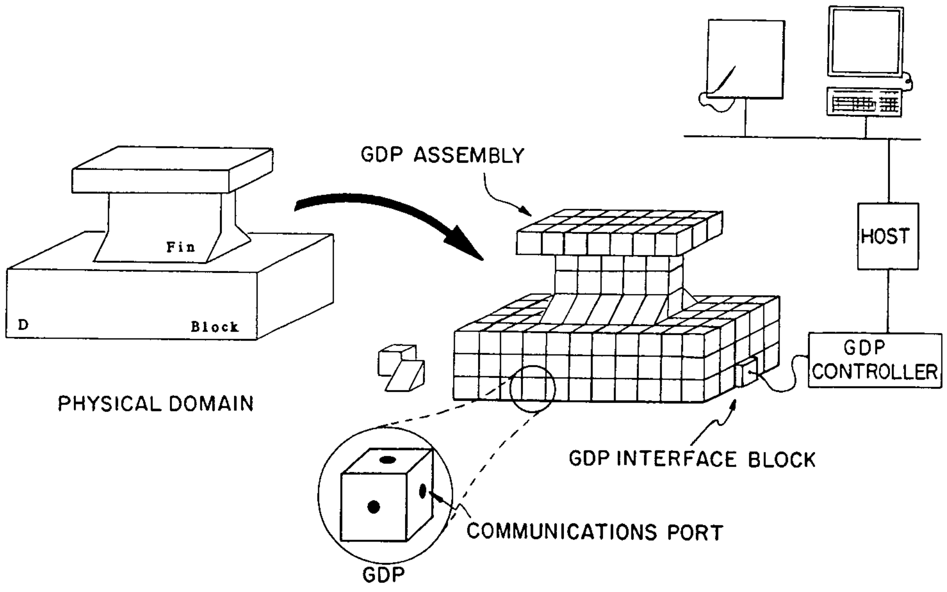

Figure 9. Geometry-Defining Processor assembly corresponding to a physical domain D, which in this case is a block cooled by a heat-sinking fin. The GDP subsystem consists of the assembly of GDPs, a GDP interface block, and a controller through which the host communicates with the GDPs.

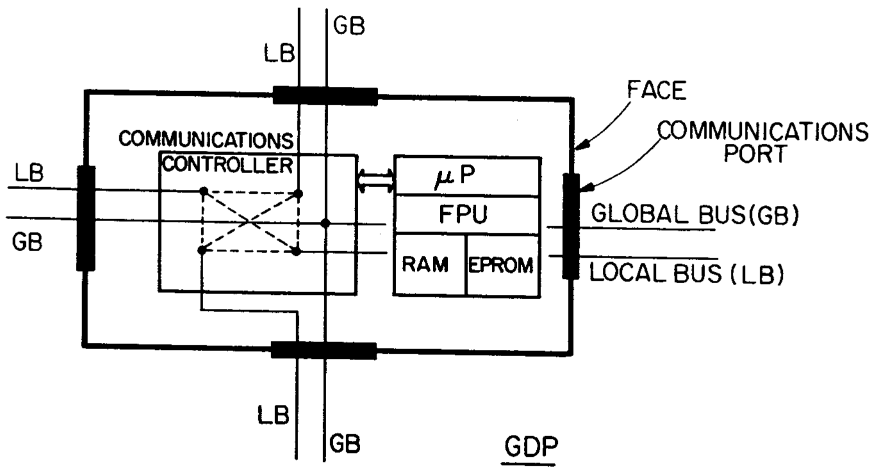

Figure 10. Schematic of a (rectangular, two-dimensional) Geometry-Defining Processor. A GDP consists of a package, communication ports on each face, a microprocessor, a floating point unit (FPU), RAM and EPROM, a communications controller, a global bus (GB), and a reconfigurable local bus (LB).

This bus is primarily used for communication between the host computer and one or more of the GDPs.

Each GDP-resident microprocessor, in addition to controlling communications, performs local calculations related to mesh generation or equation solution using programs and parameters downloaded from the host computer via the global bus. The GDPs in an assembly are synchronized by the global bus, however they otherwise operate independently, performing autonomous calculations on locally-available data.

With the above description of an individual GDP, the following sections present details of the operation of GDP assemblies when applied to mesh generation and parallel solution.

Mesh Generation with GDPs. GDP mesh generation proceeds through several steps of communication and computation. After (manual) assembly of the desired geometry, the host interrogates the assembly of GDPs to determine the number, names, and geometry types of the constituent GDPs. Next, each GDP is instructed to determine and report any connections between itself and its neighbors. By virtue of their physical proximity, and hence communication proximity through the local busses, the information needed to determine these connections is locally available to each GDP. Finally, by "integrating" the geometry-type and connection information obtained from the GDP assembly, the host-resident software can determine the absolute geometry of the assembly.

With a philosophy similar to that of the graphics tablet, the GDPs allow for data input directly in physical space, with all symbolic, computational-space processing performed transparently to the user. Both the minimal learning curve associated with initial use of GDPs, and the minimal time required for subsequent particular realizations of GDP assemblies, should make the devices a useful addition to conventional software geometry-input techniques.

Parallel Solution with GDPs. Geometry-Defining Processor solution of partial differential equations proceeds as follows. The physical domain is first (manually) constructed as shown in Figure 9, and the system geometry is then determined by the host as described in the previous subsection. It should be noted that the geometry as input by the GDPs need only represent a "rough-cut" of the actual system geometry, as conventional computer-aided design tools can be used to subsequently tailor the GDP shapes in software.

Next, each GDP is assigned a single, or more generally, a substructure of finite or spectral elements. Note that the mapping problem has been completely eliminated: with no computational overhead the processor/elements are now optimally connected in that neighboring elements in problem space are represented by communicating processors in computational space. In addition to this nearest-neighbor communication many solution algorithms, such as conjugate gradient iteration, require global summation and broadcast;

these vector reduction operations can be efficiently provided through the use of the reconfigurable local bus. Thus, the GDP assembly next determines a (time-dependent) re-routing of the local busses that will allow these operations to be performed in log(M) time (Dewey and Patera, 1987). Finally, the discrete equations, physical properties, and boundary conditions for each GDP are downloaded from the host processor.

With these mesh generation and equation definition preliminaries performed, the actual parallel solution is then initiated and subsequently synchronized by the host through the global bus. The operations carried out by the GDPs in this solution stage are essentially identical to those now implemented on the hypercube processor as described in Section 3, with the advantage of an optimal connection topology that results in reduced communication latency and greatly reduced hardware and software complexity.

<u>Development of Prototype GDPs</u>. We have developed prototype hardware GDPs and associated host software to perform simple three-dimensional geometry input. The GDP shapes chosen for these prototypes are the cubical and wedge elements shown in

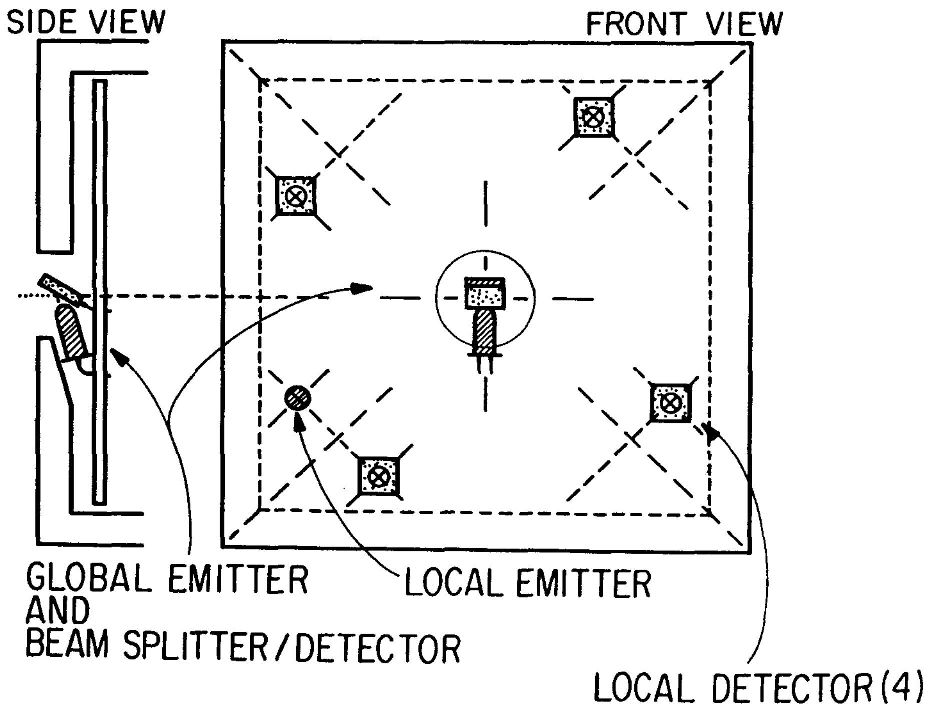

Figure 11. Layout of a typical face of the prototype GDPs, showing the arrangement of emitters and detectors used for local and global communication. The use of four local detectors allows rotational orientation to be determined.

the example of Figure 9. These prototype GDPs have faces outfitted with optical communication ports for the local and global busses; a typical face is diagrammed in Figure 11. Optical emitters and detectors (rather than physical/electrical connections) are used in order to have rotationally independent, easily reconfigurable interfaces. A central port on each face is used for global communication, while the four ports along the periphery of each face allow for local communication and the determination of the rotational orientation of communicating faces. A 16 bit microprocessor with nominal amounts of RAM and EPROM in each GDP controls the communications ports and runs the host-downloaded mesh generation program.

In operation, these prototypes allow simple three-dimensional geometries to be conveniently input and reconfigured. With these prototypes as a starting point, continued research aims to develop sufficient hardware and software to allow for a meaningful evaluation of the GDP concept as a mesh-generation and parallel solution architecture.

Conclusion. In this paper, we have shown how high-order spectral element algorithms coupled to medium-grained message-passing or geometry-defining architectures can result in significant improvements in performance over conventional serial solution methods. The results presented demonstrate that the use of parallel processors for the solution of complex problems is no longer only a promise, but in fact a reality. Future gains in computational power will be the result of continued emphasis on parallelism and algorithm-architecture coupling.

Acknowledgements. We would like to acknowledge our extensive collaborations with Yvon Maday of the Universite Pierre et Marie Curie, and Daniel I. Meiron of the California Institute of Technology. This work was supported by the ONR and DARPA under Contract N00014-85-K-0208, by the NSF under Grant DMC 87-04357, by NASA under Grant NAG1-574, by the Massachusetts Institute of Technology, by Intel Scientific Computers, and by Nektonics, Incorporated.

REFERENCES

(1) S.H. BOKHARI, On the mapping problem, in Proc. 1979 Int. Conf. on Parallel Processing, 1979, p. 239-248.

(2) C. BERNARDI AND Y. MADAY, A collocation method over staggered grids for the Stokes problem, Int. J. Num. Meth. in Fluids, to appear, 1987.

(3) M.O. BRISTEAU, R. GLOWINSKI, AND J. PERIAUX, Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flows, Computer Physics Report, to appear, 1987.

(4) M. DEVILLE AND E. MUND, Chebyshev pseudospectral solution of second order elliptic equations with finite element preconditioning, J. Comput. Phys., 60 (1985), p. 517.

(5) D. DEWEY AND A.T. PATERA, Geometry-defining processors for partial differential equations, in Architecture and performance of specialized computer systems, ed. B.J. Alder, Academic Press, 1987.

(6) G.C. FOX AND S.W. OTTO, Concurrent computation and the theory of complex systems, in Hypercube Multiprocessors, ed. by M.T. Heath, SIAM, 1986.

(7) D. FUNARO, A multidomain spectral approximation of elliptic equations, Num. Meth. for Partial Differential Equations, 2 (1986), p. 187.

(8) N.K. GHADDAR, K.Z. KORCZAK, B.B. MIKIC AND A.T. PATERA, Numerical investigation of incompressible flow in grooved channels. Part 1: Stability and self-sustained oscillations, J. Fluid Mech., 163 (1986), pp. 99.

(9) N.K. GHADDAR, M. MAGEN, B.B. MIKIC AND A.T. PATERA, Numerical investigation of incompressible flow in grooved channels. Part 2: Resonance and oscillatory heat transfer, J. Fluid Mech., 168 (1986), p. 541.

(10) V. GIRAULT AND P.A. RAVIART, Finite element approximation of the Navier-Stokes equations, Springer, 1986.

(11) G.H. GOLUB AND C.F. VAN LOAN, Matrix Computations, Johns Hopkins University Press, 1983.

(12) D. GOTTLIEB AND S.A. ORSZAG, Numerical analysis of spectral methods, SIAM, 1977.

(13) D.S. KERSHAW, The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations, J. Comput. Phys., 26 (1978), p. 43.

(14) H.X. LIN AND H.J. SIPS, A parallel vector reduction architecture, in Proc. 1986 Int. Conf. on Parallel Processing, 1986, p. 503-510.

(15) Y. MADAY AND A.T. PATERA, Spectral element methods for the incompressible Navier-Stokes equations, in State-of-the-Art Surveys in Computational Mechanics (ed: A.K. Noor), ASME, 1987.

(16) Y. MADAY, D.I. MEIRON, E.M. RØNQUIST AND A.T. PATERA, Iterative saddleproblem decomposition methods for the steady and unsteady Stokes equations, in preparation, 1987.

(17) Y. MADAY, A.T. PATERA AND E.M. RØNQUIST, Optimal Legendre spectral element methods for the Stokes semi-periodic problem, in preparation, 1987a.

(18) Y. MADAY, A.T. PATERA AND E.M. RØNQUIST, Optimal Legendre spectral element methods for the multi-dimensional Stokes problem, in preparation, 1987b.

(19) C. MAVRIPLIS, Ph.D. Thesis, Dept. of Aero. & Astro., Massachusetts Institute of Technology, in progress.

(20) J.A. MEIJERINK AND H.A. VAN DER VORST, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comp., 31 (1977), p. 148.

(21) S.A. ORSZAG, Spectral methods for problems in complex geometries, J. Comput. Phys., 37 (1980), p. 70.

(22) A.T. PATERA, A spectral element method for fluid dynamics: Laminar flow in channel expansions, J. Comput. Phys., 54 (1984), p. 468.

(23) E.M. RØNQUIST AND A.T. PATERA, A Legendre spectral element method for the Stefan problem, Int. J. Num. Meth. in Engr., to appear, 1987.

(24) Y. SAAD AND M.H. SCHULTZ, Topological properties of hypercubes, Research Report YALEU/DCS/RR-389, Yale University, New Haven, June 1985.

(25) G. STRANG AND G.J. FIX, An Analysis of the Finite Element Method, Prentice-Hall, 1973.