55

Parallel Implementation of the Spectral Element Method with Nonconforming Mesh

H.Feng[1], C.Mavriplis[2]

The Nonconforming Spectral Element Method (NSEM) solves PDEs in complex geometries with high accuracy, however, it is an expensive method. Since parallel computation is effective in decreasing CPU time, a parallel algorithm for the NSEM is presented. Implementations on SGI Power Challenge using MPI are evaluated in terms of measured speedup and parallel efficiency for schemes of one element and multiple elements per processor.

Introduction

The Spectral Element Method (SEM) [Pat84] is a high order discretization scheme for the solution of nonlinear PDEs. The method draws its strengths from the finite element methods e.g.[SF73] for geometrical flexibility and the spectral methods e.g.[GO77] for high accuracy. The NSEM [MMP89], which allows nonconforming matching at element interfaces, provides increased geometrical flexibility. The flexibility in turn provides computational efficiency by allowing finer resolution in areas where physical variables are changing rapidly.

1 Civil, Mechanical and Environmental Engineering Department, School of Engineering and Applied Science, George Washington University, Washington, DC, 20052, USA . email: fhy@seas.gwu.edu

2 Address same as above.

Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund ©1999 DDM.org

The NSEM is capable of solving complicated PDEs such as the Navier-Stokes equations for fluid flow with high accuracy. However, high accuracy is costly, i.e. it requires long CPU times. Parallel computation is an effective way to decrease CPU time by using multiprocessors to do computations simultaneously. Due to the heterogeneous nature of the discretization, the SEM is naturally suited to concurrent implementation [FHK+88, FP91]: the high accuracy within each element provides sufficient loading of individual processors and the finite element approach decouples elemental work which can be assigned to independent processors.

This paper presents a parallel algorithm implementation of the NSEM. First, the implementation is based on a one element per processor scheme (OEPP). This work is based on a previous investigation of Ebrat et al [EMD97]. From execution time evaluations, we observe that the speedup and parallel efficiency are poor due to the processor load being low and the communication taking a big part of the total execution time. This paper therefore focuses on a scheme of multiple elements per processor (MEPP). Mappings of two and four elements per processor are tested for the algorithm, using message passing interface (MPI) on a SGI Power Challenge.

Spectral Element Method

SEMs are high-order weighted residual domain decomposition techniques based on spectral expansions for solution of PDEs such as the Navier-Stokes equations for direct simulation of fluid flow. The computational domain is broken up into macro-elements within which variables and geometry are represented by high-order tensor product polynomial expansions. The weak coupling between dependent variables for adjacent elements result in symmetric, and relatively sparse assembled matrices, which help to decrease the computational complexity in terms of memory requirements and processing time.

A simple test Poisson problem is presented to illustrate the spectral element discretization technique: $-\nabla^2 u = f$ in Ω, $u = 0$ on $\partial\Omega$ which has the following equivalent variational form: $a(u, v) = (f, v)$, $\forall v \in H_0^1(\Omega)$. The spectral element method proceeds by discretizing the domain Ω into $k = 1, \ldots, K$ subdomains Ω^k. The discretized equation is obtained from the Galerkin approximation with a finite set of functions. Legendre polynomials are introduced as the basis functions and Gauss-Lobatto quadrature is used to obtain the matrix equation

$$Au = Bf \tag{1}$$

where A is the discrete Laplacian operator and B is the mass matrix [MP89].

Mortar Method for Nonconforming Mesh

While the SEM, in comparison with spectral methods, is better suited to modeling complex geometries, the conforming geometric matching required between neighboring high-order elements leads to complications and inefficiencies in mesh generation, dynamic mesh refinement, and the treatment of moving boundaries. Allowing nonconforming matching at the element interfaces, the Nonforming Mortar Element Method [MMP89, BMP94] greatly increases the flexibility of the spectral element domain decomposition.

The method introduces a new mortar trace space which contributes to decoupling the local residual evaluation and the transmission of elemental boundary, i.e., continuity conditions. The configuration of the discretization of a nonconforming mesh is described in Figure 1(b) where each element is surrounded by a mortar structure γ with vertices ν at each corner. In the conforming case, the discretization space $X_h \subset H_0^1(\Omega)$; in nonconforming approximations, X_h is not a subspace of H_0^1, thereby introducing additional "consistency" errors. However, a carefully constructed discretization space retains spectral accuracy and with corresponding basis functions [MMP89], we obtain the fully discrete equations for the Poisson problem for the nonconforming mesh

$$Q^T A Q u = Q^T B f \tag{2}$$

where A and B represent the elemental conforming discrete Laplace operator and mass matrix respectively and Q is a projection operator from the mortar to the elemental structure (as shown in Figure 1(c)). The global Laplace operator is generated by local operators "mortared" together by Q and Q^T operations, where the Q^T operator is the algebraic form of the standard direct stiffness summation procedure.

The Nonconforming Spectral or Mortar Element method lends itself well to efficient implementation on parallel supercomputers: it allows for the classification and decoupling of the computational work into loosely coupled subdomains; it lays the foundation for sparse inter-domain communication and preserves local structure for fast evaluation procedures.

Parallel Implementation

The implementation is accomplished on a SGI Power Challenge with the intent of testing and future implementation on a larger more powerful parallel platform. Message passing between processors is achieved by MPI.

One Element per Processor Scheme (OEPP)

The simplest mapping of elements to processors is to assign each processor one element. Figure 2(b) shows a simple form of element-processor mapping for a model nonconforming mesh. Since each processor can only access the data of its own element, a neighborhood investigation operation is performed first to establish an array for each element recording the information about the neighborhood. After decomposition, a local mortar is established based on the global mortar configuration. Each processor retains only its corresponding portion of the global mortar. Figure 2 shows the relation between global mortar and local mortar for both the sequential and parallel implementation. All basic operators involved in spectral element discretization are

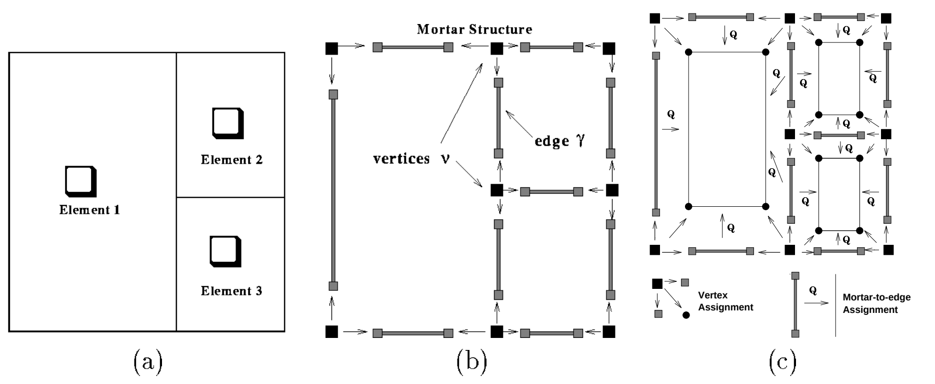

Figure 1 (a)A typical nonconforming mesh, (b) the structure of its corresponding mortar, (c) projection operation Q on nonconforming mesh.

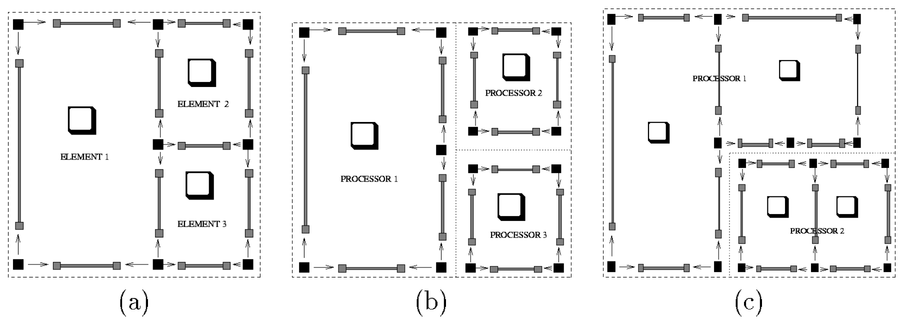

Figure 2 The structure of the mortar: (a) the global mortar for sequential implementation, (b) the local mortars for OEPP, (c) the local mortars for MEPP

formed in an initialization part of the program by all processors concurrently. The projection operators Q and Q^T are then created separately. For edges which have nonconforming matching, short communication with adjacent elements is necessary in order to obtain information about the size and coordinates of the nonconforming interfaces. Since the SEM generates a sparse linear positive definite system [MP89], the use of a conjugate gradient iterative solver is effective. After each iteration, neighboring elements exchange their edge information [EMD97]. Finally, a parallel I/O operation is performed by all processors concurrently to report the solution.

Multiple Elements per Processor Scheme (MEPP)

From the execution time analysis (in section 55), it is clear that OEPP has some weak points. Firstly, it is not practical since it requires the target machine to have the same number of processors as the number of elements. For a practical problem with a large number of elements, it is usually difficult to access that large number of processors. And if the number of processors is less than specified, then some processors will have

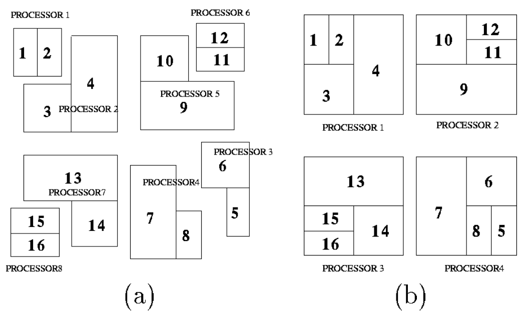

Figure 3 (a)Two and (b)four elements per processor mappings

twice or more load than others. Load-balancing is then upset and the operating system will spend all the time synchronizing MPI threads. Secondly, the OEPP scheme spends more time on communication than computation. To achieve good speedup and parallel efficiency, the computational load for each processor needs to be increased to get an optimal balance between computation and communication. The MEPP scheme is therefore developed as follows.

In the MEPP scheme, the same number of elements are assigned to each processor. Figure 3 shows the two and four elements per processor mappings for a 16 elements case.

As in the OEPP scheme, each element has an array recording neighborhood information. Using the information from element-processor mapping, it is known whether the adjacent elements are in the same processor or not. So in comparison with OEPP, MEPP has an additional array recording the information indicating whether the adjacent elements are located in the same processor or not. Thus, communication will only be performed between processors, not elements. After decomposition, a local mortar is established based on the global mortar configuration. Each processor retains only its corresponding portion of the global mortar. The mortar is shared by elements having adjacent interfaces in the same processor as shown in Figure 2(c). All processors do the initialization concurrently. Within each processor, every element does it sequentially. On each processor, the computation of Q and Q^T is performed sequentially for each element if needed. The short communication necessary to obtain information about size and coordinates of the nonconforming interface may be eliminated if the adjacent elements are on the same processor. The size of the linear system solved by conjugate gradient is proportional to NEP (number of elements per processor) times that of OEPP. In that case, the local computational load is increased NEP times (for a fixed number of C.G. iterations). After each iteration, only adjacent elements located in different processors need to exchange their mortar information. Otherwise, the mortar is updated locally, no message passing is needed. Thus communication cost is decreased.

Elements	Processors	Order	T_s(sec)	T_p(sec)	$\frac{T_c}{T_p}$	S	η
4	4	4	0.338	0.418	44.7%	0.809	20.2%
4	4	6	0.820	0.564	34.0%	1.454	36.3%
4	4	8	1.624	0.860	27.1%	1.888	47.2%
4	4	10	2.865	1.182	21.0%	2.424	60.6%
4	4	12	4.671	1.694	18.9%	2.757	68.9%
8	8	4	0.625	0.751	68.1%	0.832	10.4%
8	8	6	1.587	0.951	54.3%	1.669	20.9%
8	8	8	3.234	1.179	43.6%	2.743	34.3%
8	8	10	5.664	1.513	37.9%	3.744	46.8%
8	8	12	9.267	2.003	24.7%	4.627	57.8%

Table 1 Timing results for 250 conjugate gradient iterations for different problem sizes and order of approximation with the OEPP scheme.

Elements	Processors	Order	T_s(sec)	T_p(sec)	Speedup	Efficiency
16	8	4	1.254	0.843	1.488	18.6%
16	8	6	3.242	1.246	2.602	32.5%
16	8	8	6.528	1.900	3.436	42.9%
16	4	4	1.254	0.935	1.342	33.5%
16	4	6	3.242	1.585	2.045	51.1%
16	4	8	6.528	2.538	2.572	64.3%

Table 2 Timing results for 250 conjugate gradient iterations of 16 elements problem performed on 4 and 8 processors using MEPP scheme

Results and Evaluation

To test the correctness and effectiveness of the algorithm, error analysis and execution timings are performed on a test 2D Poisson problem with solution $u = sin(2\pi x)sin(2\pi y)$ on a square domain of side 4. Three cases of 4, 8, and 16 elements for the same computational domain decomposition are tested. Tables 1 and 2 show the speedup S and how parallel efficiency η varies with the number of elements and the order of approximation, where T_p and T_s refer to the parallel and sequential running time respectively and $\frac{T_c}{T_p}$ refers to the percentage of the communication time in the total running time. A fixed number of conjugate gradient iterations is used for time comparisons.

From these results, it can be observed that the speedup is not exactly proportional to the number of processors allocated to the problem. For a fixed problem, using more processors results in lower efficiency. Increasing the order of approximation for a fixed number of processors increases efficiency due to a decreasing percentage of the communication time in the overall running time.

Table 2 presents the speedup and parallel efficiency on 8 and 4 processors with two and four elements per processor respectively using the MEPP scheme. Obviously, the more processors used, the higher speedup should be attained. However, if a large part of the running time is spent on communication rather than computation, it will lead to lower speedup. Thus for the 4th order cases, the speedup does not change much even though twice as many as processors are used.

Conclusion and Future Research Direction

By the measured running time of the parallel algorithm, it can be observed that speedup increases with an increasing number of processors. However, the increase is not proportional to the number of processors. There is a limit beyond which speedup will not continue growing by adding more processors, because the communication time will dominate the execution time due to machine limitations.

Parallel efficiency decreases when more processors are used due to the increase of the interprocessor message exchange, but increases with increasing order of approximation. This means, to achieve good parallel efficiency, we need to load each processor with more computational work while minimizing the total interprocessor communication. Based on this observation, as well as the machine limitations (the number of processors available), the MEPP scheme is developed and comparatively high parallel efficiency is achieved. MEPP shows better performance than OEPP and will be the practical algorithm used in future research.

To increase computational load while decreasing communication work, an advanced data structure to deal with the connectivity information of the domain decomposition is required. This new structure should lead to an optimal element-processor mapping assigning adjacent elements to the same processor. To make the SEM more powerful and increase computational efficiency, dynamic mesh refinement and adaptive order of approximation assignment are needed. An adaptive sequential algorithm has been developed in [HM97]. In parallel, decomposition of the computational domain will be a dynamic problem which requires that the element-processor mapping also to be dynamic during execution. A parallel algorithm associated with dynamic mesh refinement and adaptive order of approximation will be the subject of future research.

REFERENCES

[BMP94] Bernardi C., Maday Y., and Patera A. T. (1994) A new nonconforming approach to domain decompositions: The mortar element method. *Pitman Research Notes in Mathematics* series 1, No. 299: 13.

[EMD97] Ebrat O., Mavriplis C., and Deane A. E. (1997) A parallel nonconforming spectral element algorithm on the Paragon. In *Proceedings of ISUG'97.*

[FHK$^+$88] Fischer P. F., Ho L., Karniadakis G. E., Rϕnquist E. M., and Patera A. T. (1988) Recent advances in parallel spectral element simulation of unsteady incompressible flows. In *Proceedings of the Symposium on Advances and Trends in Computational Structural Mechanics and Computational Fluid Dynamics.* Washington, D.C.

[FP91] Fischer P. F. and Patera A. T. (1991) Parallel spectral element solution of the Stokes problem. *Journal of Computational Physics* 92: 380–421.

[GO77] Gottlieb D. O. and Orszag S. (1977) *Numerical Analysis of Spectral Methods: Theory and Applications.* No.26, SIAM. NSF-CBMS Monograph, Philadelphia.

[HM97] Hsu L. and Mavriplis C. (1997) A two-dimensional adaptive spectral element method. In *13th AIAA Computational Fluid Dynamics Conference.*

[MMP89] Maday Y., Mavriplis C., and Patera A. T. (1989) Non-conforming mortar element methods: Application to spectral discretizations. In Chan T. F., Glowinski R., Periaux J., and Widlund O. B. (eds) *Domain Decomposition Methods*, pages 392–418. SIAM, Philadelphia.

[MP89] Maday Y. and Patera A. T. (1989) Spectral element method for the Navier-Stokes equations. In Noor A. K. (ed) *State-of-the-Art Surveys in Computational Mechanics*, pages 71–143. New York.

[Pat84] Patera A. T. (1984) A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computation Physics* 54(4): 468–488.

[SF73] Strang G. and Fix G. (1973) *An Analysis of The Finite Element Method.* Prentice-Hall.