Spectral Elements: Adaptivity and Applications in Fluid Dynamics

Ronald D. Henderson

CRPC-TR97696
June 1997

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Submitted June 1997

Spectral Elements: Adaptivity and Applications in Fluid Dynamics[†]

Ronald D. Henderson

California Institute of Technology, Pasadena, CA U.S.A.

1. INTRODUCTION

At the heart of all numerical methods for solving mathematical problems lies the requirement that numerical errors be computed, estimated, or at least bounded in some way. Except for the simplest cases, i.e. linear problems or benchmark cases with known solutions, this is done in a refinement study by systematically varying some parameter related to the accuracy of the numerical method. For example, in structured algorithms like regular finite differences and spectral methods the accuracy of the solution is governed by a single parameter: the mesh spacing, h, or the order of the approximating functions, N. Refinement is easy: just change the single parameter. For the sake of analogy, we compare this to finding the minimum of a function in one dimension that only decreases in one direction.

For multidomain methods (finite elements and related schemes based on domain decomposition) there are M choices that affect accuracy: the size, location, order and so forth, of the computational elements. While systematic refinement is still possible, say by changing only a few of the M parameters at a time, we are now searching in M-dimensions for a minimum of some possibly complex function. One broad definition of "adaptive refinement" is an algorithm that relates an indicator of accuracy to the M parameters of the numerical solution, so that by changing the size of a single parameter we can change the M parameters of the discretization in a well-defined, consistent way. As both scientific and engineering calculations address increasingly complex problems in solid and fluid mechanics, adaptivity becomes a vital tool for guaranteeing both accuracy and efficiency.

This paper gives a short description of various concepts related to adaptivity in high-order finite element methods. Specific examples are presented for *spectral* elements—discretizations that incorporate ideas from spectral methods such as tensor-product polynomial expansions. We place special emphasis on methods that incorporate task parallelism as a fundamental part of the discretization.

2. ADAPTIVITY

Mesh refinement can be classified as one of three broad types. First is *mesh movement*, or simply redistributing the computational elements in a mesh. This is the easiest type of refinement but it is quite limited, and usually results in highly distorted meshes. Second is p-refinement, accomplished by increasing the order

[†]Presented at the Int. Conf. Comp. Eng. Sci., July 31–August 3, 1995. Mauna Lani, Hawaii.

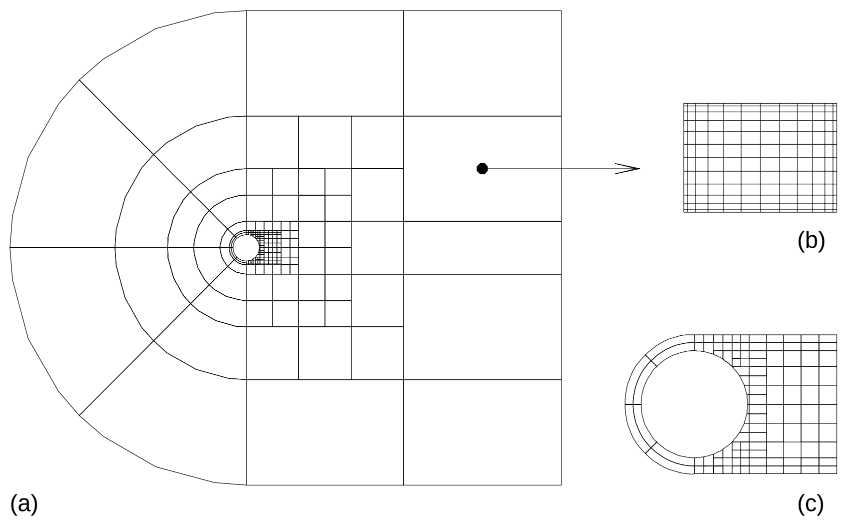

Figure 1: A typical unstructured spectral element mesh used in flow calculations: (a) global domain partitioned into 184 elements; (b) interior mesh on a single element corresponding to a 12th order polynomial basis; (c) close-up of the element distribution near the body.

of the approximating functions. If the solution is sufficiently smooth (an important restriction), p-refinement reduces errors exponentially fast. If done locally it presents some practical problems associated with re-mapping the surrounding data, so here we only consider p-refinement of a global nature. Last is h-refinement, which amounts to splitting computational elements into elements of a smaller size. This is the approach generally followed in adaptive finite element methods and probably the most powerful. Figure 1 shows a spectral element discretization of the two-dimensional space around a circular cylinder, generated by successive h-refinements of an initial coarse mesh. During convergence tests, p-refinement is performed by changing the polynomial order in each element simultaneously.

In the target application (fluid dynamics), we consider two possible strategies for adaptivity. In a steady problem, adaptivity can be an integral part of the solution algorithm: the goal is to solve the problem with a given level of accuracy one time and stop. Since the mesh is constantly changing we use iterative methods exclusively. In computing the solution to an unsteady problem, the goal is to maintain a given level of accuracy but proceed as fast as possible. In this case we can invest more time in pre-processing so the solution phase runs faster, possibly using direct rather than iterative solvers.

3. ALGORITHMS

User input to a numerical boundary value problem is generally a domain, source terms, and boundary conditions. The challenge within a computer code is how to put all this together, i.e. how to manage a flat array of data that really represents an approximation to the solution on a complex two- or three-dimensional domain.

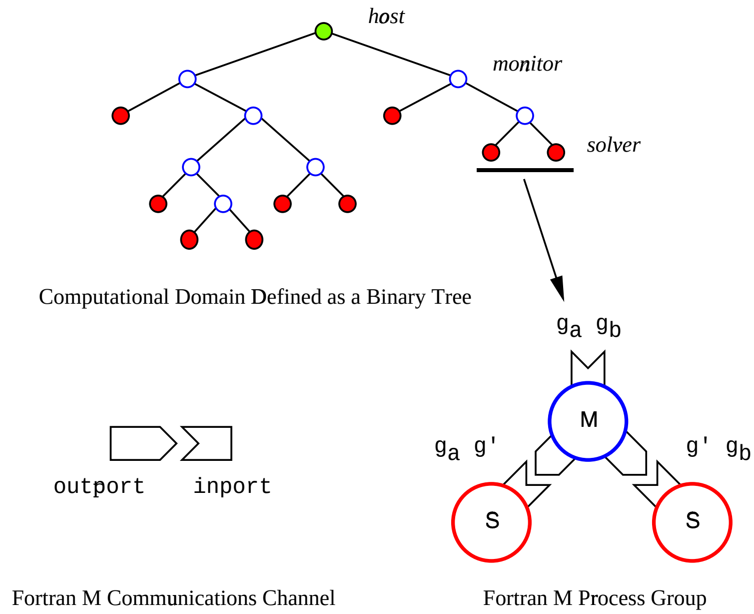

Figure 2: Basic components of an adaptive calculation expressed as a collection of parallel tasks coordinated through the Fortran M abstraction of a channel.

This is what makes finite element methods so complicated and at the same time so powerful: discrete equations based on finite elements can be generated on arbitrarily complex domains, but at the cost of highly irregular data structures.

Here we present the basic data structure used to organize the computational elements that make up a spectral element approximation as a *tree*. Figure 2 shows a binary tree for a one-dimension problem; the equivalent structure would be a quad- or oct-tree in two or three dimensions. The tree is used to maintain an index set that maps a flat array of nodal values to the correct geometric position in the mesh. There are a number of advantages to this data structure: it is good for searches and global decision making; refinement takes place only at the "leaves"; it has a natural hierarchical structure for multigrid-type solution algorithms; and it is simple to program because operations can be expressed recursively. In two and three dimensions a tree needs multiple roots to represent complex geometries, but the basic concepts are the same.

Another advantage is that solution algorithms based on this data structure can be implemented in a natural way using task-oriented languages like Compositional C++ and Fortran M [1]. Nodes in the tree are implemented as two types of tasks: *monitors* for error tracking and decision making, and *solvers* to handle the calculations associated with a computational element of the mesh. In the programming model, all of the these tasks execute independently and in parallel. Tasks are coordinated through a set of *channels* that allow communication between any pair of tasks. When refinement takes place in the one-dimensional case, a solver switches to the role of a monitor and spawns two new solvers to handle the refined element.

4. APPLICATIONS

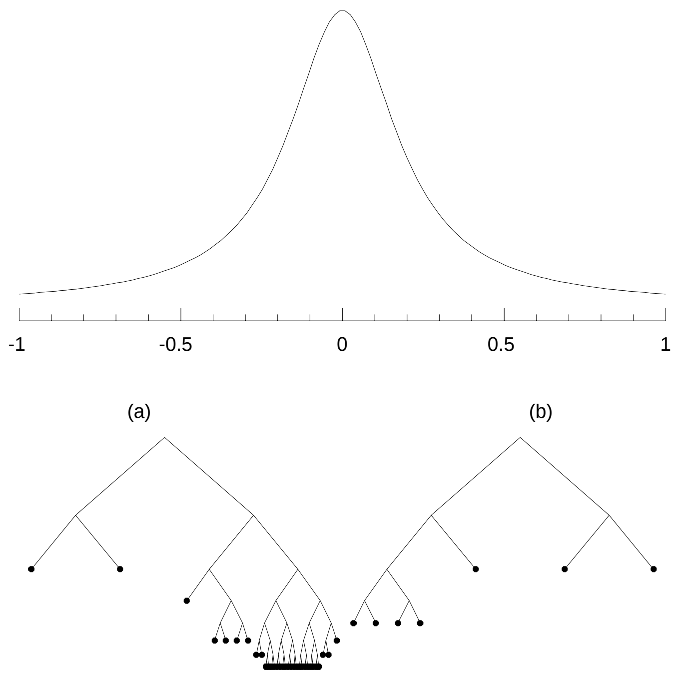

Figure 3: Combined h-p refinement for the solution of a one-dimensional Poisson equation, $u(x) = 1/(1 + 25x^2)$: (a) refinement tree with a 4th order basis in the left half of the domain; (b) refinement tree with an 16th order basis in the right half.

Although the interesting applications are in two and three dimensions, we start with a simple one-dimensional problem to further explain some of the basic ideas and demonstrate the advantages of h- versus p-refinement. The function shown in figure 3 was computed by solving the Poisson equation with appropriate forcing and Dirichlet boundary conditions over the domain $-1 \leq x \leq 1$. This function is symmetric about $x = 0$, and the spectral element solution is restricted to $N = 4$ in the left half of the domain and $N = 16$ in the right half. Adaptivity is based on both the gradient of the solution and how it changes with refinement: if $||du/dx||$ over an element is too large the element is split, and if the refinement produces no change in $||du/dx||$ integrated over the same area then the solution has been resolved locally. Refinement based on estimated local truncation error [2] gives similar results. Parts (a) and (b) of figure 3 show the mesh represented as a binary tree, with each node '•' terminating inside an element. Each subtree represents pure h-refinement and left versus right represents global p-refinement. The high-order solution is 10 000 times more accurate with 10% fewer nodes. For a simple problem like this example, "high-order" corresponds to much higher accuracy and lower cost, but this depends completely on the smoothness of the solution.

The second example is a solution to the incompressible Navier–Stokes equations for flow around an impulsively-started cylinder (figure 4). Solutions to this set of nonlinear equations depend on the *Reynolds number*, a nondimensional parameter defined as $Re \equiv Ud/\nu$, where U is the final velocity of the cylinder, d is its diameter, and ν is the kinematic viscosity of the fluid. This calculation was performed on a

sequence of meshes, and the final one is shown in figure 1. Note that in dimensions higher than one, local refinement produces nonconforming elements that require special treatment, although it is possible to construct patching schemes that allow nonconforming systems to be solved as efficiently as those used in standard Galerkin methods [3]. The calculation determines the "primitive variables", velocity $\mathbf{u}$ and pressure p, by integrating Navier–Stokes with a high-order splitting scheme.

In this application we are interested in computing the unsteady drag on the cylinder and relating it to the evolving vorticity field, $\xi_z = \text{curl } \mathbf{u}$. The baseline for comparison is a high-resolution vortex particle calculation [4]. Figure 5 shows the unsteady drag coefficient from the three methods indicated. The spectral element calculation on an initial course grid (based on the element size) still shows mesh dependence at $N = 14$th order; successive mesh refinement based on the local magnitude of ξ_z produced the mesh shown in figure 1, with full convergence at $N = 12$th order (approximately 30 000 mesh points). In contrast to the simple one-dimensional Poisson equation, solutions that exhibit strong local behavior, like the boundary layers in this example where ξ_z varies rapidly, depend critically on h-refinement for both accuracy and efficiency.

5. CONCLUSIONS

There are a number of issues we have not considered in this brief overview: load balancing of task-oriented computations, communication costs, and various other performance issues. More important to the issue of accuracy are error estimates or indicators, especially as they relate to the decision for pursuing h- versus p-refinement of a solution [2, 5]. Each of these is the subject of current work.

This work is supported by the National Science Foundation under Grant No. CDA-9318145.

References

[1] Ian Foster. *Designing and Building Parallel Programs.* Addison-Wesley, 1995.

[2] Catherine Mavriplis. Adaptive mesh strategies for the spectral element method. *Comput. Methods Appl. Mech. Engrg.*, 116:77–86, 1994.

[3] Ronald D. Henderson and George Em Karniadakis. Unstructured spectral element methods for simulation of turbulent flows. *J. Comput. Phys.*, 1995. (*to appear*).

[4] P. Koumoutsakos and A. Leonard. High resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 1995. (*to appear*).

[5] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward a universal *h-p* adaptive finite element strategy, part 1. constrained approximation and data structure. *Comput. Methods Appl. Mech. Engrg.*, 77:79–112, 1989.

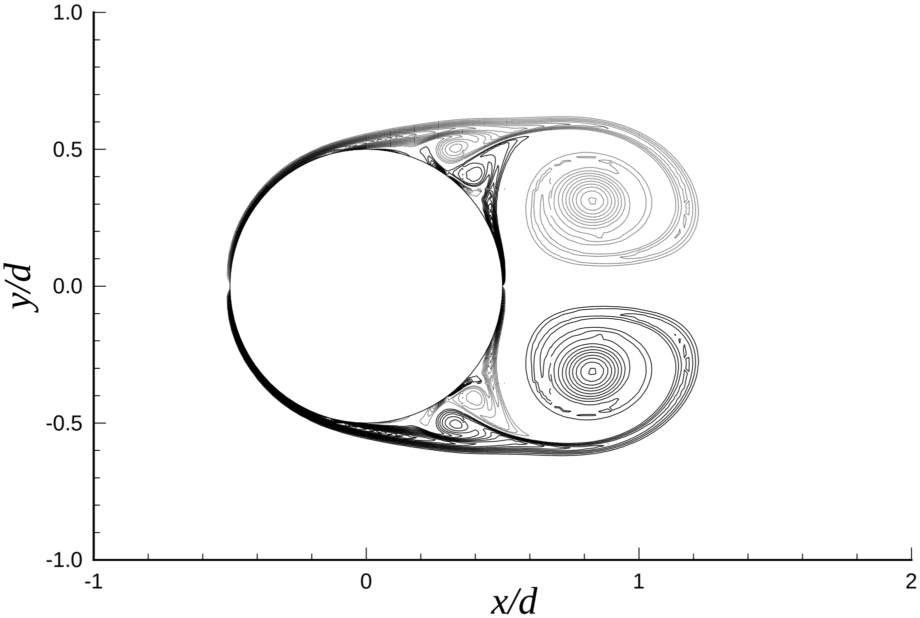

Figure 4: Instantaneous vorticity field in the near wake of an impulsively-started cylinder at $tU/R = 3$, $Re = 3000$; contours are evenly spaced over $-130 \leq \xi_z \leq 130$. Flow calculations were performed using the mesh shown in figure 1.

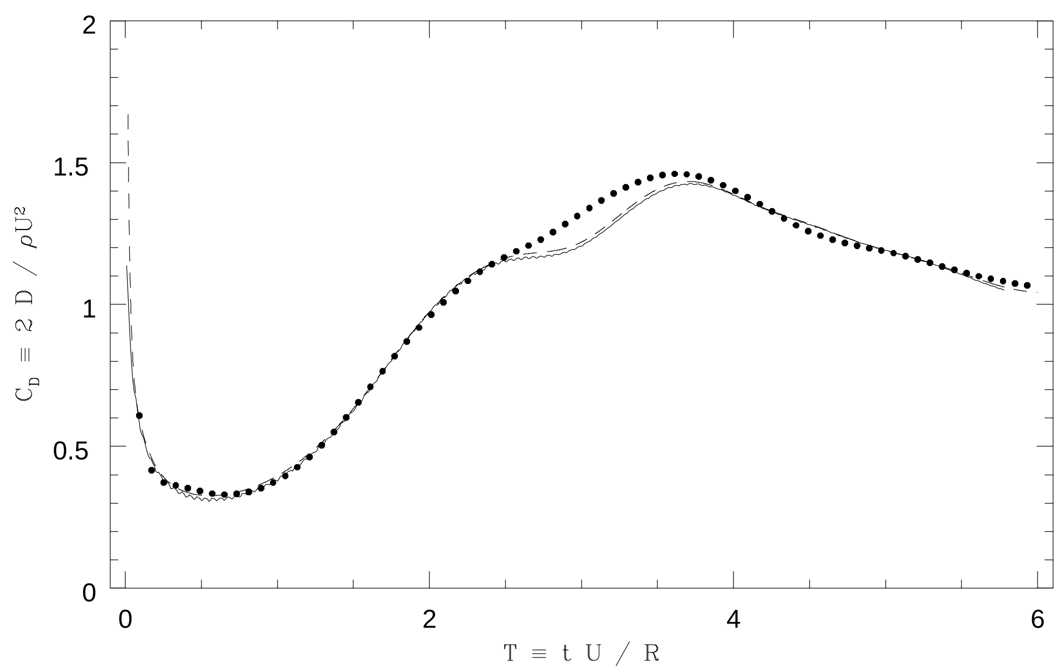

Figure 5: Unsteady drag coefficient: $\bullet$, initial course-grid calculation at $N = 14$th order; (dashed line), refined calculation at $N = 12$th order, giving 3×10^4 final mesh points; (solid line) vortex particle calculation with 3×10^5 particles.