English Series | NUMERICAL MATHEMATICS A Journal of Chinese Universities | Vol.14, No.1 Feb. 2005

A MPI PARALLEL PRECONDITIONED SPECTRAL ELEMENT METHOD FOR THE HELMHOLTZ EQUATION*

Hong Taoli(洪桃李) Xu Chuanju(许传炬)†

Abstract *Spectral element method is well known as high-order method, and has potential better parallel feature as compared with low order methods. In this paper, a parallel preconditioned conjugate gradient iterative method is proposed to solving the spectral element approximation of the Helmholtz equation. The parallel algorithm is shown to have good performance as compared to non parallel cases, especially when the stiffness matrix is not memorized. A series of numerical experiments in one dimensional case is carried out to demonstrate the efficiency of the proposed method.*

Key words *Spectral element method, parallel computing, finite element preconditioner.*

AMS(2000)subject classifications 65N35, 65N55

1 Introduction

The spectral element method (SEM) is essentially a discretization method for the approximate solution of partial differential equations expressed in a weak form, based on high-order Lagrangian interpolants used in conjunction with particular quadrature rules. It combines the geometric flexibility of finite element techniques with rapid convergence rate of spectral schemes [7]. Due to these advantages, the spectral element method is a viable alternative to currently popular methods such as finite volumes and finite elements, if accurate solutions of regular problems are sought. Another benefit of the SEM is that it is convenient to be paralleled in the implementation. In practical applications, moderate number of elements and high order polynomial degree are used, consequently most computation is performed within, rather than between,

* This work was supported by Natural Science Foundation of Fujian under Grant A0310002 and the Excellent Young Teachers Program (EYTP) of MOE of China.

† Corresponding author.

Received: Sep. 9, 2004.

elements. This feature makes the SEM great feasible to be implemented in parallel way.

Despite these nice features, the SEM has been thought to be more expensive in terms of the computational complexity and condition number of the stiffness matrix, as compared to the finite element methods with the same number of degree-of-freedom. There exists different approaches currently adopted in the SEM. The traditional approach in the spectral element community [7] has been to used a nodal tensorial expansion basis within structured subdomains (i.e., quadrilaterals or hexahedral elements). These bases are typically constructed from Lagrange polynomials through Gauss Lobatto Legendre quadrature points. In large-scale problems, long-range interactions between the basis elements within each substructure result in quite dense and expensive factorizations of the stiffness matrix, and the use of direct methods is hence not economical because of the large memory requirements [5]. In the past decade, iterative methods have been developed to solving the spectral element discretization problems of various equations. However, it has also been proven that the condition number of the SE stiffness matrix of the Laplacian operator is of order $O(KN^{d+1})$ [2], where K is the element number, N is the polynomial degree, d is the spatial dimension. Naive iterative methods are not really efficient due to the fact that too many iterations are required to reach the convergence. For a long time, many preconditioners have been proposed to overcome this difficulty. Orszag [12] and Deville et al. [3] proposed the use of a finite difference and a finite element model, respectively, as preconditioners for the spectral matrix. The triangulation for this finite element method was based on the hexahedrals of the Gauss-Lobatto-Legendre (GLL) mesh of one element. A theoretical justification of this preconditioning is provided by Parter and Rothman in [10,8]. Extension of these ideas to the SEM was proposed by Fischer [4], who used overlapping Schwarz methods applied to the GLL finite element model. The generalization of the result of [8] to the multi-element case was given by Huang and Xu in [6].

In this paper we follow the above mentioned ideas, but consider a different way to implement iterative algorithm for the preconditioned spectral element method. Precisely, we will consider a parallel preconditioned conjugate gradient method for the spectral element stiffness matrix using the linear finite element method as the preconditioner. Our main aim is to carry out a series of numerical experiences to show that use of the parallel algorithm can significantly reduce the CPU time of solving the spectral element system of the Helmholtz equation.

2 Formulation of the problem

We consider the following one dimensional Helmholtz equation: Find a function u defined in $\Omega = (a, b)$, such that

$$\begin{cases} -(pu')' + \lambda^2 u = f, \text{ in } \Omega, \\ u(a) = u(b) = 0, \end{cases} \tag{1}$$

where λ is a real number, prime (') denotes differentiation with respect to x, $f(x), p(x)$ are functions defined over Ω, which are assumed that there exists two positive constants τ_0 and τ_∞

such that

$$\tau_\infty > p(x) > \tau_0, \forall x \in \Omega. \tag{2}$$

The variational formulation of the problem (1) is given by: Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega p(x)u'(x)v'(x)\mathrm{d}x + \lambda^2 \int_\Omega u(x)v(x)\mathrm{d}x = \int_\Omega f(x)v(x)\mathrm{d}x, \forall v \in H_0^1(\Omega). \tag{3}$$

The spectral element discretization consists in choosing a pair of integers $h = (N, K)$ and divide the interval Ω into subintervals

$$\Omega = \bigcup_{k=1}^{K} \Omega_k,$$

where the k-th interval is assumed to have the length l_k. The $\Omega_k, k = 1, \cdots, K$ are generally called spectral elements. The spectral element space for the approximate solution, denoted by u_h, is the subspace X_h of $H_0^1(\Omega)$, defined as the set of all piecewise polynomials of degree less or equal than N:

$$X_h = H_0^1(\Omega) \cap P_{N,K}(\Omega),$$

where

$$P_{N,K}(\Omega) = \{\phi \in L^2(\Omega), \phi_{|\Omega_k} \in P_N(\Omega_k), k = 1, \cdots, K\}.$$

The SEM discrete problem is: Find $u_h \in X_h$ such that

$$\int_\Omega p(x)u_h'(x)v_h'(x)\mathrm{d}x + \lambda^2 \int_\Omega u_h(x)v_h(x)\mathrm{d}x = \int_\Omega f(x)v_h(x)\mathrm{d}x, \forall v_h \in X_h. \tag{4}$$

Let $\xi_0, \xi_1, \cdots, \xi_N$ stand for the Gauss-Labatto-Legendre (GLL) points, given by the zeros in $\Lambda = [-1, 1]$ of the Legendre polynomial of degree N:

$$\xi_0 = -1, \xi_N = 1, L_N'(\xi_i) = 0, \qquad \forall i \in \{1, \cdots, N-1\}.$$

The associated weights are denoted by $\rho_i, i = 0, \cdots, N$.

Let $\Omega_k = [a_{k-1}, a_k]$, then the global GLL points and associated weights are defined as follows:

$$\xi_{i,k} = a_{k-1} + (\xi_i + 1)l_k/2, \ \rho_{i,k} = \rho_i l_k/2, \qquad \forall i, 0 \le i \le N, \ \forall k, 1 \le k \le K.$$

Using the well-known Gauss-Lobatto quadrature to approximate the integrals in problem (4) gives the following SEM discretization problem: Find $u_h \in X_h$ such that

$$\begin{aligned} &\sum_{k=1}^{K}\sum_{i=0}^{N} p(\xi_{i,k})u_h'(\xi_{i,k})v_h'(\xi_{i,k})\rho_{i,k} + \lambda^2 \sum_{k=1}^{K}\sum_{i=0}^{N} u_h(\xi_{i,k})v_h(\xi_{i,k})\rho_{i,k} \\ &\quad = \sum_{k=1}^{K}\sum_{i=0}^{N} f(\xi_{i,k})v_h(\xi_{i,k})\rho_{i,k}, \qquad \forall v_h \in X_h. \end{aligned} \tag{5}$$

To arrive at a matrix statement, we must choose a basis for the discrete space X_h. There exists several possibilities on the basis choice. The most natural one is the Lagrangian interpolants based on the elemental GLL points [7].

A function $w_h \in X_h$ can be expressed as

$$w_h^k(r) = \sum_{i=0}^{N} w_i^k h_i(r),\ x \in \Omega_k \to r \in \Lambda \tag{6}$$

with

$$h_i \in P_N(\Lambda), h_i(\xi_j) = \delta_{ij}, \qquad \forall i,j \in \{0,\cdots,N\} \tag{7}$$

and $w_i^k = w_h(\xi_{i,k})$. The interfacial continuity and the boundary conditions of w_h is imposed by the requirements

$$w_N^k = w_0^{k+1},\ \forall k \in \{1,\cdots,K-1\}$$

and

$$w_0^1 = w_N^K = 0.$$

By expressing u_h in this way and choosing each test function v_h to be nonzero at only one global collocation point, we obtain the following algebraic equations given in matrix form:

$$\sum_{k=1}^{K}{}' \sum_{j=0}^{N} S_{ij}^k u_j^k = \sum_{k=1}^{K}{}' \sum_{j=0}^{N} B_{ij}^k f(\xi_{j,k}), \tag{8}$$

where

$$\begin{cases} S_{ij}^k = L_{ij}^k + \lambda^2 B_{ij}^k \cdots\cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\}, \\ L_{ij}^k = \dfrac{4}{l_k^2} \displaystyle\sum_{q=0}^{N} D_{qi} D_{qj} p(\xi_{q,k}) \rho_{q,k} \cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\}, \\ B_{ij}^k = \rho_{i,k}\delta_{ij} \cdots\cdots\cdots\cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\} \end{cases}$$

with $D = (D_{ij})_{(N+1)\times(N+1)}$ denoting the Legendre derivative matrix [1], Σ' denotes elemental direct stiffness summation, in which the continuity and boundary conditions imposed on u_h are taken into account.

3 FE Preconditioning

Since the condition number of the spectral element matrix is generally large, preconditioning will be necessary for an iterative method to be efficient. In this section, we recall the preconditioner based on the linear finite element method using the GLL mesh. This kind of preconditioner has been introduced by Deville [3], and then investigated by Parter [9] and Huang and Xu [6]. We refer to [6] for a detailed description of this preconditioner. Briefly, the finite element matrix, denoted by P, has a similar structure as the spectral element matrix S:

$$S = \begin{pmatrix} S^1 & & & \\ & \oplus & & \\ & & \ddots & \\ & & \oplus & S^K \end{pmatrix}, \qquad P = \begin{pmatrix} P^1 & & & \\ & \oplus & & \\ & & \ddots & \\ & & \oplus & P^K \end{pmatrix},$$

where the symbol $\oplus$ is defined: if B, C are two matrices as follow:

$$B = \begin{pmatrix} \hat{B} & \alpha \\ \alpha^T & b \end{pmatrix}, \qquad C = \begin{pmatrix} c & \beta^T \\ \beta & \hat{C} \end{pmatrix},$$

then

$$\begin{pmatrix} B & \\ & \oplus \\ & C \end{pmatrix} = \begin{pmatrix} \hat{B} & \alpha & \\ \alpha^T & b+c & \beta^T \\ & \beta & \hat{C} \end{pmatrix}.$$

The matrices S^1, S^K, P^1, P^K are of order $N \times N$, $S^k, P^k, 2 \le k \le K-1$, are of order $(N+1) \times (N+1)$. A difference between S^k and P^k is that S^k are full matrices while P^k are tridiagonal. Huang and Xu [6] have proven that using the preconditioner P to the matrix S results in a preconditioned system having the condition number independent of N and K.

4 Parallel preconditioned conjugate gradient method

4.1 MPI introduction

As a standard parallel environment, Message-Passing Interface (MPI) is employed in the design of our numerical algorithm. This is for the following reasons: first, MPI is a standard specification for message-passing libraries. Its portable implementation MPICH has been used for a wide variety of parallel and distributed computing environments. Second, MPICH is freely available. It supports a variety of operating systems including Windows, Linux, Unix, and a variety of programming language including FORTRAN, C, C++. Third, the program based on MPI can run on a computer having many CPU and can also run on a cluster of workstation network.

4.2 Parallel algorithm

Now we describe in detail our PCG parallel algorithm for the SEM Helmholtz system. As a powerful iterative method, PCG method has been one of the most widely applied iterative methods in the literature, especially when facing symmetric positive systems. We refer to [11] for a classical description of the PCG method. Generally speaking, in the PCG we need to solve the preconditioner once and calculate the product of matrix-vector once in each step of iteration. The formation of the spectral element stiffness matrix and the matrix-vector products cost most in the overall calculation.

Suppose we have $M(M > 1)$ processors in our architecture, we call the first processor as master processor and other $M-1$ processors as slave processor. A processor can be a CPU or a workstation in the cluster. We assign the tasks for each processor in the following way: first we divide the K elements into M groups: $K_1, \cdots, K_M$, such that

$$\sum_{n=1}^{M} K_m = K, \ K_m \ge 0, m = 1, \cdots, M.$$

Each group is assigned to a processor. The tasks K_m for the m-th processor are numbered from

k_m to $k_m + K_m - 1$. Hence

$$k_m + K_m = k_{m+1},\ 1 \le m \le M-1,$$
$$k_1 = 1,$$
$$k_M + K_M = K + 1.$$

Let $E_m := \{k_m, \cdots, k_m + K_m - 1\},\ 1 \le m \le M$, then all products of the matrix S^k for $k \in E_m$ and a vector are calculated by the m-th processor. If all processors have same performance, then reasonable workload of each processor should be balanced. In this case we would require the following balance condition:

$$|K_i - K_j| \le 1, 1 \le, i, j \le M,$$

which can be realized by choosing

$$K_m = \left[\frac{K}{M}\right], \quad \text{if} \quad 1 \le m \le M - (K \bmod M),$$

$$K_m = \left[\frac{K}{M}\right] + 1, \quad \text{if} \quad M - (K \bmod M) < m \le M.$$

An example of natural choice for K_m is given in Tab.1, where we have 17 elements ($K = 17$), 5 processors ($M = 5$). In this case each processor can be distributed 3 or 4 elements.

Table 1 Distribution of the tasks for $K = 17$ and $M = 5$.

m	E_m
1	1,2,3
2	4,5,6
3	7,8,9
4	10,11,12,13
5	14,15,16,17

The m-th processor calculates $S^k u^k$ for each iterative solution u^k for all $k \in E_m$ for $m = 1, \cdots, M$. Once all $S^k u^k$ are obtained, the master processor assembles the products coming from the slave processors to carry out the elemental stiffness summation. Other related work such as the global norm calculation, determination of convergence and so on, are also performed in the master processor.

In summary, the overall algorithm for the preconditioned SEM of the Helmholtz equation can be described as follows:

(1) The master processor computes the finite element preconditioner.

(2) Each processor computes corresponding $S^k u^k, \forall k \in E_m$.

(3) Each slave processor sends the elemental component of their own to the master processor. The master processor assembles these components to form the full vector, and then carry out the stiffness summation.

(4) The master processor solves the global preconditioner, and other necessary works required by the PCG method.

(5) The master processor computes the residual to determine whether the solution is obtained with required accuracy. If non, it sends to each slave processor the corresponding components of the full vector, go to (2).

The above process can also be represented in a schematic way, see Fig.1.

Figure 1 Schematic representation of the PCG algorithm.

Remark 1 Note that the data exchange between different processors is not significant as compared to the evaluation of the matrix-vector products within each processor. Actually the preconditioner is solved by a direct method by the master processor, the data exchange required by our algorithm is optimal (this means that the communication is minimal). However, in the case of two or three dimensional problems, the preconditioner have to be solved iteratively due to the memory limit, the data exchange strategy described above can be improved by only sending the interfacial data of the vector to the master processor. Doing so could furthermore reduce the communications between the processors.

Remark 2 In the step (2), the spectral element matrix was constructed explicitly in order to reduce the CPU time for the matrix-vector products. However in the 2- and 3-dimensional cases, the storage of the spectral element matrix needs much more memory ($O(KN^4)$ in the 2D, $O(KN^6)$ in the 3D), and will become quickly unrealistic when K or N gets large. As a result, implicity evaluation of the matrix-vector products would be unavoidable. This change would save memory, but cost more time.

5 Numerical experiments

In this section, we report the results of the numerical experiments. All the tests have been

run on the SGI Origin3800 server in the State Key Laboratory of Scientific and Engineering Computing (LSEC), Institute of Computational Mathematics & Scientific & Engineering Computation, Chinese Academy of Sciences. The numerical configuration is chosen as follow:

$$\Omega = (0, 2\pi),$$
$$p(x) = \delta,$$
$$\lambda = 1,$$
$$f(x) = (\delta k^2 + 1)\sin(kx).$$

With this configuration, the exact solution of the problem (1) is known:

$$u(x) = \sin(kx).$$

In all tests, we choose $\delta = 0.001, k = 100$, and the unit of time is 10 milliseconds. The domain is broken into equi-elements.

Test 1 Study of the convergence rate of the PCG, accuracy in function of K, as well as the CPU time report in different steps.

Parameters: $N = 3, M = 1$, matrix memorized with preconditioning.

2 Convergence rate of the PCG, solution errors, and CPU time for $N = 3, M = 1$.

K	1000	2000	4000	8000	16000
Iteration numbers	5	6	6	5	5
Max error	4.78e-06	1.51e-07	4.75e-09	1.48e-10	5.72e-12
Total time	16	33	67	128	258
Create Preconditioner	11	23	45	90	180
Create Matrix S	1	2	4	8	16
PCG operation	3	7	14	23	48

From the table 1, we see that when N is fixed, CPU time increases linearly as K increases, as being evident from the description of our algorithm. Also seen is the errors of the SEM solution as a function of the element number. As expected, the errors behaves like $O(h^{N+1})$ with h being the element size. Independence of the iteration number of the PCG method on K shows the efficiency of the FE preconditioner.

Test 2 Similar to test 1, but with K fixed.

Parameters: $K = 500, M = 1$, matrix memorized with preconditioning.

Table 3 Convergence rate of the PCG, solution errors, and CPU time for $K = 500, M = 1$.

N	3	5	7	9	11
Iteration numbers	3	5	7	9	11
Max error	1.55e-4	1.85e-7	1.67e-10	1.78e-13	1.03e-13
Create S	1	2	3	5	9
PCG operation	2	3	5	9	14

In Table 3 we repeat the comparisons as in Table 2, but now with K fixed. Surprisely, when N increases within a small range, the iteration number increases linearly with N. However when N is large, the iteration number seems to become stable, as shown in Figure 2. The reason for this behavior is not yet clear for us. From Table 3, the spectral convergence with increasing N is clearly observed. This is in good agreement with the general theoretical results about the SEM. The CPU time in different steps conforms once again with the description of the algorithm.

Figure 2 Iteration numbers as a function of the polynomial degree N.

Test 3 parallel computing: comparison of the CPU time when processor number increases.

Parameters: $K = 500, N = 11$, matrix memorized with preconditioning.

Table 4 CPU time for different processor numbers used.

M	1	2	4	8
Create matrix S	8	4	2	1
PCG operation	14	10	8	7

In Table 4, we list the CPU time when different processor numbers are used. When the processor number M is increased, the time to form the spectral element matrix S decreases linearly, but the time to solve the preconditioned system decreases more slowly, this is due to the fact that solving the preconditioner is not paralleled and the time for communication increases as M increases. Another reason is that the spectral element matrix S is formed in the pre-procedure, the evaluation of the matrix-vector products is rapid as compared with the preconditioning and

the communication.

Test 4 Same as in Test 3 except that the SE matrix S is not memorized.

Parameters: $K = 500, N = 11$, matrix not memorized with preconditioning.

Table 5 Same as in Table 4, but matrix not memorized.

M	1	2	4	8
Total time	89	51	25	16

We repeat the test 3 but now the matrix S is not memorized. In this case, the evaluation of the matrix-vector products is much more expensive, hence the parallelization can be expected to be more efficient. In Table 5, the total CPU time is listed when different numbers of processor are used in the computation. We observe that when the processor number M is increased, the total CPU time is reduced quickly (almost decreases linearly with M). This is one of the most interesting cases of using parallel algorithm.

Conclusions

We have introduced and tested a parallel iterative algorithm to solve the problem arising from the spectral elements method of the Helmholtz problem in one dimensional case. The numerical tests have shown the interests by using the finite element preconditioner and parallel coding. This parallel algorithm could be extended to the 2- and 3-dimensional cases. This is our ongoing work.

References

1 Canuto C, Hussaini M Y, Quateroni A, Zang T A. Spectral Methods in Fluid Dynamics. Springer-Verlag, New York. 1988

2 Bernardi C, Maday Y. Approximations spectrales de problèmes aux limites elliptiques. Springer-Verlag, Paris, 1992

3 Deville M O, Mund E H. Finite-element preconditioning for pseudospectral solutions of elliptic problems. SIAM J. Sci. Statist. Comput., 1990, 11: 311-342

4 Fischer P F. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. J. Comput. Phys., 1997, 133: 84-101

5 Fischer P F, Ronquist E. Spectral element methods for large scale parallel Navier-Stokes calculations. Comput. Methods Appl. Mech. Engrg, 1994, 116: 69-76

6 Huang W B, Xu C J. Analysis for a finite element preconditioned spectral element method of the Poisson Equation. J. Xiamen Univ., 2003, 42(4): 421-424

7 Maday Y, Patera A T. Spectral element methods for the incompressible Navier-Stokes equations. in State of the Art Surveys in Computational Mechanics, A. K. Noor (ed.), 1989, 71-143

8 Parter S V. Preconditioning Legendre spectral collocation approximations to elliptic problems II: finite element operators. SIAM J. Numer. Anal, 2001, 39(1): 348-362

9 Parter S V, Rothman E E. Preconditioning Legendre spectral collocation approximations to elliptic equations and systems. SIAM J. Numer. Anal, 1992, 29(4): 917-936

10 Parter S V, Rothman E E. Preconditioning Legendre spectral collocation approximations to elliptic problems. SIAM J. Numer. Anal., 1995, 32: 333-385

11 Quarteroni A, Valli A. Numerical Approximation of Partial Differential Equations. Springer-Verlag, New York, 1997

12 Orszag S A. Spectral methods for problems in complex geometries. J. Comput. Phys., 1980, 37: 70-92

Hong Taoli Department of Computational Mathematics, School of Mathematical Sciences, Xiamen University, Xiamen 361005, PRC.

Xu Chuanju Corresponding author, Department of Computational Mathematics, School of Mathematical Sciences, Xiamen University, Xiamen 361005, PRC.