

P-refinement and P-threads

Steven Dong and George Em Karniadakis *

Division of Applied Mathematics

Brown University

Providence, RI 02912

Submitted to Computer Methods in Applied Mechanics and Engineering

August 15, 2002

Abstract

P-type refinement leads to exponential decay of numerical errors for sufficiently smooth solutions and has been used effectively in turbulence and structural mechanics simulations in the context of spectral and hp finite element discretizations. However, it induces a computational cost of $\mathcal{O}(P^{d+1})$ in d dimensions, which is higher than lower-order methods. In this paper, we demonstrate that by employing multi-threading within MPI processes we manage to counter-balance the cost increase associated with P-refinement. This approach reduces effectively the wall clock time, and keeps it essentially constant as the polynomial order is increased while achieving exponential convergence rate. Since the number of threads within MPI processes can be dynamically adjusted through thread library functions, the algorithm can be readily adapted for dynamic P-refinement. The resulting hybrid MPI/threads dual-level parallelism is particularly suitable for modern supercomputers consisting of “SMP” nodes. We demonstrate this approach in simulations of two three-dimensional fluid dynamics problems.

*Corresponding author, gk@cfm.brown.edu

1 Introduction

Spectral and hp finite element methods, both based on polynomial interpolations, offer dual paths to convergence in numerical solutions of partial differential equations: First by increasing the number of elements while keeping the polynomial order fixed (h-type), and, second by increasing the degree of expansion polynomials while keeping the mesh fixed (p-type). In particular, a combination of both approaches, i.e. hp -refinement leads to very effective simulations and has been used with great success in simulations of structural and fluid mechanics [24, 19, 20, 14, 5].

One desirable property of p-type refinement is its ability to produce exponential decay of the discretization errors for sufficiently smooth solutions, e.g. most of the solutions encountered in elasticity and incompressible flow dynamics. In contrast, the rate of convergence for h-refinement is algebraic; this approach is employed for resolving discontinuities or other singularities present in the data, the geometry or the solution.

More specifically, the effectiveness of P-refinement depends on several other factors, including the h-mesh, i.e. the number of elements and its non-uniformity, the form of geometric singularities such as corners, and the discontinuities in the boundary conditions. These factors degrade convergence because they propagate into the high-order components of the solution that would otherwise be exponentially small. On the other hand, h-refinement is less susceptible to such singularities in the solution domain. To circumvent this problem with P-refinement it is often enough to select a fixed “good” mesh, i.e., one that is sufficiently refined near potential singularities [1, 14], so that these troubling factors are isolated and the error then decreases exponentially. To this end, discontinuous Galerkin methods can also be employed [6].

The fast convergence of P-refinement is achieved at the expense of significantly increased computational cost. The increase in polynomial order results in an algebraic increase of the total degrees of freedom of the system, and even larger increase of the operation count in the associated transforms and matrix manipulations. Specifically, assuming that the number of elements is N and the

polynomial order P , then the computational complexity of the problem is proportional to

$$W_C \propto \mathcal{O}(NP^{d+1})$$

in d -dimensional space. This assumes that the multi-dimensional trial basis has the form of a tensor-product, otherwise the cost would be of $\mathcal{O}(NP^{2d})$ which is totally impractical. In large-scale simulations involving tens of thousands of elements, the computational cost corresponding to P-refinement can quickly become prohibitive. How to effectively reduce the wall clock time associated with P-refinement is the focus of the current paper.

The straightforward approach to accomplish this is to follow a domain decomposition method and use MPI, i.e. re-partition the entire domain to facilitate utilizing more processors. This process is effective for h-refinement but it is difficult to implement in dynamic refinement simulations where significant load imbalance may occur. Moreover, in some situations the computational domain is decomposed only in one direction. In this case, even if the domain is partitioned to the maximum extent, the computational cost associated with a single subdomain can still be too big to be computationally tractable.

In this paper we propose a new approach to tackling the computational cost associated with P-refinement by employing multi-threading within MPI processes. This leads to a hybrid dual-level parallelism. The hybrid programming paradigm (MPI/OpenMP, MPI/Pthreads) combining message passing and shared memory parallelism has emerged in the past few years, largely due to its potential to exploit modern supercomputers consisting of “Symmetric Multi-Processor” (SMP) nodes more effectively than the pure message passing model. It has been applied in applications such as coastal wave analysis [3, 17], atmospheric modeling [15] and molecular dynamics analysis [12]. Model problems and kernel calculations have been used to analyze the performances of the hybrid, pure MPI and pure OpenMP programs, see e.g. [4]. Most of the hybrid implementations, except e.g. [15], apply shared memory parallelism only to the loop level (e.g. [4, 12]) or major subroutines (e.g. [17, 11]). While such fine-grain approach to shared memory parallelism enjoys the

ease of programming, the benefit of multi-threading and scalability is generally also more limited owing to frequent thread creations and destructions as well as the associated synchronizations.

For P-refinement we employ multiple threads in conjunction with domain decomposition with this method. Specifically, as the order of expansion polynomials is increased we deploy multiple threads in each MPI process to balance the increase in the computational cost while keeping the domain decomposition unchanged. This process can be readily made dynamic. The objective of this paper is to introduce the multi-threading algorithm for P-refinement and to demonstrate its effectiveness in the context of three-dimensional time-dependent flow simulations.

About the title: Since the OpenMP constructs and functions are specified on a level higher than those in the commonly available thread libraries, it is more suitable for scientific computations and indeed has been adopted by most hybrid applications including ours. On many systems OpenMP is implemented based on the POSIX thread (P-thread) library (e.g., Compaq True64 Unix, IBM SP). So we use “P-threads” in the title of this paper for reasons of textual symmetry.

2 Spectral/*hp* Element Method and Computational Complexity

In the spectral/*hp* element method the three-dimensional flow variables, e.g. velocities $u(x_1, x_2, x_3)$, are expanded in terms of polynomial basis functions in each element Ω^e , see [14]:

$$u(x_1, x_2, x_3) = \sum_{pqr} \hat{u}_{pqr} \phi_{pqr}(\xi_1, \xi_2, \xi_3), \quad (x_1, x_2, x_3) \in \Omega^e, \quad (1)$$

where

$$\xi_1 = (\chi_1^e)^{-1}(x_1, x_2, x_3), \quad \xi_2 = (\chi_2^e)^{-1}(x_1, x_2, x_3), \quad \xi_3 = (\chi_3^e)^{-1}(x_1, x_2, x_3),$$

are coordinates in the standard (reference) regions. In the above equation \hat{u}_{pqr} are expansion coefficients (called modes), and ϕ_{pqr} are modal basis functions; they are chosen to be Jacobi polynomials element-wise. In analogy to Fourier transform, we refer to $u(x_1, x_2, x_3)$ as representing a *physical*

space variable and \hat{u}_{pqr} as being in *transformed* (or *modal*) space. The number of modes or the total degrees of freedom of the system scales as NP^d , where N is the number of elements, P is the order of Jacobi polynomials, and d stands for the dimension of the elements. An important issue, from the standpoint of computational complexity, is the form of the trial basis $\phi_{pqr}(\xi_1, \xi_2, \xi_3)$. The basis we consider here has a *tensor-product* form even in non-orthogonal domains; it was originally developed by Dubiner [8] and extended in [23] for three-dimensional tetrahedral, prismatic and pyramidal elements.

In solving the Navier-Stokes equations, the most time-consuming computations include the following three types:

1. Transforms between modal and physical spaces,
2. Calculations of first derivatives of flow variables,
3. Poisson and Helmholtz solves.

The pie chart in figure 1 shows the percentage in wall clock time of the three substeps, namely, non-linear, pressure, and viscous substeps in a third-order time integration scheme [13] for the turbulent flow past a circular cylinder at Reynolds number $Re = 3,900$ that will be discussed in section 4.2. The non-linear step is mainly composed of modal-physical space transformations, derivative calculations and inter-process communications. It accounts for the majority of the CPU cycles. The pressure step contains Poisson solves, while the viscous step mainly consists of derivative calculations and the Helmholtz solves.

The backward transformation from modal space \hat{u}_{pqr} to physical space $u(x_1, x_2, x_3)$, as represented by equation (1), involves NP^4 multiplications in three-dimensional space and NP^3 multiplications in two-dimensional space. The forward transformation from physical space to modal space involves solving a system of linear equations element-wise, i.e.

$$\sum_{pqm} (\phi_{rst}, \phi_{pqm}) \hat{u}_{pqm} = (\phi_{rst}, u) \quad \forall r, s, t.$$

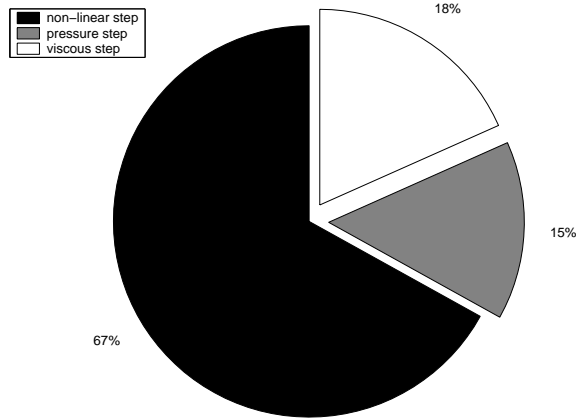


Figure 1: CPU profiles for different substeps of the cylinder flow presented in section 4.2.

where (\cdot, \cdot) denotes inner product. Since the mass matrix (ϕ_{rst}, ϕ_{pqm}) is symmetric and positive definite, a one-time Cholesky decomposition can be performed in the pre-processing stage. The derivatives of flow variables are obtained by multiplying the differential matrix with the flow variable vectors in physical space at the element level, involving $\mathcal{O}(P^{d+1})$ multiplications for each element.

The Poisson and Helmholtz solves are typically performed with a multi-level Schur complement approach [14]. This technique naturally decouples the boundary modes from the interior modes. The reduced set of boundary modes on the coarsest grid is obtained by solving a sparse linear system with either a direct solver or a preconditioned conjugate gradient algorithm. The interior modes are solved subsequently element-by-element using the boundary nodal solutions; this operation is fully parallelizable. Other fast solvers for spectral and hp element methods for flow problems are based on low-energy basis preconditioning and “coarse grid” problems solved in parallel, e.g. see [21], [25], [10], and other references therein.

In summary, for N elements in d -dimensional space with expansion polynomials of order P , the operations per time step involved in solving the Navier-Stokes equations scale as $\mathcal{O}(NP^{d+1})$. Therefore, in P-refinement doubling the polynomial order will increase the computational cost by at least *eight times* in two-dimensions, in contrast to about a factor of four for low-order discretizations (e.g., second-order finite differences). This implies that P-refinement will break even with low-order

methods in terms of efficiency and be better only for simulations requiring high-order accuracy. On the other hand, we can exploit the fact that this extra computational complexity is mostly *local* work consisting of floating point operations to obtain very high parallel efficiencies with suitable parallel paradigms. We explain this in the next section.

3 Multi-threading within MPI Processes — Dual-level Parallelism

To counter-balance the aforementioned increase in computational cost due to P-refinement, we employ *multiple threads* within the MPI processes, giving rise to a hybrid MPI/threads dual-level parallelism. Figure 2 provides a schematic of the hybrid approach with domain decomposition and multi-threading as well as the pure MPI approach. At the outer level multiple MPI processes are responsible for different subdomains of the flow problem. At the inner-level, multiple threads within each MPI process conduct the computations of that subdomain in parallel. Data exchange across domains is implemented with MPI library calls, while conflicts within each subdomain caused by accessing shared objects by multiple threads are resolved with the synchronization primitives provided by the thread library. In contrast, in the pure MPI approach a single thread in the MPI process performs the computations in the subdomain.

Here we consider the hybrid MPI/OpenMP dual-level parallelism for a particular class of three-dimensional unsteady flow problems, in which the flow domain contains at least one homogeneous direction while the non-homogeneous two-dimensional “slice” domain is of arbitrary geometric complexity. Examples of this type of applications include the flow between parallel plates or past long circular cylinders as well as axis-symmetric flows. The flow itself is fully three-dimensional due to intrinsic three-dimensional instabilities but the geometry is homogeneous along that one direction.

A combined spectral/*hp* element-Fourier discretization (see [14]) can be used to accommodate the requirements of high-order as well as the efficient handling of multiply-connected computational

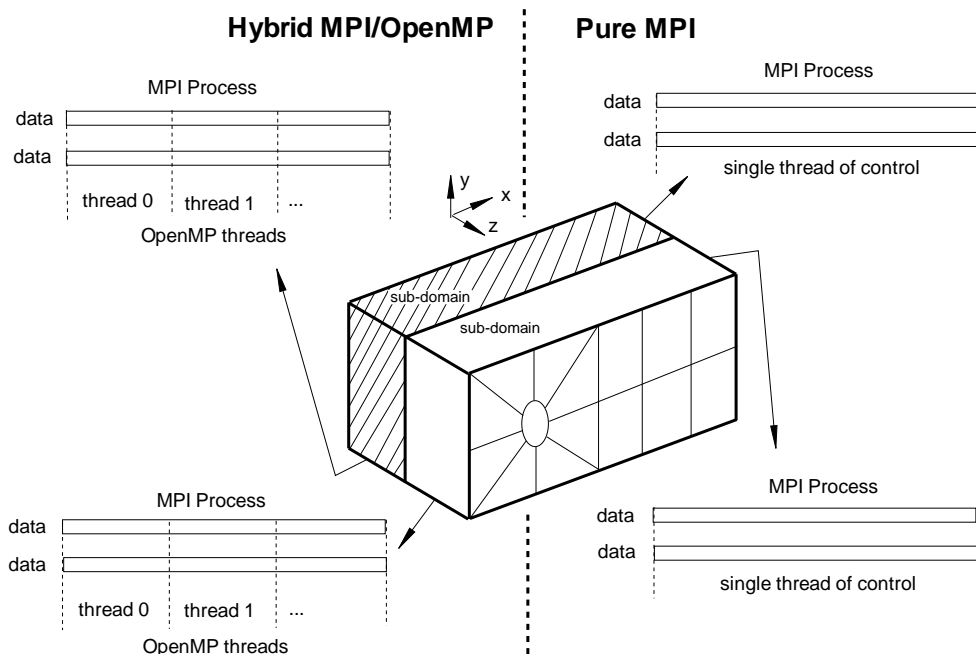


Figure 2: Schematic showing the pure MPI and hybrid MPI/OpenMP paradigms. In both approaches the MPI processes are responsible for different subdomains. However, only a single thread computes in pure MPI while multiple threads split the work in each subdomain in hybrid MPI/OpenMP.

domain in the $x - y$ planes. Using multiple FFTs, the three-dimensional problem is decomposed into two-dimensional problems of the Fourier modes. The flow domain is decomposed in the homogeneous direction. Each MPI process computes a certain number of Fourier modes. Typically, one process is assigned to one Fourier mode corresponding to two (real and imaginary parts) spectral/ hp element planes. Multiple threads share the computations in the two-dimensional spectral element slices within the subdomain.

The multi-threading algorithm is implemented in the high-order general-purpose spectral element CFD code *Nektar* [22, 26] that has been originally parallelized with MPI [7, 9]. We next discuss several issues of particular interest pertaining to the hybrid implementation.

We take a coarse-grain approach to the shared memory parallelism with OpenMP threads in MPI processes. A single parallel region encloses the main time-integration loops at the topmost level. Thus, multiple threads are created after the pre-processing stage. This avoids the overhead associated with frequent thread creations and destructions inherent in fine-grain programs. The number of OpenMP threads within each MPI process is set and can be varied dynamically by invoking the OpenMP library functions. This can be dictated by the expansion polynomial order or by the requirement of offsetting the load imbalance in different subdomains. Thus, multi-threading can be readily used for *dynamic* P-refinement, with the order P of different elements changing as the time integration proceeds to adapt to the flow characteristics.

Multiple threads share the computations within the subdomain by working on disjoint sections of the vectors, disjoint groups of elements, or disjoint computations within the element. Thus, the vector length, the element number and the number of entries in linked lists are split based on the number of threads on the team. For runs with fixed polynomial order these computations can be done only once at the pre-processing stage, and the results are stored in a shared table. Different threads reference the results using their IDs. For dynamic P-refinement the results need to be updated when the polynomial order or the number of threads on the team is changed.

Inter-domain communications are implemented with MPI library calls and are handled by a single thread within each process. Advantageous over pure MPI programs, this configuration consolidates the nodal messages into a single large message and thus reduces the network latency overhead. For the class of applications considered here, the global data exchange with `MPI_Alltoall` is the dominant communication pattern, which is also typical of any $2D$ or $3D$ FFT-based solvers because of its support of the transposition of distributed matrices. Due to the blocking nature of `MPI_Alltoall`, an OpenMP barrier is placed before its call to ensure the completeness of the input data to be exchanged. An implied OpenMP barrier is also associated with the “single” construct after the MPI call.

Two types of OpenMP synchronizations are involved in the hybrid implementation of *Nektar*: *named critical sections* and *barriers* (explicit or implied). Named critical sections are utilized to protect the exclusive write access to the global linked lists containing the roots/weights of various-order Jacobi polynomials, the interpolation matrices of quadrature points, matrices of basis function values on quadrature points, and differential matrices. Read access to these structures is allowed in parallel. OpenMP barriers occur at the beginning of time integration substeps, the beginning and end of MPI calls, the inner products in the iterative solver for boundary-mode solves, and the switching points between global and local operations. The barriers at the switching points between global and local operations account for the majority of OpenMP barriers. However, they can be eliminated completely by careful workload splitting schemes.

Increasing the number of Fourier modes in the homogeneous direction can be accomplished without increasing the amount of memory required per MPI process if the number of processes is increased accordingly. However, the P-refinement in the non-homogeneous two-dimensional spectral element planes (slices) induces an absolute increase in computational cost, and cannot be balanced by increasing the number of MPI processes if each process is already assigned to only one Fourier mode. P-refinement directly causes an increase in the number of modes and quadrature points, the

size of the vectors and various matrices (mass, differential, stiffness), and hence the cost. This cost increase can be compensated by *dynamically* deploying a number of OpenMP threads to share the computations within the MPI processes.

4 Results and Discussions

In this section, we demonstrate the effectiveness of multi-threading for P-refinement with two three-dimensional flow simulations. The first one is a relatively small problem and involves a small mesh and an analytic solution to demonstrate exponential convergence. The second one is a big problem; it involves a direct numerical simulation of the turbulent wake formed behind a circular cylinder. The timing data are collected on the NPACI IBM SP3 at San Diego Supercomputer Center (SDSC).

4.1 Three-Dimensional Analytic Flow

The convergence rate and computation cost of P-refinement are examined first with the steady incompressible flow given by

$$\begin{aligned} u &= e^{\lambda x} \cos \mu y \cos mz, \\ v &= e^{\lambda x} \sin \mu y \cos mz, \\ w &= -\frac{\lambda + \mu}{m} e^{\lambda x} \cos \mu y \sin mz. \end{aligned}$$

This divergence-free velocity field satisfies the Navier-Stokes equations subject to the following forcing

$$\begin{aligned} f_x &= Re^{-1} e^{\lambda x} \cos \mu y \cos mz A_1 + e^{2\lambda x} \cos^2 \mu y A_2 + \\ &\quad \mu e^{2\lambda x} [\cos^2 \mu y \sin^2 mz - \cos^2 mz \sin^2 \mu y], \\ f_y &= 0, \\ f_z &= Re^{-1} e^{\lambda x} \cos \mu y \sin mz B_1 + e^{2\lambda x} \sin 2mz B_2, \end{aligned}$$

where

$$\begin{aligned}
 A_1 &= -\lambda^2 + m^2 - \lambda^3/\mu + \lambda m^2/\mu + \lambda\mu + \mu^2; & A_2 &= 2\lambda + \lambda^2/\mu, \\
 B_1 &= \lambda^3/m - \lambda m + \lambda^2 m/\mu - m^3/\mu + \lambda^2 \mu/ - 2m\mu - \lambda\mu^2/m - \mu^3/m; & B_2 &= \frac{\lambda\mu + \mu^2}{2m}.
 \end{aligned}$$

For the simulations presented here we choose the parameters in the above equations as follows:

$$\lambda = 0.15, \quad \mu = 4\pi, \quad m = 2\pi,$$

$$Re = 1.0.$$

The simulation is performed on the three-dimensional domain $x \in [-1, 1]$, $y \in [0, 1]$ and $z \in [0, 1]$. Periodic conditions are applied in the z direction. The analytic solution is used to impose Dirichlet boundary conditions along the x and y directions. Two Fourier modes are used in z direction, and four equal quadrilateral elements are placed in the $x - y$ planes. The Navier-Stokes equations are integrated in time to obtain a steady-state solution in the interior of the domain. Knowing the exact solution, we can calculate the convergence rate with respect to the polynomial expansion order in $x - y$ planes; the result is shown in figure 3 (left). In this figure we plot the H^1 error as a function of the expansion order for modal quadrilateral spectral elements; exponential convergence of the error is obtained.

The wall clock time per time step for different expansion orders with *one* OpenMP thread per MPI process is collected and plotted as a function of the polynomial order in figure 3 (dashed line). The computational cost increases substantially with respect to the expansion order with a scaling factor 2.34 ($t/t_0 = (P/P_0)^{2.34}$). This is consistent with our analysis in section 2 since spectral elements are used only in the $x - y$ planes. Next, we increase the number of the OpenMP threads per process in proportion to the cost increase for different expansion orders in the single-thread case, and plot the wall clock time per step in the same figure. As we see, multi-threading significantly reduces the wall clock time. Compared with the lowest polynomial order used here ($P = 7$), the

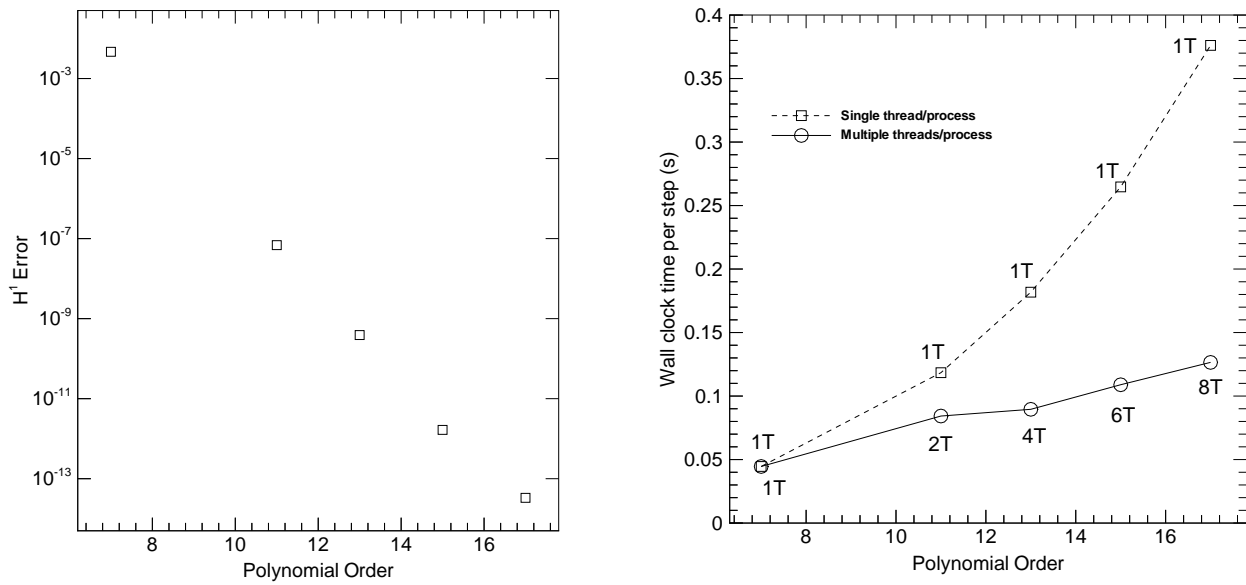


Figure 3: Convergence rate and timings for the three-dimensional analytic flow. Left: H^1 error as a function of polynomial order showing the exponential convergence of P-refinement. Right: wall clock time per step as a function of polynomial order. The number of threads per process is fixed at one in one case, and varied in proportion to the cost increase in another case. The labels indicate the number of threads per process (e.g. $2T$ means 2 threads per process).

N	P	M	C_D	C_L	$-C_{P_b}$	θ_{sep}	L_B/D
902	8	32	1.014	0.021	0.9123	89.83^0	1.254
902	16	32	1.012	0.027	0.9134	90.26^0	1.263

Table 1: Physical quantities in the cylinder flow: drag coefficient C_D , lift coefficient C_L , base pressure coefficient C_{P_b} , flow separation angle θ_{sep} and length of recirculation zone L_B/D . N and P are the number and order of spectral elements in $x - y$ plane, respectively. M is the number of Fourier modes in z direction.

highest order ($P = 17$) induces an increase in computational cost by a factor of *nine* with single thread per process, and a factor of only about *two* with eight threads per process.

Although the above is a small size problem we still manage to achieve substantial wall clock time reduction. In the next section, we will demonstrate that we can actually compute at *constant* wall clock time if the size of the problem is large.

4.2 Turbulent Flow Past a Circular Cylinder

We consider turbulent flow past a long circular cylinder at Reynolds number $Re = 3,900$ based on the inflow velocity and the cylinder diameter. The physics of this flow has been studied systematically in [2, 18]; in the latter reference a parallel computation was involved using an MPI only approach. Figure 5 shows a “z-slice” of the three-dimensional computational domain in the $(x - y)$ plane with 902 triangular prisms (elements). In the homogeneous z direction 32 Fourier modes (i.e., 64 planes) are employed. The flow domain extends from $-15D$ (where D is the cylinder diameter) at the inflow to $25D$ at the outflow, and from $-9D$ to $9D$ in the cross-flow direction. The spanwise length is fixed at $L_z/D = \pi$. A constant *uniform* flow is prescribed at the inflow. Neumann boundary conditions (zero flux) are applied at the outflow and on the sides of domain. Periodic boundary conditions are used in the homogeneous direction.

We use 32 MPI processes in the computations so that each process is responsible for only one Fourier mode (2 spectral element planes). P-refinement is performed systematically in the $x - y$ plane with the polynomial order increasing from 8 to 16 on all elements. This is a typical process in verifying new results in a large-scale flow simulation, i.e. after some successful initial runs P-

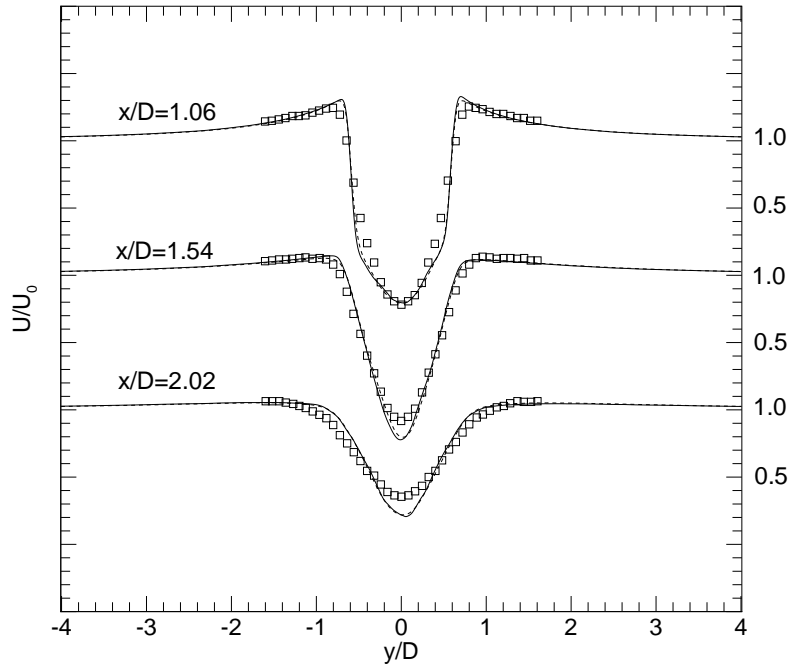


Figure 4: Mean streamwise velocity profiles at $x/D = 1.06, 1.54, 2.02$. Dashed line: order=8; Solid line: order=16; Square symbols: experimental data by Lourenco & Shih [16].

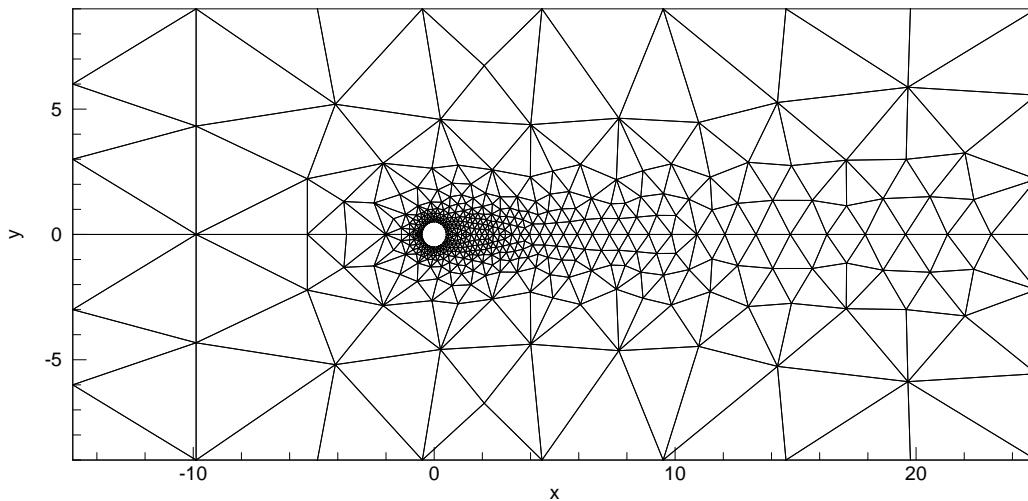


Figure 5: Grid showing the triangular spectral elements of a two-dimensional “slice” for the simulation of turbulent flow past a circular cylinder.

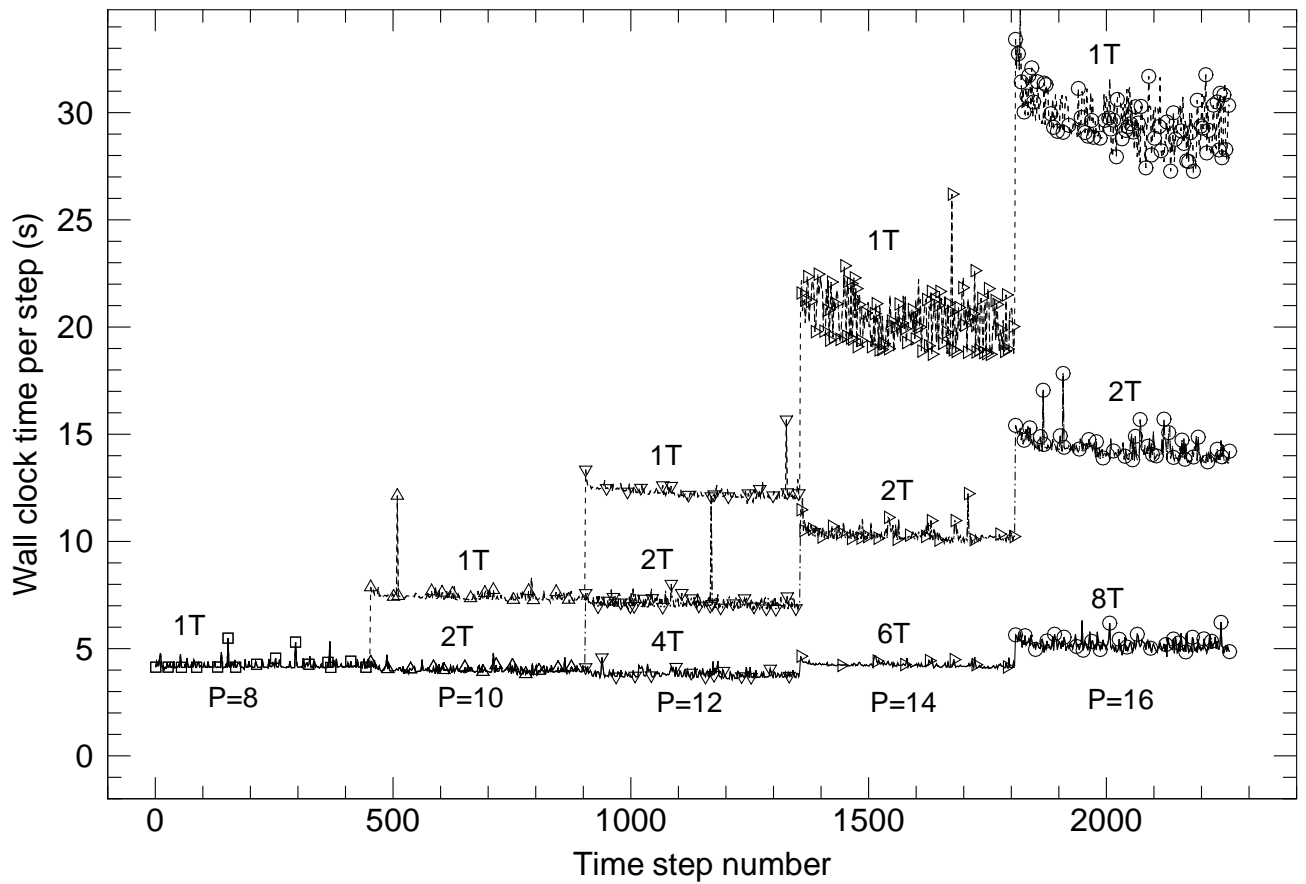


Figure 6: Wall-clock time for P-refinement with one thread per process (dashed line), 2 threads per process (dashed-dot line), and multiple threads per process (solid line). The labels indicate the number of OpenMP threads per process, e.g. “2T” means 2 threads per process.

refinement is undertaken to establish convergence and resolution-independent physical results.

The effect of P-refinement on the numerical results is demonstrated by the flow quantities in table 1 (drag coefficient C_D , lift coefficient C_L , base pressure coefficient C_{P_b} , separation angle θ_{sep} , and length of the recirculation zone L_B/D) obtained from the two extreme cases ($P = 8, 16$), and mean streamwise velocity profiles in figure 4. The differences between the two sets of results are small despite a reduction in numerical discretization error of two orders of magnitude (assuming exponential error decay) from $P = 8$ to $P = 16$.

To demonstrate the effect of multi-threading, the histories of the wall clock time per step for all the polynomial orders are collected. We start with the lowest order $P = 8$, and increase the polynomial order by two each time until the highest order $P = 16$ is reached. For each polynomial order a history of 450 time steps is recorded. The timing data with single thread per MPI process are first collected. Then, as the polynomial order is increased the number of OpenMP threads per process is increased in proportion to the cost increase in the single-thread case. Corresponding to the polynomial orders $P = 8, 10, 12, 14, 16$, we deploy 1, 2, 4, 6 and 8 OpenMP threads per process, respectively.

In reality, the physical “SMP” nodes in supercomputers or workstation clusters usually come with relatively few processors (e.g., the IA64 cluster at NCSA and the ASCI Red at Lawrence Livermore consist of dual-processor nodes while the TCS cluster at PSC is composed of 4-processor nodes). So we also collected the timing data with 2 OpenMP threads per process, the most likely situation in practice. We plot the timing history as a function of the time step for single-, 2- and multiple-thread cases in figure 6. For the case of single thread per process, the wall clock time per step increases from about 4 seconds to about 30 seconds. When two OpenMP threads are used per process, the wall clock time per step is reduced essentially by half. As the number of threads per process increases in proportion, the wall clock time per step decreases dramatically, and essentially remains *constant* for all the polynomial orders.

5 Summary

Multi-threading within MPI processes is proposed to counter-balance the cost increase induced by P-refinement in spectral element and hp finite element methods [5]. As the polynomial order is increased the method can essentially keep the wall clock time *constant* by increasing the number of threads per MPI process in proportion. Because the number of threads within MPI processes can be dynamically adjusted by invoking thread library functions, the algorithm can be readily adapted for dynamic P-refinement. The hybrid MPI/threads dual-level parallelism resulting from this method is also particularly suitable for modern supercomputer architectures consisting of multiple “SMP” nodes. Although the dual-level parallelism proposed here is particular to *hp*-type methods, with small modifications the same paradigm can be employed in other approaches involving either structured or unstructured discretizations.

Acknowledgements

We would like to thank Dr. C. Evangelinos for useful discussions. This work was supported by DOE, ONR, AFOSR, NSF-ITR and DARPA. Computations were performed at Brown’s TCASCV and NPACI’s (SDSC) facilities.

References

- [1] I. Babuska and H.S. Oh. The p-version of finite element method for domains with corners and for infinite domains. *Numer. Meth. PDEs*, 6, 371, 1990.
- [2] P. Beaudan and P. Moin. Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number. Report No. TF-62, Stanford University, 1994.
- [3] S.W. Bova, C. Breshears, C. Cuicchi and Z. Demirbilek. Dual-level parallel analysis of harbor wave response using MPI and OpenMP. *Int. J. High Perform Comput. Appl.*, 14, 49, 2000.

- [4] F. Cappello and D. Etiemble. MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks. *Supercomputing 2000: High Performance Networking and Computing (SC2000)*, 2000.
- [5] Spectral, Spectral Element and hp Methods in CFD, vol. 175, Special Issue of Computer Methods in Applied Mechanics and Engineering, edited by G.E. Karniadakis, 1999.
- [6] B. Cockburn, G. E. Karniadakis and C.-W. Shu. The development of discontinuous Galerkin methods. in *Discontinuous Galerkin Methods: Theory, Computation and Applications*, B. Cockburn, G.E. Karniadakis, C.-W. Shu, editors. Lecture Notes in Computational Science and Engineering, vol 11, Spring 200, Part I: Overview, pp.3-50.
- [7] C.H. Crawford, C. Evangelinos, D. Newman and G.E. Karniadakis. Parallel benchmarks of turbulence in complex geometries. *Computers & Fluids*, 25, 677, 1996.
- [8] M. Dubiner. Spectral methods on triangles and other domains. *J. Sci. Comp.*, 6, 345, 1991.
- [9] C. Evangelinos and G.E. Karniadakis. Communication patterns and models in Prism: A spectral element-Fourier parallel Navier-Stokes solver. *Supercomputing 1996: High Performance Networking and Computing (SC96)*, 1996.
- [10] P.F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. Comp. Phys.*, 133, 84, 1997.
- [11] W.D. Gropp, D.K. Kaushik, D.E. Keyes and B.F. Smith. High-performance parallel implicit CFD. *Parallel Computing*, 27, 337, 2001.
- [12] D.S. Henty. Performance of hybrid message-passing and shared-memory parallelism for discrete event modeling. *Supercomputing 2000: High Performance Networking and Computing (SC2000)*, 2000.

- [13] G.E. Karniadakis, M. Israeli and S.A. Orszag. High-order splitting methods for incompressible Navier-Stokes equations. *J. Comput. Phys.*, 97, 414, 1991.
- [14] G.E. Karniadakis and S.J. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, 1999.
- [15] R.D. Loft, S.J. Thomas and J.M. Dennis. Terascale spectral element dynamical core for atmospheric general circulation models. *Supercomputing 2001: High Performance Networking and Computing (SC2001)*, 2001.
- [16] L.M. Lourenco and C. Shih. Characteristics of the plane turbulent near-wake of a circular cylinder. A particle image velocimetry study. (Unpublished, results taken from Beaudan and Moin (1994)).
- [17] P. Luong, C.P. Breshears and L. N. Ly. Costal ocean modeling of the U.S. west coast with multi-block grid and dual-level parallelism. *Supercomputing 2001: High Performance Networking and Computing (SC2001)*, 2001.
- [18] X. Ma, G.-S. Karamanos and G.E. Karniadadakis. Dynamics and low-dimensionality of a turbulent near wake. *J. Fluid Mech.*, 410, 29, 2000.
- [19] T. J. Oden, A. Patra and Y. Feng. An hp adaptive strategy. *Adaptive, Multilevel, and Hierarchical Computational Strategies*, 157, 23, 1992. ASME.
- [20] A.T. Patera. A spectral method for fluid dynamics: Laminar flow in a channel expansion. *J. Comput. Phys.*, 54, 468, 1984.
- [21] S.J. Sherwin and M. Casarin. Low-energy preconditioning for elliptic substructured solvers based on unstructured spectral/hp element discretization. *J. Comp. Phys.*, 171, 394, 2001.

- [22] S. J. Sherwin and G.E. Karniadakis. A new triangular and tetrahedral basis for high-order finite elements. *Int. J. Num. Meth. Eng.*, 38, 3775, 1995.
- [23] S.J. Sherwin, T.C. Warburton and G.E. Karniadakis. Spectral/ hp methods for elliptic problems on hybrid grids, *Contemporary Mathematics*, 218, 191, 1998.
- [24] B. Szabo and I. Babuska. *Finite Element Analysis*. John Wiley & Sons, Inc. 1991.
- [25] H.M. Tupo and P.F. Fischer. Fast parallel direct solvers for coarse grid problems. *J. Parallel and Distributed Computing*, 61, 151, 2001.
- [26] T.C. Warburton. Spectral/ $h - p$ methods on polymorphic multi-domains: Algorithms and applications. Ph.D. thesis, Brown University, 1999.