#543

An $O(n \log n)$ solution algorithm for spectral element methods

I. Lee [a], P. Raghavan [a], S. Schofield [b], P. Fischer [b,*]

[a] *Department of Computer Science and Engineering, The Pennsylvania State University, 220 Pond Lab, University Park, PA 16802-6106, USA*

[b] *Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439-4844, USA*

Abstract

To leverage significant software development effort, general purpose unstructured codes are often used in structured or semi-structured applications. We show that $O(n \log n)$ computational complexities, competitive with classic Fourier methods, are achievable for some classes of semi-structured spectral element applications.

Keywords: PLEASE PROVIDE KEYWORDS

?#1

1. Introduction

Many three-dimensional flow problems in science and engineering feature geometries that are homogeneous in at least one flow direction. Two examples are the high-aspect-ratio domains in Fig. 1, which shows spectral element (SE) meshes currently being used for simulations of Rayleigh–Bénard convection (a) and reactor core cooling (b). In these cases, the numerical simulation costs can often be reduced by recasting the original problem (or subproblem) in terms of eigenvectors of the (discrete) one-dimensional operators to yield a set of decoupled problems of lower dimension. Here, we consider application of this approach to reduce three-dimensional problems to independent two-dimensional subproblems. The costs of performing the transformations and solving the subproblems is addressed, and we show that it is possible to achieve $O(n \log n)$ complexity for n-gridpoint problems in $\mathbb{R}^3$.

2. Problem formulation

We consider the Poisson equation, which is representative of the pressure solve, the stiffest substep in our Navier–Stokes solver [1]. Let the domain of interest be $\Omega := [0, L_z] \times \Omega_{2D}$, with Ω_{2D} invariant with respect to z. The SE method is based on the weak formulation: Find $u \in X^N$ such that $(\nabla v, \nabla u) = (v, f)\ \forall\ v \in X^N$, where X^N is the finite-dimensional set of SE basis functions that vanish on the boundary $\partial\Omega$. Let

$$f(x,y,z) = \sum_{j=1}^{n_z} \sum_{i=1}^{n_{xy}} f_{ij} \phi_i(x,y) \psi_j(z),$$

$$(f,g) = \int_0^{L_z} dz \int_{\Omega_{2D}} fg\, dx\, dy,$$

denote respectively the functional form of elements in X^N and the L^2 inner-product on Ω. Inserting these into the weak formulation gives rise to the matrix problem $A\underline{u} = B\underline{f}$, where $\underline{u}$ is the vector of $n = n_z n_{xy}$ unknown basis coefficients, $A = B_z \otimes A_{2D} + A_z \otimes B_{2D}$, $B = B_z \otimes B_{2D}$, and

$$(A_z)_{ij} := \int_0^{L_z} \frac{d\psi_i}{dz} \frac{d\psi_j}{dz} dz, \qquad (B_z)_{ij} := \int_0^{L_z} \psi_i\, \psi_j\, dz,$$

$$(A_{2D})_{ij} := \int_{\Omega_{2D}} \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y}\, dx\, dy,$$

$$(B_{2D})_{ij} := \int_{\Omega_{2D}} \phi_i\, \phi_j\, dx\, dy.$$

(We recall that the tensor, or Kronecker, product $C = A \otimes B$ denotes the block matrix with entries $a_{ij} B$.)

Under the tensor-product assumptions, the discrete problem can be cast as

$$A\underline{u} = (S^T \otimes I)^{-1} (I \otimes A_{2D} + \Lambda_z \otimes B_{2D})(S \otimes I)^{-1} \underline{u}$$
$$= B\underline{f} =: \underline{g}, \tag{1}$$

* Corresponding author. Tel.: +1 630 252 6018; Fax: +1 630 252 5649; E-mail: fischer@mcs.anl.gov

Computational Fluid and Solid Mechanics 2003
K.J. Bathe (Editor)

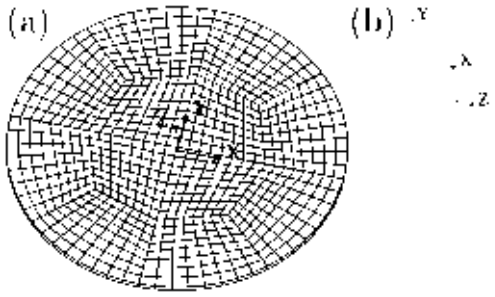

Fig. 1. Tensor-product mesh examples for Rayleigh–Bénard convection (a) and reactor core cooling (b).

where $S = (\underline{s}_1, \underline{s}_2, \ldots, \underline{s}_{n_z})$ and $\Lambda_z = \mathrm{diag}(\lambda_k)$ contain the eigenpairs satisfying $A_z \underline{s}_k = \lambda_k B_z \underline{s}_k$ and $\underline{s}_i^T B_z \underline{s}_j = \delta_{ij}$. The advantage of (1) is that the matrix $I \otimes A_{2D} + \Lambda_z \otimes B_{2D}$ is block diagonal and thus represents n_z independent two-dimensional subproblems of size n_{xy}.

Solution of (1) involves three steps: $\underline{w} = (S^T \otimes I)\underline{g}$; solution of $(A_{2D} + \lambda_k B_{2D})\underline{v}_k = \underline{w}_k$, for $k = 1, \ldots, n_z$; and $\underline{u} = (S \otimes I)\underline{v}$. If we view the basis coefficients as matrices, then the transformation from physical to wave space,

$$w_{ik} = [(S^T \otimes I)\underline{g}]_{ik} = \sum_{j=1}^{n_z} S_{kj}^T g_{ij} = \sum_{j=1}^{n_z} g_{ij} S_{jk} = (GS)_{ik}, \quad (2)$$

and the inverse transform, $U = VS^T$, are simply matrix–matrix products, which are very efficient on all architectures. If $n_z \ll n_{xy}$ (Fig. 1a), then the cost of applying S or S^T is subdominant to that of solving the two-dimensional subproblems. However, if n_z is relatively large compared with n_{xy} (Fig. 1b), then the cost of this step can become prohibitive, especially in the multiprocessor case, where it is desirable to store the $n_z \times n_z$ matrix S on each processor. One can significantly reduce this cost if the one-dimensional problem in the z direction has periodic boundary conditions and is discretized with K_z spectral elements of uniform size and order N_z. Under these conditions, A_z has the block-circulant form $A_z = I \otimes \tilde{A} + \Pi \otimes \Gamma + \Pi^T \otimes \Gamma^T$, where $\Pi = (\underline{\hat{e}}_{K_z}, \underline{\hat{e}}_1, \ldots, \underline{\hat{e}}_{K_z-1})$ is a cyclic permutation of the columns of the identity, $\underline{\hat{e}}_j$, and $\tilde{A}$ and Γ are $N_z \times N_z$ blocks. Because Π and Π^T are diagonalized by the Fourier matrix [2], it is possible to evaluate (2) in only $O(nN_z \log K_z)$ operations. Most important, the storage for S is reduced by a factor of K_z.

3. Two-dimensional solves

The reduced SE problems are solved by using conjugate gradient iteration. As originally proposed by Deville and Mund [3], we precondition these problems with a system A_{FE} that is based on a finite-element discretization having nodes coincident with the SE nodes. Finite-element preconditioning gives bounded iteration counts, so the approach is scalable provided that the preconditioner cost is low. Fortunately, for the two-dimensional case, the sparse matrix problems arising from the low-order discretization can be solved in $O(n_{xy} \log n_{xy})$ time using nested-dissection orderings [4].

Consider the solution of $A_{FE}\underline{x} = \underline{b}$ using a direct method when A_{FE} is sparse and symmetric positive definite. A sparse Cholesky factorization $A_{FE} = LL^T$ is computed and the solution obtained by using forward and backward sparse triangular solutions $L\underline{y} = \underline{b}$ and $L^T\underline{x} = \underline{y}$. The major difficulty with sparse Cholesky factorization is *fill-in*, namely, zeros in the original matrix that become nonzeros in the factors. Fill-in is controlled by computing a suitable permutation using graph-theoretic methods in a first ordering step; 'nested dissection' orderings result in optimal fill-in (to within a constant factor) for sparse matrices associated with finite-difference and finite-element grids [4]. Ordering is followed by a symbolic factorization to determine the nonzero structure of L. Both these steps are relatively inexpensive and are followed by the numeric factorization step, which dominates the cost. The cost of triangular solution using the factor is of lower order, and on serial machines it is quite negligible compared with the cost of factorization.

Parallel sparse numeric factorization can be performed effectively on multiprocessors using either multifrontal or column-block methods [5,7,9], which leave the factors distributed across the processors, ready for the next solution step. Parallel triangular solution involves tree-structured computation; task parallelism is exploited by allocating disjoint subtrees to each processor for local phase computations. Computations at a vertex i involve all processors assigned local phase subtrees rooted at descendants of i. Columns of L associated with vertex i are stored as a small dense matrix L_i which can be partitioned into the triangular part $\tilde{L}_i$ and an update part $\tilde{U}_i$. The system used for forward solution has the form $\tilde{L}_i \underline{y}_i = \underline{b}_i$, where $\underline{b}_i$ contains the corresponding components of the right-hand side vector. The vector $\underline{y}_i$ is then used to compute $\underline{z}_i = \tilde{U}_i \underline{y}_i$, an update to the right-hand side vector components associated with the system at the parent of node i. The backward solution step

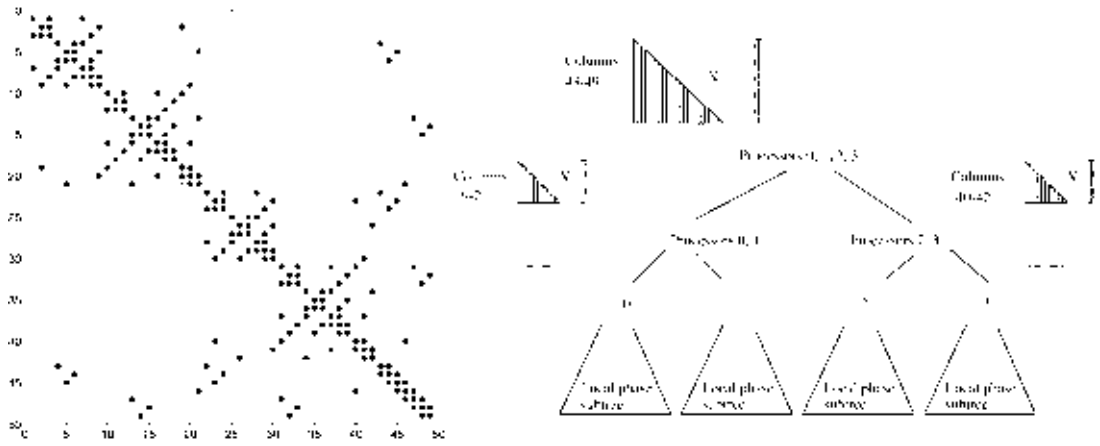

Fig. 2. Structure of $L + L^T$ associated with the matrix of a 7×7 five-point finite-difference grid ordered using nested dissection (left). Forward triangular solution is shown using 4 processors $0, \cdots, 3$ (right). Shaded triangular submatrices are inverted once to replace distributed substitution by matrix-vector multiplication in subsequent triangular solutions.

is a direct analog with the computation proceeding down from the root to the leaves of the compute tree.

On multiprocessors, the columns of the matrix L_i are mapped to processors to share the arithmetic work and memory requirements. When the latency of communication is relatively large, distributed dense substitution for solving $\tilde{L}_i \underline{y}_i = \underline{b}_i$ becomes a performance bottleneck. Even with the use of sophisticated pipelining techniques very large matrices are required to achieve good efficiency [6]. For sparse matrices from finite-difference and finite-element schemes in two space dimensions, however, the size of largest dense submatrix is relatively small (proportional to the square root of the matrix dimension). Thus, on parallel multiprocessors, if the triangular solution is not performed efficiently, a few solutions can easily require more time than factorization. To address this problem, we developed latency-tolerant *selective inversion* (SI) in which the inverse of $\tilde{L}_i$ is computed and stored explicitly at each vertex i in the compute tree [8,10–12]. Since $\underline{y}_i$ is given by $\tilde{L}_i^{-1} \underline{b}_i$, we therefore replaced substitution with matrix-vector multiplication, which is highly parallel. For this application, we use SI-2, the SI scheme with a two-way data mapping [12] implemented in our solver package [9]. Fig. 2 relates a tree-structured four-processor SI-based scheme to the Cholesky factors of a sparse matrix associated with a 7×7 five-point finite-difference grid.

We next provide some complexity results for our SI implementation. We assume that the multiprocessor has the following model of communication costs. The cost of sending m words in a single message is $\alpha + \beta m$, where α is the startup (or latency) and β is the per word transfer cost (α is typically orders of magnitude larger than β). We also assume that a global reduction operation with P processors can be implemented with $\log_2 P$ messages using a multidimensional hypercube type algorithm. We now state the following results adapted from [11,12].

Lemma 1 *Consider the sparse linear system of dimension n_{xy} associated with the model five-point, finite-difference two-dimensional $\sqrt{n_{xy}} \times \sqrt{n_{xy}}$ grid. Assume that a nested dissection ordering is used for fill-reduction and that the triangular solution is performed by using P processors with the SI-2 scheme. The communication costs at a processor are*

$$\frac{\alpha}{2}\{(log_2 P)^2 + \log_2 P\} + 3\beta\sqrt{n_{xy}}(\log_2 P).$$

The results in Lemma 1 also hold for sparse matrices associated with two-dimensional finite-element schemes. The number of messages and hence the latency cost remain unchanged. The communication volume is of the form $c\sqrt{n_{xy}}(\log_2 P)$, where c is a constant that depends on the type of elements used and the order of the method. Observe that for SI-2, the latency cost of $\frac{\alpha}{2}\{(log_2 P)^2 + \log_2 P\}$ is independent of the matrix dimension (n_{xy}). Consequently, the performance of SI-2 is significantly better than that of a traditional substitution-based scheme with latency costs that grow as $O(\sqrt{n_{xy}}/P)$.

4. Conclusions

We have presented an approach for achieving $O(n \log n)$ solution complexity for three-dimensional problems discretized with the spectral element method or other tensor-product bases. Initial numerical tests indicate that this theoretical complexity estimate can be realized in practice.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Dept. of Energy, under Contract W-31-109-Eng-38; by the Computational Sciences Graduate Fellowship Program and Maria Goeppert Mayer Award from the U. S. Department of Energy and Argonne National Laboratory; and by the National Science Foundation through grants ACI-0102537, DMR-0205232, and EIA-0221916.

References

[1] Fischer PF. An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. J Comp Phys 1997;133:84–101.

[2] Davis PJ. Circulant Matrices. Chelsea Publishing, New York, 1994.

[3] Deville MO, Mund EH. Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning. J Comp Phys 1985;60:517–533.

[4] George JA, Liu JWH. Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[5] Gupta A, Kumar V. A scalable parallel algorithm for sparse matrix factorization. TR 94-19, Dept. of Comp. Sci., University of Minnesota, Minneapolis, 1994.

[6] Heath MT, Romine CH. Parallel solution of triangular systems on distributed-memory multiprocessors. SIAM J Sci Stat Comput 1988;9:558–587.

[7] Heath MT, Ng E, Peyton BW. Parallel algorithms for sparse linear systems. SIAM Rev 1991;33:420–460.

[8] Heath MT, Raghavan P. Parallel sparse triangular solution. In: Gulliver R, Heath M, Schreiber R, Bjorstad P (Eds), IMA Volumes in Mathematics and Its Applications (105). Springer-Verlag, Berlin, 1998.

[9] Raghavan P. DSCPACK: A domain-separator code for the parallel solution of sparse liner systems. TR CSE-02-004, Dept. of Comp. Sci. and Eng, The Pennsylvania State University, 2002.

[10] Raghavan P. Efficient parallel sparse triangular solution using selective inversion. Parallel Processing Lett 1998;8(1):29–40.

[11] Raghavan P, Teranishi K, Ng E. Towards scalable preconditioning using incomplete factorization. Proc. of 2001 Int. Conf. on Preconditioning Techniques for Large Sparse Matrix Problems in Industrial Applications, Tahoe City, CA, April 29–May 1, 2001.

[12] Teranishi K, Raghavan P, Ng E. A new data-mapping scheme for latency-tolerant distributed sparse triangular solution. Proc. of Supercomputing 2002, Baltimore, Nov. 2002.