Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 225 (2007) 1721–1735
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An efficient direct parallel spectral-element solver for separable elliptic problems

Yuen-Yick Kwan, Jie Shen *

Department of Mathematics, Purdue University West Lafayette, IN 47907, USA

Received 30 October 2006; received in revised form 29 January 2007; accepted 13 February 2007
Available online 24 February 2007

Abstract

An efficient direct parallel elliptic solver based on the spectral element discretization is developed. The direct solver is based on a matrix decomposition approach which reduces multi-dimensional separable problems to a sequence of one-dimensional problems that can be efficiently handled by a static condensation process. Thanks to the spectral accuracy and the localized nature of a spectral element discretization, this elliptic solver is spectrally accurate and can be efficiently parallelized, and it can serve as an essential building block for large scale high-performance solvers in computational fluid dynamics and computational materials science.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Spectral-element; Domain decomposition; Parallel computing; Elliptic solver

1. Introduction

In recent years, spectral and spectral-element methods have enjoyed an increased popularity among computational scientists who value accurate and reliable numerical approximations. In particular, the spectral-element method (cf. [15]), which combines the advantages of the high accuracy of the spectral method and the geometric flexibility of the finite-element method, has been used successfully in many large scale computations for real life applications (see, for instance, [8,12] and the references therein).

For obvious reasons, well-developed spectral-element codes such as NEK5000 (cf. [23]) and NEKTAR (cf. [11]) do not really take advantage of the situations where the domains are simple separable domains, and use in general a suitable iterative procedure to solve the underlying linear systems. However, there are many emerging applications which are set on simple domains but require integrating PDEs with very large numbers of degree of freedoms for a very long time interval. In these situations, it is desirable to have an accurate, fast and direct parallel solver. The goal of this paper is to develop such an elliptic solver based on a spectral-element discretization. We recall that such elliptic solvers based on a (one-element) spectral discretization

* Corresponding author.
E-mail addresses: ykwan@math.purdue.edu (Y.-Y. Kwan), shen@math.purdue.edu (J. Shen).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.02.013

are well developed (cf., for instance, [3,10,18,19,25]), but they are not particularly tuned for parallel computations due to the global nature of the (one-element) spectral method. On the other hand, many applications have some localized features, such as layered materials, thin interfaces, that can be better treated by a spectral-element discretization.

In practice, a spectral-element discretization is often combined with a domain decomposition approach (cf. [7]). In particular, the technique of static condensation (also known as sub-structuring) is often used to decouple the interior unknowns from the interface ones, making direct solvers affordable in one- and two-space dimensions (see, for instance, [5,15]). Nevertheless, for three-dimensional problems, the size of the interface unknowns become prohibitively large to allow a direct method so an iterative technique become necessary (see, e.g., [17,22]). Although optimal preconditioners are available for finite element discretizations, only sub-optimal (with respect to the polynomial degree used in each element) preconditioners are available for a spectral-element discretization (cf. [1,16,22]), making the spectral element solvers (particularly when high-order polynomials are used) considerably more expensive than finite element solvers in three-space dimension. This is one of the reasons why usually relatively low degrees of polynomials ($\sim$12) are used in a spectral-element discretization.

We develop in this paper a direct, parallel spectral-element solver for separable elliptic problems. The solver is based on a matrix decomposition approach [6,13,25] which reduces multi-dimensional problems to a sequence of one-dimensional problems that can be efficiently handled by a static condensation process. The advantages of this solver include: (i) it is a direct solver for both two and three dimensions with a competitive computational complexity; (ii) it can be efficiently parallelized thanks to the localized nature of the spectral-element discretization; and (iii) it allows for larger time step for time-dependent problems since the collocation points are not as close together as in the global spectral methods.

The paper is organized as follows. In Section 2, we describe the algorithm and its extension to multi-dimensional problems. The parallel implementation is discussed in Section 3. In Section 4, we illustrate the efficiency of the algorithm with several numerical examples. Finally, some concluding remarks are given in Section 5.

2. Description of the algorithm

To simplify the notation, we consider the Poisson-type equation with homogeneous Dirichlet boundary conditions

$$\alpha u - \Delta u = f, \ \text{ in } \Omega; \quad u|_{\partial\Omega} = 0, \tag{2.1}$$

where α is a non-negative constant and Ω is a two- or three-dimensional separable domain. For the purpose of illustration, we shall describe in some detail the solution procedures for two particular cases: (i) Ω being a 3-D rectangular box, i.e.

$$\Omega = (x_-, x_+) \times (y_-, y_+) \times (z_-, z_+); \tag{2.2}$$

and (ii) Ω being the region between two concentric cylinders, i.e.

$$\Omega = \{(r, \theta, z) : r_- < r < r_+, \ 0 \leqslant \theta < 2\pi, \ z_- < z < z_+\}. \tag{2.3}$$

Extensions to other separable elliptic equations with more general and/or non-homogeneous boundary conditions are quite straightforward and will be discussed along the way. As the finite element methods, the spectral-element method for (2.1) is based on a variational formulation. Let us denote $V = H_0^1(\Omega)$. Then, the variational formulation for (2.1) is: Find $u \in V$ such that

$$\mathscr{A}(u, v) = \mathscr{F}(v) \quad \forall v \in V, \tag{2.4}$$

where

$$\begin{aligned} &\mathscr{A}(u, v) := \alpha(u, v) + (\nabla u, \nabla v), \\ &\mathscr{F}(v) := (f, v), \end{aligned} \tag{2.5}$$

and $(\cdot, \cdot)$ represents the inner-product in $L^2(\Omega)$.

Our basic strategy is to use the matrix diagonalization method (cf. [13]), which is a discrete version of *separation of variables*, to reduce the multi-dimensional problem to a sequence of one-dimensional problems. Hence, we shall start with the one-dimensional case.

2.1. One-dimensional case

Let $\Omega = (x_-, x_+)$ and Δ_K be a partition of (x_-, x_+) with K elements:

$$\Delta_K : x_- = x_0 < x_1 < \cdots < x_{K-1} < x_K = x_+.$$

and set $\Omega_k = (x_{k-1}, x_k),\ k = 1, \ldots, K$. We now describe the spectral-element method (see, for instance, Chapter 22 in [5] for a similar description). We define the spectral-element space

$$V_N^K := \{u \in V : u|_{\Omega_k} \in \mathbb{P}_N,\ k = 1, \ldots, K\},$$

where $\mathbb{P}_N$ is the space of polynomials of degree less than or equal to N. Then the spectral-element method for (2.4) in the one-dimensional case is to find $u_N^K \in V_N^K$ such that

$$\mathscr{A}(u_N^K, v) = \mathscr{F}(v) \quad \forall v \in V_N^K. \tag{2.6}$$

Let $\{\phi_i^k,\ i = 1, \ldots, N-1\}$ be a set of basis functions (to be specified later) for $V^k = \mathbb{P}_N \cap H_0^1(\Omega_k)$, and $\{\psi_k,\ k = 1, \ldots, K-1\}$ be the piecewise linear hat functions based on the grid $\{x_k, k = 0, \ldots, K\}$. Then

$$V_N^K = \operatorname{span}\{\phi_i^k,\ k = 1, \ldots, K;\ i = 1, \ldots, N-1; \quad \psi_k,\ k = 1, \ldots, K-1\}. \tag{2.7}$$

Let us denote by A^k, B^k $(k = 1, \ldots, K)$ and C the matrices with the elements

$$\begin{aligned}
A_{i,j}^k &= \mathscr{A}(\phi_j^k, \phi_i^k), \quad i, j = 1, \ldots, N-1,\\
B_{i,j}^k &= \mathscr{A}(\psi_j, \phi_i^k), \quad i = 1, \ldots, N-1,\ \ j = 1, \ldots, K-1,\\
C_{i,j} &= \mathscr{A}(\psi_j, \psi_i), \quad i, j = 1, \ldots, K-1,
\end{aligned}$$

and set

$$A = \operatorname{diag}(A^1, \ldots, A^k), \quad B = (B^1, \ldots, B^k)^{\mathrm{T}}.$$

Writing u_N^K as an expansion in the basis functions, i.e.

$$u_N^K = \sum_{k=1}^{K} \sum_{i=1}^{N-1} u_{k,i}^1 \phi_i^k + \sum_{k=1}^{K-1} u_k^2 \psi_k$$

and denoting

$$\begin{aligned}
\underline{u}^1 &= (u_{1,1}^1, \ldots, u_{1,N-1}^1, \ldots, u_{K,1}^1, \ldots, u_{K,N-1}^1)^{\mathrm{T}},\\
\underline{u}^2 &= (u_1^2, \ldots, u_{K-1}^2)^{\mathrm{T}},\\
\underline{f}^1 &= (f_{1,1}^1, \ldots, f_{1,N-1}^1, \ldots, f_{K,1}^1, \ldots, f_{K,N-1}^1)^{\mathrm{T}} \quad \text{with } f_{k,i}^1 = \mathscr{F}(\phi_i^k),\\
\underline{f}^2 &= (f_1^2, \ldots, f_{K-1}^2)^{\mathrm{T}} \quad \text{with } f_k^2 = \mathscr{F}(\psi_k),
\end{aligned}$$

where the inner-products $\mathscr{F}(\phi_i^k)$ and $\mathscr{F}(\psi_k)$ are computed with Gauss–Lobatto rule or other numerical integration. Then (2.6) is reduced to the linear system

$$\begin{bmatrix} A & B \\ B^{\mathrm{T}} & C \end{bmatrix} \begin{bmatrix} \underline{u}^1 \\ \underline{u}^2 \end{bmatrix} = \begin{bmatrix} \underline{f}^1 \\ \underline{f}^2 \end{bmatrix}. \tag{2.8}$$

2.1.1. Static condensation

To solve the linear system (2.8), we first compute the Cholesky factorization of A. As we shall demonstrate below, we can choose suitable basis functions such that $\{A^k,\ k = 1, \ldots, K\}$ are banded symmetric matrices, so

the Cholesky factorization of A can be computed in $\mathcal{O}(NK)$ operations. With a block Gaussian elimination, (2.8) can be transformed to

$$\begin{bmatrix} I & \widetilde{B} \\ 0 & \widetilde{C} \end{bmatrix} \begin{bmatrix} \underline{u}^1 \\ \underline{u}^2 \end{bmatrix} = \begin{bmatrix} \underline{g}^1 \\ \underline{g}^2 \end{bmatrix}, \tag{2.9}$$

where $\widetilde{B} = A^{-1}B$, $\widetilde{C} = C - B^{\mathrm{T}}A^{-1}B$ is the Schur complement, and

$$\underline{g}^1 = A^{-1}\underline{f}^1, \quad \underline{g}^2 = \underline{f}^2 - B^{\mathrm{T}}\underline{g}^1.$$

Note that $B^k_{i,j}$ is nonzero only when $j = k-1$ and $j = k$. Hence $\widetilde{B}$ can be computed in $\mathcal{O}(NK)$ operations. Note that both C and $\widetilde{C}$ are symmetric tridiagonal so $\underline{u}^2$ can be solved in $\mathcal{O}(K)$ operations, and then, $\underline{u}^1$ can be computed as

$$\underline{u}^1 = \underline{g}^1 - \widetilde{B}\underline{u}^2.$$

We summarize below the steps and operation counts for solving (2.6), assuming that the input is the vectors $\underline{f}^1$, $\underline{f}^2$, and the output is the vectors $\underline{u}^1$, $\underline{u}^2$.

Step	Operation count
Preprocessing	
Compute A	$\mathcal{O}(NK)$
Cholesky factorization of A	$\mathcal{O}(NK)$
Compute B	$\mathcal{O}(NK)$
Compute $\widetilde{B} = A^{-1}B$	$\mathcal{O}(NK)$
Compute C	$\mathcal{O}(K)$
Compute $\widetilde{C} = C - B^{\mathrm{T}}\widetilde{B}$	$\mathcal{O}(NK)$
Solving	
Compute $\underline{g}^1 = A^{-1}\underline{f}^1$	$\mathcal{O}(NK)$
Compute $\underline{f}^2$	$\mathcal{O}(K)$
Compute $\underline{g}^2 = \underline{f}^2 - B^{\mathrm{T}}\underline{g}^1$	$\mathcal{O}(K)$
Compute $\underline{u}^2 = \widetilde{C}^{-1}\underline{g}^2$	$\mathcal{O}(K)$
Compute $\underline{u}^1 = \underline{g}^1 - \widetilde{B}\underline{u}^2$	$\mathcal{O}(NK)$

2.1.2. Basis functions

We now introduce the basis functions $\{\phi_i^k,\ k = 1,\ldots,K;\ i = 1,\ldots,N-1\}$ that would result in banded matrices A^k $(k = 1,\ldots,K)$.

Let us consider first the 1D equation reduced from (2.1):

$$\alpha u - u_{xx} = f, \ \text{ in } \ (x_-, x_+); \quad u(x_-) = u(x_+) = 0. \tag{2.10}$$

Introducing the local coordinate

$$\tilde{x}_k = 2 \cdot \frac{x - x_{k-1}}{x_k - x_{k-1}} - 1, \quad x \in \Omega_k, \tag{2.11}$$

which transforms the physical domain Ω_k to the reference domain $(-1,1)$. Following [18] we define

$$\phi_i^k(x) = \begin{cases} L_{i-1}(\tilde{x}_k) - L_{i+1}(\tilde{x}_k), & x \in \Omega_k, \\ 0, & \text{otherwise}, \end{cases} \qquad i = 1,\ldots,N-1, \tag{2.12}$$

where L_i is the Legendre polynomial of degree i. Then, the elements of A^k are

$$A^k_{i,j} = \frac{\alpha h_k}{2} \int_{-1}^{1} \phi_j^k(x)\phi_i^k(x)\,\mathrm{d}x + \frac{2}{h_k} \int_{-1}^{1} \partial_x\phi_j^k(x)\partial_x\phi_i^k(x)\,\mathrm{d}x, \quad i,j = 1,\ldots,N-1,$$

where $h_k = x_k - x_{k-1}$. Thanks to the orthogonality of the Legendre polynomials, one finds that A^k is symmetric with three nonzero diagonals [18].

Next we consider the following 1D equation which, as we shall we later, appears after applying the matrix diagonalization method to the problem (2.1) in the domain (2.3):

$$\alpha u - \frac{1}{r}(ru_r)_r + \frac{\gamma}{r^2}u = f, \ \text{ in } \ \Omega = (r_-, r_+); \quad u(r_-) = u(r_+) = 0. \tag{2.13}$$

Here γ is a non-negative integer. Treating the variable r in the same way as we treat x, the spectral-element approximation to (2.13) is still (2.6) with the bilinear form $\mathscr{A}(u,v)$ and $\mathscr{F}(v)$

$$\mathscr{A}(u,v) = \alpha(u,v)_r + (u_r, v_r)_r + \gamma\left(\frac{u}{r}, \frac{v}{r}\right)_r, \quad \mathscr{F}(v) = (f,v)_r,$$

where the inner product is: $(u,v)_r = \int_{r_-}^{r_+} uvr\,\mathrm{d}r$. Note that the basis functions (2.12) will lead to full matrices A^k due to the term $\left(\frac{u}{r}, \frac{v}{r}\right)_r$. Hence, we shall use

$$\begin{aligned} \phi_1^k(r) &= \begin{cases} L_0(\tilde{r}_k) - L_2(\tilde{r}_k), & r \in \Omega_k = (r_{k-1}, r_k), \\ 0, & \text{otherwise}, \end{cases} \\ \phi_i^k(r) &= \begin{cases} (\tilde{r}_k + c_k)(L_{i-2}(\tilde{r}_k) - L_i(\tilde{r}_k)), & r \in \Omega_k, \\ 0, & \text{otherwise}, \end{cases} \quad i = 2, \ldots, N-1, \end{aligned} \tag{2.14}$$

where $\tilde{r}_k = 2 \cdot \frac{r - r_{k-1}}{r_k - r_{k-1}} - 1$, $r \in \Omega_k$, and $c_k = \frac{r_{k-1} + r_k}{2}$. Therefore, the elements of A^k are

$$A_{i,j}^k = \frac{\alpha h_k}{2}\int_{-1}^{1} \phi_j^k(r)\phi_i^k(r)(r + c_k)\,\mathrm{d}r + \frac{2}{h_k}\int_{-1}^{1} \partial_r\phi_j^k(r)\partial_r\phi_i^k(r)(r + c_k)\,\mathrm{d}r + \frac{2\gamma}{h_k}\int_{-1}^{1} \frac{1}{r + c_k}\phi_j^k(r)\phi_i^k(r)\,\mathrm{d}x,$$
$$i, j = 1, \ldots, N-1,$$

where $h_k = r_k - r_{k-1}$. Then, it can be easily verified that A^k is symmetric with 11 nonzero diagonals.

2.1.3. Some immediate extensions

Equations with variable coefficients. It is clear that we can replace Eq. (2.10) by the more general equation:

$$\alpha(x)u - (\beta(x)u_x)_x = f, \ \text{ in } \ (x_-, x_+); \quad u(x_-) = u(x_+) = 0, \tag{2.15}$$

where $\beta(x) > 0$ and $\alpha(x) \geqslant 0$ could be piecewise continuous. The only difference is that in case $\alpha(x)$ and $\beta(x)$ are not constants or piecewise constants, the matrices A^k should be computed with a discrete inner product based on the local Gauss–Lobatto quadrature and A^k will be in general full.

By the same token, Eq. (2.13) can also be replaced by the more general equation:

$$\alpha(r)u - \frac{1}{r}(\beta(r)u_r)_r + \frac{\gamma}{r^2}u = f, \ \text{ in } \ \Omega = (r_-, r_+); \quad u(r_-) = u(r_+) = 0. \tag{2.16}$$

More general boundary conditions. First of all, non-homogeneous boundary conditions can be easily treated by subtracting a suitable simple function satisfying the non-homogeneous boundary condition from the solution, so we shall only deal with homogeneous boundary conditions in this paper.

Consider for instance the following general boundary conditions for (2.10) or (2.13):

$$u_x(x_-) + a_-u(x_-) = 0, \quad u_x(x_+) + a_+u(x_+) = 0. \tag{2.17}$$

Although in general such boundary conditions are usually handled weakly, as showed in [20], it is, however, beneficial and possible to treat them strongly in a spectral approximation to achieve sparsity for the stiffness and mass matrices. It is clear that one only needs to change the basis functions supported in the two elements which include a boundary point. For example, the basis function $\phi_i^1(x)$ with support in (x_-, x_1) should satisfy

$$\partial_x\phi_i^1(x_-) + a_-\phi_i^1(x_-) = 0, \quad \phi_i^1(x_1) = 0, i = 1, 2, \cdots, N-1. \tag{2.18}$$

It is shown in [20] that one can uniquely determine a_i and b_i such that $\phi_i^1(x) = L_{i-1}(\tilde{x}_1) + a_iL_i(\tilde{x}_1) + b_iL_{i+1}(\tilde{x}_1)$ (where $\tilde{x}_1$ is the local variable defined in (2.11)) satisfies (2.18).

2.2. Multi-dimensional case

For the sake of simplicity, we shall start with solving (2.1) in the rectangle $\Omega = \Omega_x \times \Omega_y = (x_-, x_+) \times (y_-, y_+)$. The three-dimensional cases (2.2) and (2.3) will be treated afterwards. Eq. (2.1) now reads

$$\begin{aligned} &\alpha u - u_{xx} - u_{yy} = f, \ \text{ in } \Omega; \\ &u|_{\partial\Omega} = 0. \end{aligned} \tag{2.19}$$

We now describe the spectral-element approximation for (2.19). Given a rectangular grid on $\overline{\Omega}$ with nodes $\{(x_i, y_j) : 0 \leqslant i \leqslant K_x; 0 \leqslant j \leqslant K_y\}$, we set

$$V_{\mathbb{M}} = \{v \in H_0^1(\Omega) : v|_{(x_{i-1},x_i)\times(y_{j-1},y_j)} \in \mathbb{P}_{N_x} \times \mathbb{P}_{N_y}, 1 \leqslant i \leqslant K_x, 1 \leqslant j \leqslant K_y\}, \tag{2.20}$$

where $\mathbb{M} = (M_x, M_y) = (N_x K_x - 1, N_y K_y - 1)$. Then, the spectral-element approximation to (2.19) is: find $u_{\mathbb{M}} \in V_{\mathbb{M}}$ such that

$$\mathscr{A}(u_{\mathbb{M}}, v_{\mathbb{M}}) = \mathscr{F}(v_{\mathbb{M}}) \quad \forall v_{\mathbb{M}} \in V_{\mathbb{M}}. \tag{2.21}$$

Let $\{\phi_i^x\}_{i=1}^{M_x}$ and $\{\phi_i^y\}_{i=1}^{M_y}$ be, respectively, the set of one-dimensional basis functions in the x- and y-direction (described previously for (2.10)). Let us define A^x and B^x, matrices of size M^x, and A^y and B^y, matrices of size M_y, with elements

$$\begin{aligned} A_{ij}^x &= (\phi_j^x, \phi_i^x), \quad B_{ij}^x = (\partial_x \phi_j^x, \partial_x \phi_i^x), \\ A_{ij}^y &= (\phi_j^y, \phi_i^y), \quad B_{ij}^y = (\partial_y \phi_j^y, \partial_y \phi_i^y). \end{aligned} \tag{2.22}$$

Then, writing

$$u_{\mathbb{M}} = \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} u_{ij} \phi_i^x(x) \phi_j^y(y),$$

Eq. (2.21) reduces to the linear system

$$\mathscr{L}\underline{u} := ((\alpha A^x + B^x) \otimes (A^y + A^x \otimes B^y))\underline{u} = \underline{f}, \tag{2.23}$$

where $\underline{u}$ and $\underline{f}$ are vectors with elements $(\underline{u})_{(j-1)M_y+i} = u_{ij}$ and $(\underline{f})_{(j-1)M_y+i} = (f, \phi_i^x(x)\phi_j^y(y))_\Omega$.

2.2.1. Matrix diagonalization method

We shall use the matrix diagonalization method (cf. [13]) to solve the linear system (2.23). Recall that A and B are symmetric matrices so there exists a diagonal matrix Λ and an orthonormal matrix E which solve the generalized eigenvalue problem

$$B^y E = A^y E \Lambda. \tag{2.24}$$

Then,

$$\left(I \otimes E^{\mathrm{T}}\right) \times \mathscr{L} \times (I \otimes E) = (\alpha A^x + B^x) \otimes I + A^x \otimes \Lambda.$$

It follows that

$$\begin{aligned} \mathscr{L}^{-1} &= \left(\left(I \otimes E^{-\mathrm{T}}\right) \times ((\alpha A^x + B^x) \otimes I + A^x \otimes \Lambda) \times (I \otimes E^{-1})\right)^{-1} \\ &= (I \otimes E) \times ((\alpha A^x + B^x) \otimes I + A^x \otimes \Lambda)^{-1} \times \left(I \otimes E^{\mathrm{T}}\right). \end{aligned}$$

Let

$$\underline{v} = (I \otimes E^{-1}) \times \underline{u}, \quad \underline{g} = \left(I \otimes E^{\mathrm{T}}\right) \times \underline{f},$$

Then the linear system (2.23) becomes

$$((\alpha A^x + B^x) \otimes I + A^x \otimes \Lambda)\underline{v} = \underline{g}. \tag{2.25}$$

Let V and G be $M_x \times M_y$ matrices formed by reshaping the vectors $\underline{v}$ and $\underline{g}$. Then the linear system (2.25) is equivalent to the linear systems

$$((\alpha + \lambda_j)A^x + B^x)\underline{v}_j = \underline{g}_j, \quad j = 1, \ldots, M_y, \tag{2.26}$$

where $\underline{v}_j$ and $\underline{g}_j$ are the columns of V and G. Note that (2.26) corresponds to the linear system resulted from the spectral-element approximation of the one-dimensional equation

$$(\alpha + \lambda_j)v_j - \partial_{xx}v_j = g_j, \quad j = 1, \ldots, M_y.$$

We summarize below the steps and operation counts to solve the linear system (2.23).

Step	Operation count
Preprocessing	
Preparation for solving 1D problem	$\mathcal{O}(M_x)$
Solve generalized eigenvalue problem	$\mathcal{O}(M_y^3)$
Solving	
Multiply by $I \otimes E^T$	$\mathcal{O}(M_x M_y^2)$
Solve 1D problems	$\mathcal{O}(M_x M_y)$
Multiply by $I \otimes E$	$\mathcal{O}(M_x M_y^2)$

Note that we can choose to diagonalize in the direction with fewer points so that the cost of solving above becomes of order $\mathcal{O}(M_x M_y \min(M_x, M_y))$.

Finally, the complete procedure for solving (2.21) from the input function f at the collocation points to the approximation $u_{\mathbb{M}}$ at the collocation points also involve, in addition to the solution of (2.23), the computation of $\underline{f}$ and the evaluation of $u_{\mathbb{M}}$ at the collocation points from the coefficients u_{ij}. These two procedures, which are essentially the transformation between the physical values at collocation points and vice versa, involve basically two matrix multiplications which can be performed efficiently in parallel thanks to the localized structure. Note also that after including the costs of these two matrix multiplications, the cost of the whole solver (excluding preprocessing) is still of the order estimated above.

When comparing to the domain decomposition method usually used with spectral-element discretization, the new approach results in 1D problems of much smaller size. The size of the Schur complement in the domain decomposition method is of order $\mathcal{O}(K_x K_y(N_x + N_y))$, while the new approach results in $\mathcal{O}(\min(M_x, M_y))$ decoupled tridiagonal systems each of size $\mathcal{O}(\max(M_x, M_y))$.

2.2.2. Three-dimensional rectangular box

The 3D Helmholtz equation

$$\alpha u - u_{xx} - u_{yy} - u_{zz} = f$$

in a rectangular box can be handled similarly. In this case the Galerkin method reduces to the linear system

$$\mathcal{L}\underline{u} := (\alpha A^x \otimes A^y \otimes A^z + B^x \otimes A^y \otimes A^z + A^x \otimes B^y \otimes A^z + A^x \otimes A^y \otimes B^z)\underline{u} = \underline{f}, \tag{2.27}$$

where the A-matrices and B-matrices are, respectively, the mass and stiffness matrices in the corresponding directions (see (2.22)). Assume Λ and E solve the generalized eigenvalue problem

$$B^z E = A^z E \Lambda.$$

Then

$$\left(I \otimes I \otimes E^{\mathrm{T}}\right) \times \mathcal{L} \times (I \otimes I \otimes E) = (\alpha A^x \otimes A^y + B^x \otimes A^y + A^x \otimes B^y) \otimes I + A^x \otimes A^y \otimes \Lambda.$$

It follows that

$$\mathcal{L}^{-1} = (I \otimes I \otimes E) \times ((\alpha A^x \otimes A^y + B^x \otimes A^y + A^x \otimes B^y) \otimes I + A^x \otimes A^y \otimes \Lambda)^{-1} \times \left(I \otimes I \otimes E^{\mathrm{T}}\right).$$

Hence the linear system (2.27) reduces to a set of decoupled linear system of the form

$$((\alpha+\lambda_j)A^x\otimes A^y + B^x\otimes A^y + A^x\otimes B^y)\underline{v}_j = \underline{g}_j,$$

which corresponds to the 2D Helmholtz equations

$$(\alpha+\lambda_j)v_j - \partial_x^2 v_j - \partial_y^2 v_j = g_j$$

in a rectangle.

2.2.3. Cylindrical case

Consider now the domain between two concentric cylinders described in (2.3). Writing Eq. (2.1) in the cylindrical-polar coordinates and applying a Fourier transform to (2.1) in the θ direction, we find that each Fourier coefficient of the solution will satisfy the following two dimensional equation (see, for instance, [21] for more details):

$$\begin{aligned} &\alpha u - \frac{1}{r}(ru_r)_r + \frac{\gamma}{r^2}u - u_{zz} = f, \quad \text{in } \Omega = \Omega_r\times\Omega_z = (r_-, r_+)\times(z_-, z_+);\\ &u|_{\partial\Omega} = 0, \end{aligned} \tag{2.28}$$

where γ is a non-negative constant related to the Fourier mode. Then, the variational formulation for (2.28) is still (2.4) with

$$\begin{aligned} &\mathscr{A}(u,v) := \alpha(u,v)_r + (u_r, v_r)_r + \gamma\left(\frac{u}{r}, \frac{v}{r}\right)_r + (u_z, v_z)_r,\\ &\mathscr{F}(v) := (f, v)_r, \end{aligned}$$

where $(u,v)_r = \int_\Omega uvr\,\mathrm{d}r\,\mathrm{d}z$.

Let $\{\phi_i^z\}_{i=1}^{M_z}$ and $\{\phi_i^r\}_{i=1}^{M_r}$ be, respectively, the set of one-dimensional basis functions described previously for (2.10) and (2.13). Let us define A^r and B^r, matrices of size M_r, and A^z and B^z, matrices of size M_z, with elements

$$A^r_{ij} = (\phi_j^r, \phi_i^r)_r, \quad B^r_{ij} = (\partial_r\phi_j^r, \partial_r\phi_i^r)_r + \gamma\left(\frac{\phi_j^r}{r}, \frac{\phi_i^r}{r}\right)_r, \quad A^z_{ij} = (\phi_j^z, \phi_i^z), \quad B^z_{ij} = (\partial_z\phi_j^z, \partial_z\phi_i^z). \tag{2.29}$$

Then the Galerkin method reduces to the linear system

$$\mathscr{L}\underline{u} := ((\alpha A^r + B^r)\otimes A^z + A^r\otimes B^z)\underline{u} = \underline{f}, \tag{2.30}$$

which is ready to be solved by the matrix diagonalization method. Here, one can choose to diagonalize in either the r- or z-direction. Diagonalizing in the r-direction results in linear systems correspond to 1D equations of the form

$$(\alpha+\lambda)v - v_{zz} = g.$$

Since the constant γ present in the matrix B^r is related to the Fourier mode, solving the full 3D problem will involve solving an eigenvlaue problem for each Fourier mode. On the other hand, diagonalizing in the z-direction results in linear systems correspond to one-dimensional equations of the form

$$(\alpha+\lambda)v - \frac{1}{r}(rv_r)_r + \frac{\gamma}{r^2}v = g.$$

In this case the same eigen-pair can be used for all Fourier modes.

3. Parallel implementation

Consider first the 1D case. We use K processors (P_1 to P_K) to solve the 1D problem (2.1), where processor k works on the interval Ω_k. In other words, processor k has $f|_{\overline{\Omega}_k}$ on input, and computes $\{u^1_{k,i},\ i = 1, N-1\}$ and u^2_{k-1}, u^2_k. Each processor can compute A^k, B^k and $\widetilde{B}^k := (A^k)^{-1}B^k$ independently.

Hence communication between processors is needed only in forming and solving the tridiagonal system $\widetilde{C}\underline{u}^2 = \underline{g}^2$. Define the inner-product

$$(u, v)_k = \int_{\Omega_k} uv \, dx \quad \forall u, v \in L^2(\Omega_k),$$

and let $\mathscr{A}^k$, $\mathscr{F}^k$ be defined as in (2.5) with the inner-product $(\cdot,\cdot)$ replaced by $(\cdot,\cdot)_k$. The quantities that processor k can compute with information on hand are

$$c_1^k = \mathscr{A}^k(\psi_{k-1}, \psi_{k-1}), \quad c_2^k = \mathscr{A}^k(\psi_{k-1}, \psi_k), \quad c_3^k = \mathscr{A}^k(\psi_k, \psi_k),$$
$$\underline{f}^{1,k} = (f_{k,1}^1, \ldots, f_{k,N-1}^1)^{\mathrm{T}}, \quad \underline{g}^{1,k} = (A^k)^{-1}\underline{f}^{1,k},$$
$$\tilde{f}_1^k = \mathscr{F}^k(\psi_{k-1}), \quad \tilde{f}_2^k = \mathscr{F}^k(\psi_k).$$

The relationships between these quantities and the entries of C and $\underline{f}^2$ are

$$c_{k,k} = c_3^k + c_1^{k+1}, \quad k = 1, \ldots, K-1,$$
$$c_{k-1,k} = c_{k,k-1} = c_2^k, \quad k = 1, \ldots, K,$$
$$f_k^2 = \tilde{f}_2^k + \tilde{f}_1^{k+1}, \quad k = 1, \ldots, K-1.$$

Let $\underline{b}_1^k$, $\underline{b}_2^k$ be the nonzero columns of B^k. Recall that

$$\widetilde{C} = C - B^{\mathrm{T}}A^{-1}B, \quad \underline{g}^2 = \underline{f}^2 - B^{\mathrm{T}}\underline{f}^1,$$

the entries of $\widetilde{C}$ and $\underline{g}^2$ are hence

$$\tilde{c}_{k,k} = \tilde{c}_1^k + \tilde{c}_2^{k+1}, \quad k = 1, \ldots, K-1,$$
$$\tilde{c}_{k-1,k} = \tilde{c}_{k,k-1} = c_{k-1,k} - (\underline{b}_1^k)^{\mathrm{T}}(A^k)^{-1}\underline{b}_2^k, \quad k = 1, \ldots, K,$$
$$g_k^2 = \tilde{g}_1^k + \tilde{g}_2^{k+1}, \quad k = 1, \ldots, K-1,$$

where

$$\tilde{c}_1^k = c_3^k - (\underline{b}_2^k)^{\mathrm{T}}(A^k)^{-1}\underline{b}_2^k, \quad \tilde{c}_2^k = c_1^k - (\underline{b}_1^k)^{\mathrm{T}}(A^k)^{-1}\underline{b}_1^k,$$
$$\tilde{g}_1^k = \tilde{f}_2^k - (\underline{b}_2^k)^{\mathrm{T}}\underline{g}^{1,k}, \quad \tilde{g}_2^k = \tilde{f}_1^k - (\underline{b}_1^k)^{\mathrm{T}}\underline{g}^{1,k}.$$

Note that the superscripts on the right-hand sides also denote the precessor in which the value is stored. Hence the coefficients and right-and side of the equation

$$\tilde{c}_{k-1,k}u_{k-1}^2 + \tilde{c}_{k,k}u_k^2 + \tilde{c}u_{k+1}^2 = g_k^2$$

are stored partly ($\tilde{c}_{k-1,k}$, $\tilde{c}_1^k$ and $\tilde{g}_1^k$) in processor k and partly ($\tilde{c}_2^{k+1}$, $\tilde{c}_{k+1,k}$ and $\tilde{g}_2^{k+1}$) in processor $k+1$. In other words, processors 1 and K have part of one equation, while the other processors have part of two equations.

In the multi-dimensional case, assume that the processors are mapped onto a Cartesian grid. We can solve the eigenvalue problem (2.24) and perform the matrix-matrix multiplications in parallel easily with, for example, the ScaLAPACK [4]. If one of the dimensions is periodic, the Fourier transform can be performed in parallel through a parallel matrix transposition. Hence the main issue is to solve the tridiagonal systems (corresponds to the 1D problems resulted from matrix diagonalization, see (2.26)) in parallel.

3.1. Solving the tridiagonal systems

A standard algorithm to solve the tridiagonal system is to perform reduction in a top-down fashion (forward sweep) followed by a backward substitution in a bottom-up fashion (backward sweep). However, a parallel code using the above algorithm would require $2K-2$ (K being the number of processors) parallel communications and can be very slow on clusters. While there are parallel tridiagonal solvers that only require

$\mathcal{O}(\log_2 K)$ parallel communications, they are not directly applicable to our case, since the processors do not store full equations on input. In the following, we develop algorithms specially designed for our situation.

For the sake of simplicity (in describing the algorithms), we add two more equations (0 and K) to the tridiagonal system, so that there are $K+1$ equations to solve and processor k has some information of equations $k-1$ and k. We discuss here two methods to solve the tridiagonal systems in parallel. For simplicity, we only illustrate the case with only one system, i.e. the 1D case. The multi-dimensional case can be easily handled by extending the messages in communications to M times longer. We also assume that K is an integral power of 2.

We describe the two methods below. The first one is suitable in case the communication between processors is bidirectional, while the second one is preferable if the communication between processors is multi-port bidirectional.

The first method uses a recursive algorithm as follows:

1. For k odd, P_K sends its part of equation k (i.e., $\tilde{c}_{k-1,k}$, $\tilde{c}_1^k$ and $\tilde{g}_1^k$) to P_{k+1}; P_{k+1} sends its parts of equations k (i.e., $\tilde{c}_2^{k+1}$, $\tilde{c}_{k+1,k}$ and $\tilde{g}_2^{k+1}$) and $k+1$ (i.e., $\tilde{c}_{k,k+1}$, $\tilde{c}_1^{k+1}$ and $\tilde{g}_1^{k+1}$) to P_K.
2. For k odd, P_K uses the full equation k to eliminate unknown u_k^2 in equations $k-1$ and $k+1$; P_{k+1} keeps the full equation k.
3. The odd-numbered processors altogether now contains the whole system of even-numbered equations. The problem is now reduced to one with half size. The recursion ends when processors 1 and $K/2+1$ altogether contains the whole system involving u_0^2, $u_{K/2}^2$ and u_K^2. In this case the two processors exchange information. Processor 1 solve for u_0^2 and $u_{K/2}^2$. Processor $K/2+1$ solves for $u_{K/2}^2$ and u_K^2.
4. For k odd, P_K sends the values of u_{k-1}^2 and u_{k+1}^2 to P_{k+1}; P_{k+1} sends the full equation k to P_K.
5. For k odd, P_K solves the full equation k and keep u_{k-1}^2, u_k^2; P_{k+1} solves the full equations k and keep u_k^2, u_{k+1}^2.

Fig. 1 shows the flow of communication with $K=8$, in which an arrow denotes a communication and the number above or below an arrow is the length of the message.

Assume that the communication is *bidirectional*. Let t_s be the start-up time for a communication and t_w be the time for a message of unit length to be sent over the network. Then the parallel run time for the above algorithm is

$$T_K = (t_s + t_w \cdot 5M) + T_{K/2} + (t_s + t_w \cdot 4M).$$

Solve the above recursive relation with

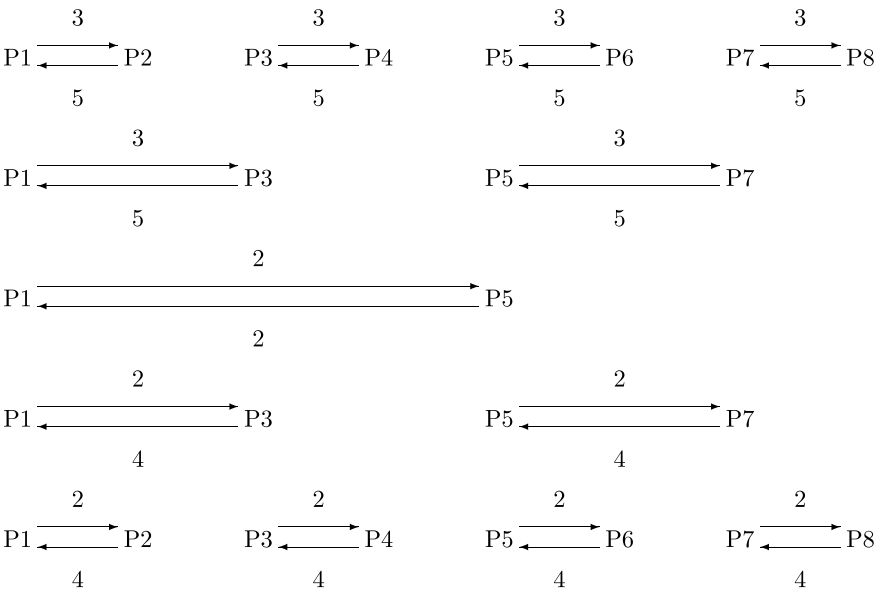

Fig. 1. Flow of communication for the first method.

$$T_2 = t_s + t_w \cdot 2M$$

gives

$$T_K = (2t_s + 9Mt_w)\log_2 K - (t_s + 7Mt_w).$$

The second method is the same as the first one up to step 3. The modification is as follows. Note that after step 3, except for P_1 and $P_{K/2}+1$, all processors contain an equation involving three unknowns. Thus these processors only need two solved unknowns to solve for another unknown. This cuts the message size to two (instead of four when sending the full equation, see step 4 of the first method). The algorithm is quite complicated and is best described by the flow of communication, which is shown in Fig. 2 in the case $K = 8$. Note that in the lower part of the figure, the numbers above an arrow is the indices of the solved unknowns being sent instead of the size of the message. Note that in the second-last step, the processors P_3 and P_7 send solved unknowns to two other processors. Hence the methods work best when the communication is *multi-port*.

Assume that the communication is *multi-port* and *bidirectional*. Then the parallel run time of this algorithm for $K > 2$ is

$$\begin{aligned} T_K &= (t_s + 5Mt_w)(\log_2 K - 1) + (ts + 2Mt_w) + (t_s + 2Mt_w)(\log_2 K - 1) + (t_s + Mt_w) \\ &= (2t_s + 7Mt_w)\log_2 K - 4Mt_w. \end{aligned}$$

4. Numerical results

We now present several examples which illustrate some of the advantages of this algorithm. All numerical experiments are performed on the IBM-SP2 cluster in Purdue University. The cluster contains 320–375 MHz POWER3-II processors. All codes are written in Fortran77 with MPI.

Example 1. We consider the Poisson equation in a layered medium

$$-(a(x)u_x)_x - u_{yy} = 1, \quad 0 < x < 4, \ \ 0 < y < 1 \tag{4.1}$$

with homogeneous Dirichlet boundary conditions, where

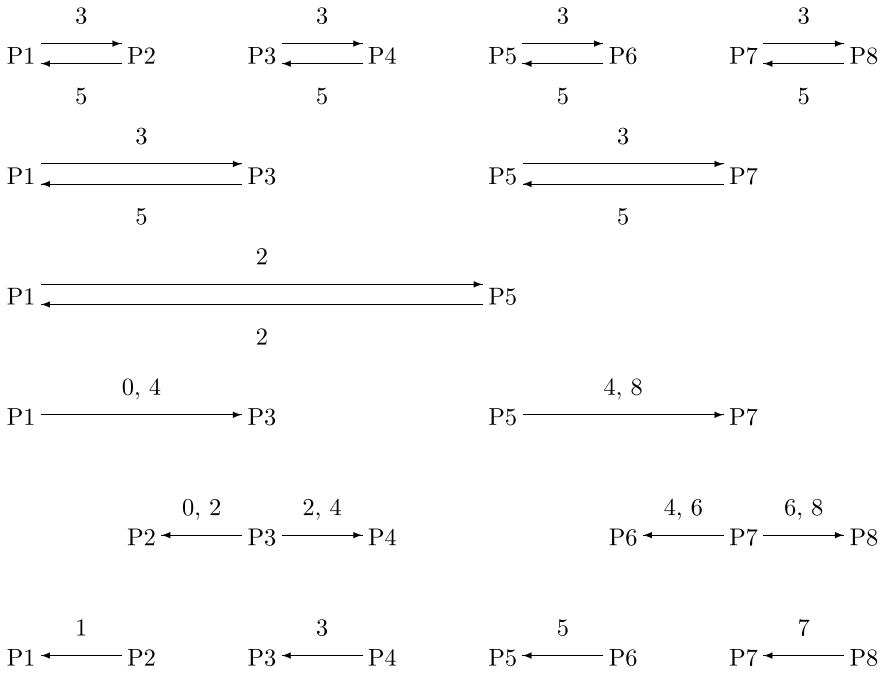

Fig. 2. Flow of communication for the second method.

$$a(x) = \begin{cases} 1.0, & 0 < x < 1, \\ 1.5, & 1 < x < 2, \\ 2.0, & 2 < x < 3, \\ 2.5, & 3 < x < 4, \end{cases}$$

is a piecewise constant function. It is clear that a direct spectral approximation will lead to an inaccurate result with large oscillations around the discontinuities of $a(x)$. This difficulty can be easily overcome by using a spectral-element method such that the coefficients are smooth within each element, for it has been shown (cf., for instance, [2]) that the convergence rate of the spectral-element method is affected by the "local smoothness" of the coefficients $a(x)$, not the "global smoothness". Here, we implemented a spectral-element method with four equally spaced intervals in the x-direction and one single interval in the y-direction. In Table 1, we list the ℓ^2-errors of the spectral-element approximation with different resolutions. The "exact" solution is computed with a high resolution $N_x = N_y = 256$. It is clear that the convergence rate of the approximate solution is not affected by the discontinuities of $a(x)$, it is, however, limited by the corner singularity.

Example 2. To illustrate the efficiency of the parallel implementation, the 3D Helmholtz equation in a rectangular box is solved using different number of processors. In the parallel run, the number of processors used is equal to the number of elements K in the x-direction. We fix $N_y = N_z = 48$ while N_x is chosen such that the total degree of freedom is constant. The serial and parallel runtimes (excluding preprocessing time) for different number of processors are shown in Table 2. We observe from the table that the parallel efficiency is essentially 75% or above for all cases.

Example 3. To further investigate the parallel efficiency, we solve the axi-symmetric Helmholtz equation in a cylinder (2.28) by subdividing the domain into a 2D grid and using one processor on each subdomain. For each subdomain, we fix $N_r = N_z = 64$ and report in Table 3 the serial and parallel runtimes (excluding time to solve the eigenvalue problem (2.24)) for different number of processors. We observe again from the table that high parallel efficiency (>80%) is achieved for all cases. The preprocessing time on a serial processor is also reported and is roughly the same as the serial runtime for the solver. Note that the computational complexities for the solver and the preprocessing are $\mathcal{O}(K_r^2 N_r^2 K_z N_z)$ and $\mathcal{O}(K_z^3 N_z^3)$, respectively.

Example 4. The goal of this last example is to validate this spectral-element solver in a complex computational situation and set the stage for future large scale parallel computations.

We consider the retraction of a cylindrical filament of radius $R = 0.25$ and height $H = 5.5$ placed at the axis of a cylinder of radius $R = 1$ and height $H = 6$ filled with another ambient fluid, where the two fluids are incompressible and have the same density and viscosity, we refer to [14] for more detail on the background of this problem. Using the phase-field model, the system governing the mixture of the two fluids can be written as

$$\begin{aligned} &\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p + \lambda\nabla\cdot(\nabla\phi\otimes\nabla\phi) = \mathbf{f}, \\ &\nabla\cdot\mathbf{u} = 0, \end{aligned} \tag{4.2}$$

and

$$\begin{aligned} &\phi_t + (\mathbf{u}\cdot\nabla)\phi = \gamma(\Delta\phi - f(\phi) + \xi(t)), \\ &\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \phi\,\mathrm{d}x = 0, \end{aligned} \tag{4.3}$$

Table 1
ℓ^2-Errors for Example 1

$N_x = N_y$	8	16	32	64
ℓ^2-Error	9.41E−6	2.24E−7	4.43E−9	7.82E−11

Table 2
Parallel efficiencies for different K in Example 2

N_x	K	Serial runtime	Parallel runtime	Parallel efficiency
1024	1	23.7534	–	–
512	2	10.9372	5.7433	0.9522
256	4	7.2480	2.4007	0.7548
128	8	5.7776	0.9698	0.7447
64	16	7.0883	0.4522	0.9796
32	32	6.8300	0.2692	0.7928

Table 3
Parallel efficiencies for different K in Example 3

$K_r \times K_z$	Serial runtime (preprocessing)	Parallel runtime	Parallel efficiency
2×2	0.1866 (0.1271)	0.04777	0.9766
4×4	0.9138 (0.8127)	0.06149	0.9288
8×8	5.1930 (6.0020)	0.08863	0.9155
16×16	32.8372 (43.8713)	0.1569	0.8175

where ϕ is the phase function ($\phi = 1$ inside the filament and $\phi = -1$ in the ambient fluid), λ is the ratio between the kinetic energy and the elastic energy, γ is the elastic relaxation time, $\xi(t)$ is the Lagrange multiplier corresponding to the constant volume constraint in the last equation, and

$$f(\phi) = \frac{1}{\eta^2}\phi(\phi^2 - 1) \tag{4.4}$$

with η the thickness of the transition layer. We refer to [24] for a detailed discussion of this model and its numerical method. It is shown in [24] that using a combination of a consistent splitting scheme [9] and a stabilized semi-implicit discretization for the phase equation, one only needs to solve a sequence of Poisson-type equations at each time step. While a spectral-Galerkin method was used in [24], we implemented the spectral-element algorithm for the same problem.

The parameters used are

$$\gamma = 0.02, \quad \lambda = 0.01, \quad \eta = 0.02, \quad \nu = 0.02, \quad \delta t = 0.005.$$

The computation is carried out with 6 equally spaced elements in the z-direction and 1 element in the r-direction. The resolution used is $N_x = N_y = 64$. Fig. 3 shows the contours of the phase function ϕ with values -0.5 and 0.5 at different times. These results are in good agreement with those reported in [24].

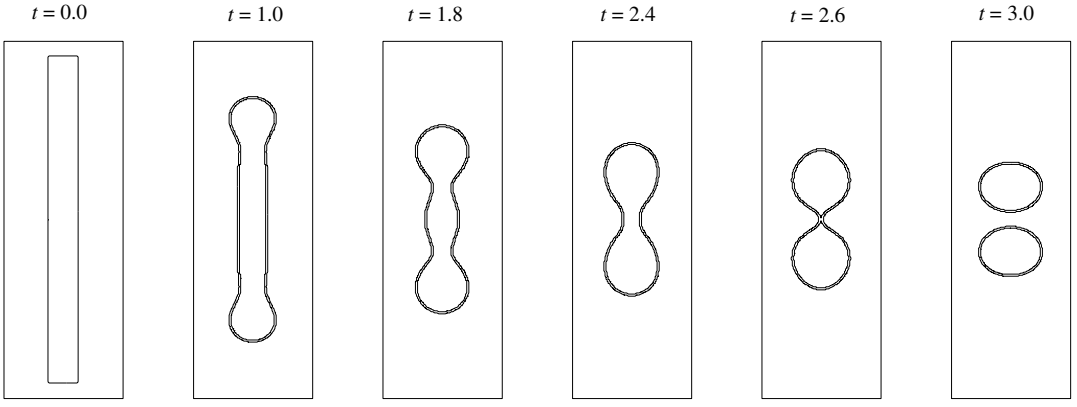

Fig. 3. Contours of the phase function ϕ at different time.

5. Concluding remarks

We presented an efficient, direct, and highly parallel spectral-element algorithm for elliptic equations in separable domains. The solver is particular suitable for domains with large aspect ratio and for problems with layer structures/discontinuous coefficients. Ample numerical experiments indicates that our algorithm achieves very high parallel efficiency ($\geqslant 75\%$) for all tested cases which use up to 256 (the largest number of nodes that we have access to) processors. It is expected that this parallel solver can be used as an essential building block for large-scale computations with implicit or semi-implicit schemes on massively parallel computers.

Acknowledgments

The authors thank Dr. Paul Fischer for suggesting this work and for hosting Kwan as a summer intern at Argonne National Laboratory. This work is partially supported by NSF Grants DMS-0509665 and DMS-0610646.

References

[1] I. Babuška, A. Craig, J. Mandel, J. Pitkäranta, Efficient preconditioning for the *p*-version finite element method in two dimensions, SIAM J. Numer. Anal. 28 (3) (1991) 624–661.
[2] Christine Bernardi, Yvon Maday, Francesca Rapetti, Discrétisations variationnelles de problèmes aux limites elliptiques, Mathématiques & Applications (Berlin) [Mathematics & Applications], vol. 45, Springer, Berlin, 2004.
[3] Petter E. Bjørstad, Bjørn Peter Tjøstheim, Efficient algorithms for solving a fourth-order equation with the spectral-Galerkin method, SIAM J. Sci. Comput. 18 (2) (1997) 621–632.
[4] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevado, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.
[5] John P. Boyd, Chebyshev and Fourier Spectral Methods, second ed., Dover, Mineola, NY, 2001.
[6] B.L. Buzbee, G.H. Golub, C.W. Nielson, On direct methods for solving Poisson's equations, SIAM J. Numer. Anal. 7 (1970) 627–656.
[7] W. Couzy, M.O. Deville, A fast Schur complement method for the spectral element discretization of the incompressible Navier–Stokes equations, J. Comput. Phys. 116 (1) (1995) 135–142.
[8] M.O. Deville, P.F. Fischer, E.H. Mund, High-order methods for incompressible fluid flow, Cambridge Monographs on Applied and Computational Mathematics, vol. 9, Cambridge University Press, Cambridge, 2002.
[9] J.L. Guermond, Jie Shen, A new class of truly consistent splitting schemes for incompressible flows, J. Comput. Phys. 192 (1) (2003) 262–276.
[10] P. Haldenwang, G. Labrosse, S. Abboudi, M. Deville, Chebyshev 3-D spectral and 2-D pseudospectral solvers for the Helmholtz equation, J. Comput. Phys. 55 (1) (1984) 115–128.
[11] G.E. Karniadakis, S.J. Sherwin, Spectral/hp Element Methods for CFD, Oxford University Press, Oxford, 1999.
[12] George Em. Karniadakis, Spencer J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, second ed., Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.
[13] Robert E. Lynch, John R. Rice, Donald H. Thomas, Direct solution of partial difference equations by tensor product methods, Numer. Math. 6 (1964) 185–199.
[14] P.K. Notz, O.A. Basaran, Dynamics and breakup of a contracting liquid filament, J. Fluid Mech. 512 (2004) 223–256.
[15] Anthony T. Patera, Fast direct Poisson solvers for high-order finite element discretizations in rectangularly decomposable domains, J. Comput. Phys. 65 (2) (1986) 474–480.
[16] Luca F. Pavarino, Olof B. Widlund, Iterative substructuring methods for spectral element discretizations of elliptic systems. I. Compressible linear elasticity, SIAM J. Numer. Anal. 37 (2) (2000) 353–374.
[17] Alfio Quarteroni, Alberto Valli, Domain Decomposition Methods for Partial Differential Equations, Numerical Mathematics and Scientific Computation, The Clarendon Press, Oxford University Press, New York, 1999.
[18] Jie Shen, Efficient spectral-Galerkin method. I. Direct solvers of second- and fourth-order equations using Legendre polynomials, SIAM J. Sci. Comput. 15 (6) (1994) 1489–1505.
[19] Jie Shen, Efficient spectral-Galerkin method. II. Direct solvers of second- and fourth-order equations using Chebyshev polynomials, SIAM J. Sci. Comput. 16 (1) (1995) 74–87.
[20] Jie Shen, Efficient Chebyshev–Legendre Galerkin methods for elliptic problems, in: A.V. Ilin, R.R. Scott (Eds.), Proceedings of ICOSAHOM'95, Houston J. Math, 1996, pp. 233–240.
[21] Jie Shen, Efficient spectral-Galerkin methods. III. Polar and cylindrical geometries, SIAM J. Sci. Comput. 18 (6) (1997) 1583–1604.
[22] Andrea Toselli, Olof Widlund, Domain decomposition methods – algorithms and theory, Springer Series in Computational Mathematics, vol. 34, Springer, Berlin, 2005.

[23] H.M. Tufo, P.F. Fischer, Terascale spectral element algorithms and implementations, in: Proceedings of Supercomputing, Gordon and Bell, London, 1999.
[24] X. Yang, J.J. Feng, C. Liu, J. Shen, Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method, J. Comput. Phys. 218 (2006) 417–428.
[25] Thomas Zang, Dale B. Haidvogel, The accurate solution of Poisson's equation by expansion in Chebyshev polynomials, J. Comput. Phys. 30 (2) (1979) 167–180.