Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 184786, 15 pages
http://dx.doi.org/10.1155/2015/184786

Research Article

A Fourier Continuation Method for the Solution of Elliptic Eigenvalue Problems in General Domains

Oscar P. Bruno, Timothy Elling, and Ayon Sen

Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125, USA

Correspondence should be addressed to Oscar P. Bruno; obruno@caltech.edu

Received 2 July 2015; Accepted 25 October 2015

Academic Editor: Salvatore Alfonzetti

We present a new computational method for the solution of elliptic eigenvalue problems with variable coefficients in general two-dimensional domains. The proposed approach is based on use of the novel Fourier continuation method (which enables fast and highly accurate Fourier approximation of nonperiodic functions in equispaced grids without the limitations arising from the Gibbs phenomenon) in conjunction with an overlapping patch domain decomposition strategy and Arnoldi iteration. A variety of examples demonstrate the versatility, accuracy, and generality of the proposed methodology.

1. Introduction

We present a new computational method for the solution of variable-coefficient elliptic eigenvalue problems in general two-dimensional domains. This method uses Arnoldi iteration to approximate the eigenvalues of numerical differential operators in such a way that the bulk of the associated computational cost arises from the needed repeated evaluations of the differential operator for given input functions (vectors). For accuracy and efficiency our algorithm produces such evaluations on the basis of the Fourier continuation (FC) method [1] which, using equispaced meshes, resolves the inaccuracies due to the Gibbs phenomenon by computing smooth, periodic extensions of arbitrary smooth functions, therefore allowing for accurate evaluation of derivatives in the frequency domain. The efficiency of the method thus results from its reliance on the Fast Fourier Transform (FFT) for numerical differentiation.

Differential eigenvalue problems arise in many areas of physical science; a prominent example is the Laplace eigenvalue problem:

$$\nabla^2 u(x, y) = \lambda u(x, y). \quad (1)$$

In the simplest physics perspective the eigenvalues correspond to the fundamental modes of vibration of a thin membrane with geometry corresponding to the domain of (1). Solutions to (1) completely characterize membrane motions since the eigenfunctions form a complete orthonormal basis for the space of membrane vibrations. Other physically relevant applications of the Laplace eigenproblem include the description of quantized energy states of particles (under various potentials) and the dynamics of electromagnetic waves traveling through waveguides [2]. Chain reactions in nuclear reactors are described by a more complicated eigenvalue problem, in which the smallest-magnitude eigenvalue describes whether the reaction is subcritical, critical, or supercritical [3]. Because natural processes do not only occur in simple separable geometric regions such as squares or disks, the only types of regions on which (1) can be solved analytically, much effort has been invested in the solution of eigenvalue problems on general domains.

Existing approaches for differential eigenvalue problems include collocation methods as well as finite element or finite difference methods. Collocation methods utilize expressions for the eigenfunction in terms of adequately chosen bases. A classical example is the method of particular solutions, in which the coefficients in the expansion are obtained via consideration of a homogeneous system of linear equations that enforces the required (homogeneous) boundary conditions at points placed along the domain boundary [4, 5]. A noteworthy aspect of this method is its use of singular (Fourier-Bessel) functions in the expansion to incorporate, for example, reentrant corners. Improvements have been made by including points on the interior of the domain where the condition that the eigenfunction is nonzero is enforced [5]. Similar approaches were put forth in [6, 7] with different choices for the basis functions. Other generalizations include the use of conformal mappings to recast the problem in a circular domain as a generalized eigenvalue problem, as is done in [8] for domains with approximately fractal boundaries.

Other eigenvalue solvers utilize finite difference or finite element approximations which reduce differential eigenvalue problem to a corresponding finite-dimensional algebraic eigenvalue problem. Finite element methods rely on unstructured meshes that can be generated and adaptively refined to account for complicated domain features [9]. Domain decomposition techniques are often used in conjunction with finite elements in order to deal with discontinuous coefficients or to reduce the problem to a number of smaller problems [10], as are iterative eigenvalue algorithms [11]; the algorithm proposed in this paper utilizes both domain decomposition and iterative methods. An extension of these ideas can be found in the method given by [12], which uses the basis from the method of particular solutions and domain decomposition, allowing for local usage of the basis near singularities. Results from a wide array of these types of methods are summarized in [2, 13].

Even though collocation methods have demonstrated highly accurate evaluation of eigenvalues for some singular geometries (such as the L-shaped membrane), the applicability of these methods is limited to locally separable geometries. As indicated above, use of conformal maps has been suggested in the literature as a remedy for this difficulty [6], but determining a mapping from a simple shape to an arbitrary region is a nontrivial task. Finite element methods can also be effective, and they are applicable to rather general configurations. Finite difference methods, finally, are efficient by virtue of their finite bandwidths, but this advantage turns into a challenge near domain boundaries. Even high-order centered finite difference schemes must be reformulated near boundaries, resulting in an asymmetric stencil and generally a lower order of accuracy at the boundary [14].

To address the difficulties arising in the treatment of general elliptic eigenvalue problems we propose a methodology wherein derivatives are produced by means of the FC(Gram) algorithm [1]. Differentiation of Fourier series is, of course, straightforward and, in view of the Fast Fourier Transform, can be produced in $O(N \log N)$ operations, where N is the number of points in the mesh. Because the FC(Gram) algorithm produces smooth periodic extensions for given functions, Fourier series truncated to a small number of terms are still highly accurate representations of the continued function: the difficulties arising from the Gibbs phenomenon are eliminated. As a result of the proposed FC-based domain decomposition approach, eigenvalue problems for complicated geometries and general smooth coefficients can be tackled with high accuracy within the FC framework. Additionally, the proposed domain decomposition approach can be used to produce local mesh refinements as well as efficient parallel implementations.

The capabilities afforded by the proposed methodologies are demonstrated by means of a variety of examples in two-dimensional space. Extensions of this framework to higher dimensions, which are straightforward, are not otherwise pursued in this paper.

2. Definitions

2.1. Overall Methodology. Our general approach relies on an decomposition of the domain into overlapping patches that are expressed in terms of curvilinear coordinates (i.e., mapped to rectangles with uniform grids), using parametrizations to describe the differential operator in terms of Cartesian reference coordinates, which in what follows are denoted by s and t. On each patch, boundary conditions and continuity are imposed, and then the operator is evaluated using Fast Fourier Transforms. The numerical differential operator thus defined on the whole domain can then be used in conjunction with an iterative method for eigenvalue evaluation. The basic concepts and notations used throughout this process are discussed in what follows.

2.2. Eigenvalue Problem. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain and let L be a linear elliptic operator of the form $L = \sum_{|\alpha| \le m} a_\alpha(x, y)\partial^\alpha$, where α is a two-dimensional multi-index. Throughout this paper, we search for a subset of the scalar values λ and corresponding functions $u \in C^\infty(\Omega, \mathbb{R})$ such that

$$\begin{aligned} Lu &= \lambda u, \\ u|_{\partial\Omega_1} &= 0, \\ \left.\frac{\partial u}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_2} &= 0, \end{aligned} \tag{2}$$

where λ are the eigenvalues and $u(x, y)$ are the eigenfunctions. The boundary conditions are defined so that $\partial\Omega_1 \cup \partial\Omega_2$ is a disjoint union that partitions $\partial\Omega$ and $\hat{\mathbf{n}}$ denotes the outward unit normal. For simplicity we restrict the scope of our study to only consider fixed (Dirichlet) and free (Neumann) boundary conditions.

2.3. Domain Decomposition. Let $\{P_i\}_{i=1}^n$ be open sets relative to Ω such that there exist invertible smooth functions $f_i : R \to P_i$, where $R = [0, 1] \times [0, 1]$. We let (s, t) denote coordinates on R, which correspond to (x, y) on P_i under the mapping f_i. Using the chain rule to translate derivatives in domain coordinates to derivatives in reference coordinates (e.g., $u_s = x_s u_x + y_s u_y$) and inverting the resulting system yields the desired expressions for derivatives in x and y in terms of s and t. The derivatives in s and t are calculated numerically on the uniform reference grid using the FC(Gram) algorithm described in the following section. For the examples considered in this paper, the domain decompositions were achieved by inspection of the domain and trial-and-error manipulations, but a variety of algorithms are available that perform overlapping patch decompositions (e.g., [15]).

2.4. FC(Gram). Evaluation of derivatives in the Fourier basis offers two main advantages: under the Fourier Transform, differentiation corresponds to scalar multiplication, and, in view of the Fast Fourier Transform algorithm, numerical evaluation of the Fourier Transform scales well to large problems, since the cost of a Fourier Transform on n modes is $O(n \log n)$. As is known, however, truncated Fourier series provide poor representations for nonperiodic functions, an inaccuracy which is only compounded by differentiation. Specifically, the Fourier series oscillates rapidly at the endpoints of the domain whenever the periodic extension is discontinuous (Gibbs phenomenon). Since general functions to which the differential operator must be applied are not periodic, high orders of accuracy cannot be expected from straightforward uses of Fourier expansions.

This problem may be addressed by means of the Fourier continuation algorithm, which, given a function f, produces Fourier series for smooth periodic extensions of f to a suitably larger interval. Thus, given a function f defined on (say) $[0,1]$ which is sampled on the discrete grid given by $x_j = (j-1)h$, where $h = 1/(n-1)$ and $j = 1,\dots,n$, the algorithm constructs a new function f_c on $[0,b]$ for some $b > 1$ such that $f_c(x) = f(x)$, $\forall x \in [0,1]$, and $f_c^{(m)}(b) = f_c^{(m)}(0)$, $\forall m \le k$. In view of these conditions the function f_c is smooth and periodic, and therefore its b-periodic trigonometric polynomial,

$$f_c(x) = \sum_{k\in t(F)} a_k e^{(2\pi i/b)kx}, \tag{3}$$

(where $t(F)$ is the set of $n + n_c$ modes distributed symmetrically about 0) is highly accurate. To enforce the approximation of the function f, one must compute the coefficients a_k in (3) such that f_c and f agree on the n original sample points. This could, for example, be accomplished by solving a (possibly oversampled) minimization problem:

$$\min_{\{a_k | k\in t(F)\}} \sum_{j=1}^{n} \left| \sum_{k\in t(F)} a_k e^{(2\pi i/b)kx_j} - f(x_j) \right|^2. \tag{4}$$

Written as a matrix equation, this amounts to solving

$$\mathbf{Ma} = \mathbf{f}, \tag{5}$$

in the least-squares sense, where $\mathbf{a}$ is a column vector whose entries are the coefficients a_k, $\mathbf{f}$ is a column vector whose entries are $f(x_j)$, and $\mathbf{M}$ is a matrix with $M_{jk} = e^{(2\pi i/b)kx_j}$.

By itself, this procedure constructs the desired extension but is not desirable for high-efficiency computation, which is needed in the context of an iterative solver. This method can be modified, however, in such a way that only a constant number of data points are used in the calculation of the continuation for arbitrarily large values of n. A brief summary of the resulting approach is presented in what follows; full details can be found in [1, 16]. Consider the set of points $\{(x_{n-n_l-1}, f(x_{n-n_l-1})), \dots, (x_n, f(x_n))\} \cup \{(x_1 + 1 + d, f(x_1)), \dots, (x_{n_r} + 1 + d, f(x_{n_r}))\}$, where n_l and n_r are fixed integers chosen independently of n. By computing the trigonometric polynomial fit via (4) on this set of points, we can construct a function f_{match} such that

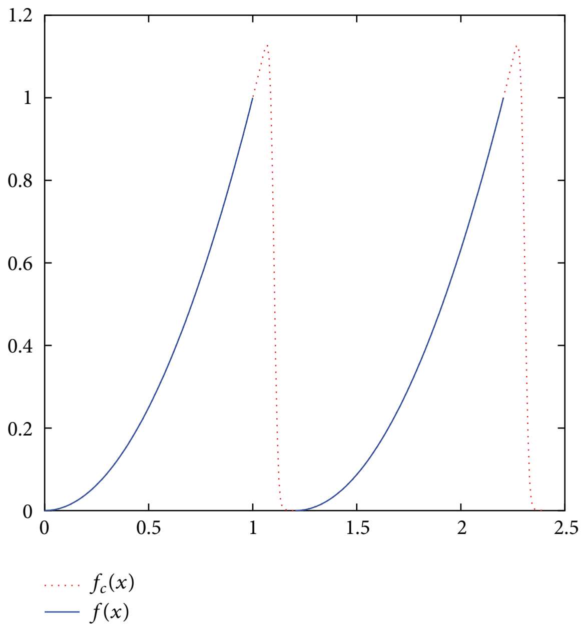


FIGURE 1: The function $f(x) = x^2$ on $[0,1]$ extended via the FC(Gram) algorithm to $f_c(x)$. The function has been extended so that it is smoothly periodic over a slightly larger domain.

$$f_c(x) = \begin{cases} f(x), & x \in [0,1], \\ f_{\text{match}}(x), & x \in (1, 1+d] \end{cases} \tag{6}$$

is $(1+d)$-periodic. A Fourier approximation of the smooth and periodic function f_c in the interval $[0, 1+d]$ (which also provides an approximation for f in the interval $[0,1]$) is highly accurate, and it provides, in particular, the needed Fourier expansion of the function f in the interval $[0,1]$.

It is possible to select n_l and n_r small while maintaining high accuracy in the continuation by expressing the aforementioned least-squares problem in terms of the orthonormal (Gram) polynomial basis for the intervals $[x_{n-n_l-1}, 1]$ and $[1+d, x_{n_r} + 1 + d]$. The algorithm is completed by relying on highly accurate precomputed extensions for pairs of Gram polynomials on the left and right subintervals, projection of a given function onto the Gram basis, and a subsequent use of the Fast Fourier Transform; see [1, 16] for details. An example of an extension computed via FC(Gram) is shown for a quadratic function on a closed interval in Figure 1. All FC(Gram) continuations in our numerical examples use $n_l = n_r = 10$ and a Gram polynomial basis up to degree 5.

2.5. Numerical Differentiation. If $f(x)$ is a 1-dimensional function whose continuation (computed via FC(Gram)) is

```
(1)  Procedure FCDIFF($u, k, h$)            ▷ returns $k$th derivative of $u$, where $h$ is the mesh step size
(2)      $u_c \leftarrow \text{cont}(u)$                ▷ perform FC(Gram) continuation
(3)      $\widehat{u_c} \leftarrow \text{fft}(u_c)$
(4)      for $j \leftarrow 1, n_c$ do                   ▷ $n_c$ is the length of $u_c$
(5)          $\widehat{u_c}[j] \leftarrow \widehat{u_c}[j] * (2\pi i/(n_c h))^k$
(6)      end for
(7)      $u_c \leftarrow \text{ifft}(\widehat{u_c})$
(8)      $u \leftarrow \text{restrict}(u_c)$            ▷ store the portion corresponding to the original domain
(9)      return $u$
(10) end procedure
```

Algorithm 1: FC(Gram) 1D kth-order differentiation.

$f_c(x)$, then the derivative of the Fourier series of f_c is computed as follows:

$$f_c(x) = \sum_{\ell=-\infty}^{\infty} f_\ell e^{i\omega_\ell x},$$
$$\frac{d^{(k)} f_c}{dx} = \sum_{\ell=-\infty}^{\infty} (i\omega_\ell)^k f_\ell e^{i\omega_\ell x}. \tag{7}$$

The derivative of f is recovered by restricting the domain of $f_c^{(k)}$. This method easily generalizes to multivariate functions defined on rectangular domains. In the discrete setting, we compute Fourier Transforms using the FFTW implementation [17] of the Fast Fourier Transform. The one-dimensional FC(Gram) differentiation scheme is summarized in Algorithm 1. Partial derivatives in two dimensions are computed numerically by applying the one-dimensional algorithm to each row (or column) of the equispaced grid in reference coordinates. This method is indeed accurate to high order, as shown in previous applications to ODE and PDE problems [1, 16, 18]. Since we use a 5th-order Gram basis to construct the fits, we expect Algorithm 1 to compute derivatives with fourth-order accuracy.

3. Solving the Eigenvalue Problem

3.1. Boundary Conditions. Enforcing a Dirichlet boundary condition is as simple as prescribing the value on a line of fringe points along the boundary. Due to the mapping f_i, the boundaries of R correspond exactly to the boundaries of P_i, so Dirichlet boundary conditions can be easily translated to reference coordinates. However, imposing a Neumann boundary condition is more complicated due to the fact that normal vectors are not held invariant by f_i. To translate the condition to reference coordinates, we again use the chain rule to write the normal derivative as a combination of change of variables terms and derivatives in parameter space (by inverting the Jacobian of the mapping f_i):

$$0 = \frac{\partial u}{\partial \hat{\mathbf{n}}} = \nabla u \cdot \hat{\mathbf{n}} = u_x n_1 + u_y n_2 = (x_s y_t - y_s x_t)^{-1} \cdot [u_s(n_1 y_t - n_2 x_t) + u_t(-n_1 y_s + n_2 x_s)] \coloneqq \gamma_1 u_s + \gamma_2 u_t. \tag{8}$$

The values γ_1 and γ_2 depend only on the unit outward normal $\hat{\mathbf{n}} = (n_1, n_2)^T$ and the partial derivatives of components of f_i (for a given patch P_i).

By treating the points on the boundary, where a Neumann condition is prescribed as unknowns, we will already have data from the candidate function u along the boundary. As a result, either u_s or u_t can be numerically computed along the boundary, which in turn allows us to describe the Neumann condition solely in terms of the other partial (i.e., by solving for it in (8)). This means that the condition on the normal derivative in domain space can be written as a condition on the normal derivative in reference coordinates. Knowing this, we use a least-squares system to fit a fifth-order 1D polynomial to each set of 10 grid points along the normal direction in parameter space. This polynomial is constructed with the condition that its derivative has the value derived from (8). Finally, the values at these grid points are replaced by the values from the polynomial fit when computing the Fourier continuation, but the original values are used for the calculation of the derivative (after continuation). This way, if a given function already obeys the Neumann condition, nothing is changed and the operator is computed properly. However, if the condition does not hold, this process introduces the desired discontinuity in the function which increases in magnitude when derivatives are computed.

3.2. Continuity. A similar method is used to enforce continuity across patches in overlap regions. Suppose P_i overlaps with P_j near one of its boundaries. A layer of fringe points is added to the patch on the overlap region. Then the function values for P_i on the fringe points are prescribed by interpolating from the function values in the interior of P_j (avoiding values in P_j that are themselves interpolated from other patches). We use Neville's algorithm for cubic polynomial interpolation and perform all interpolation in reference coordinates. A simple implementation of Newton's method is used to compute $f_j^{-1}(x, y)$ (when f_j^{-1} is not explicitly known) for a given point (x, y) at the boundary of P_i.

3.3. Eigenvalue Algorithm. The methodologies described above in this paper give rise to a numerical algorithm for evaluation of differential operators for a given geometry with a provided domain decomposition and boundary conditions. To compute the eigenvalues in (2), we combine this

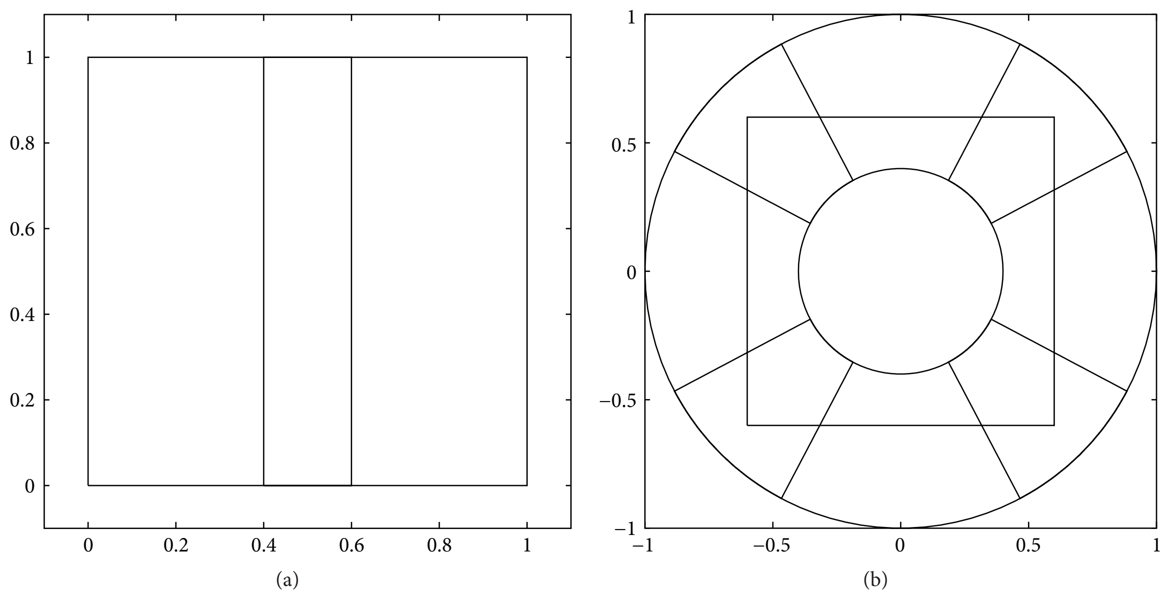


Figure 2: Decomposition of simple geometries: (a) the unit square, decomposed into two overlapping rectangles; (b) the unit disk, decomposed into eight overlapping sections of an annulus and a square centered at the origin. These examples demonstrate the usefulness of domain decomposition to avoid difficulties in mapping entire domains.

numerical differential operator with the ARPACK++ [19] implementation of the implicitly restarted Arnoldi method. The Arnoldi method iteratively builds a lower dimensional approximation of the operator L as an upper Hessenberg matrix $\mathbf{H}$. The eigenvalues of $\mathbf{H}$ ("Ritz values") are, in practice, good estimates of the eigenvalues of L, as are the eigenvectors (up to an orthogonal transformation). The special structure of $\mathbf{H}$ allows for efficient calculation of the Ritz values as well. Furthermore, we find that the implicitly restarted Arnoldi method performs better than the full Arnoldi method (i.e., without restarts); the restarted method is dominated mostly by the cost of evaluating the operator L rather than the cost of computing the eigenvalues of a large $\mathbf{H}$ matrix. Combined with acceleration techniques that damp out components of unwanted eigenvectors at each restart, the former method is faster in practice, so ARPACK++ provides the necessary computational framework for computing eigenvalues of L using our FC algorithm. All eigenvalues in the following numerical experiments are computed to a tolerance of 10^{-7} unless otherwise stated.

4. Numerical Results

4.1. Simple Geometries. To test the efficiency and accuracy of the eigenvalue solver as well as the domain decomposition approach we first present numerical results for two simple geometries, the unit square and the unit disk, using $L = \nabla^2$. For each of these geometries, we consider both Dirichlet and Neumann homogeneous boundary conditions. The exact eigenvalues and eigenfunctions are well known and easily derived. We report them here for reference:

(1) Unit square ($\Omega = [0,1] \times [0,1]$) with all sides fixed: $\lambda_{mn} = \pi^2(m^2 + n^2)$ for $m, n \in \mathbb{Z}^+$.

(2) Unit square with one side ($x = 0$) free: $\lambda_{mn} = \pi^2((m + 1/2)^2 + n^2)$ for $m \in \mathbb{Z}^+ \cup \{0\}$ and $n \in \mathbb{Z}^+$.

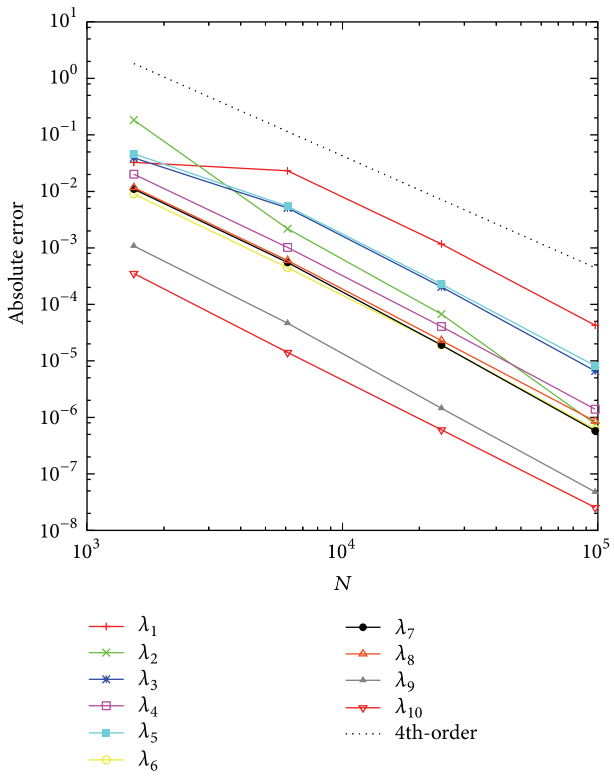
(3) Unit disk ($\Omega = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \le 1\}$) with fixed boundary: $\lambda_{mn} = j_{nm}^2$, where j_{nm} is the nth root of the mth Bessel function.

(4) Unit disk with free boundary: $\lambda_{mn} = {j'}^2_{nm}$, where j'_{nm} is the nth root of the derivative of the mth Bessel function.

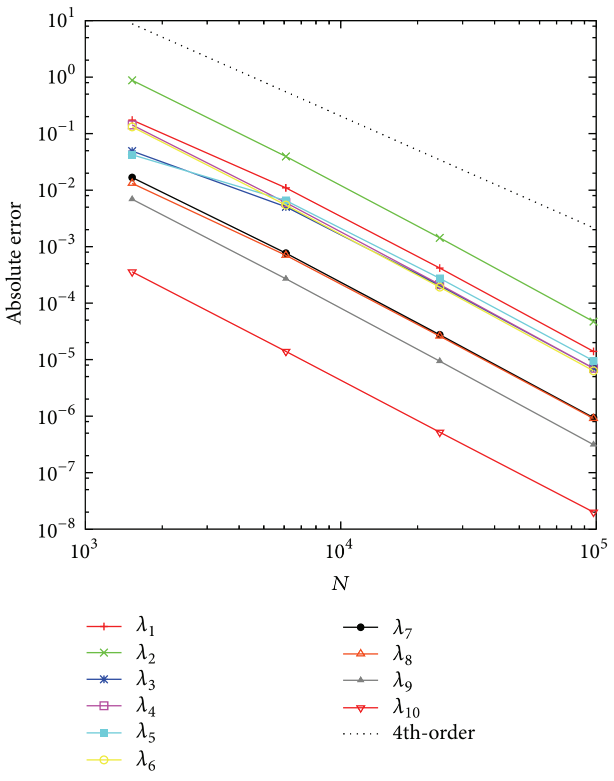
For these cases, the domain decompositions we use are not optimal (in the sense of minimizing the number of patches), as they are meant to test how well continuity conditions are enforced. In addition, we intentionally discretize neighboring patches with slightly different numbers of grid points (to prevent grid lines from overlapping and artificially increasing the accuracy of interpolation). The decompositions used for the square and disk cases are shown in Figure 2.

The error in the first 10 computed eigenvalues for all the cases and for increasingly finer discretizations is shown in Figure 3. In each case, we observe fourth-order convergence (e.g., doubling the number of points along one direction of each patch decreases the error in the eigenvalues by no less than 2^{-4}). In addition, for the finest discretization in each case, we obtain at least 8 digits of accuracy in the principal eigenvalue. Finally, the number of calls to the numerical operator (denoted by Lu) and the time taken per case are reported in Table 1.

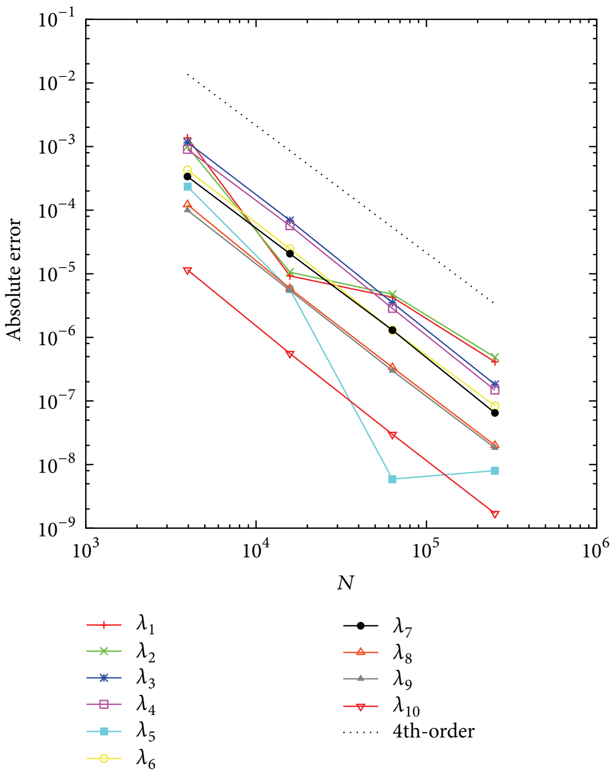
4.2. Comparison to Finite Differences. Because the Arnoldi iteration is a black box algorithm that depends only on the definition of the numerical operator, one might consider a very similar algorithm that replaces the FC(Gram) algorithm with another method for computing derivatives. In this



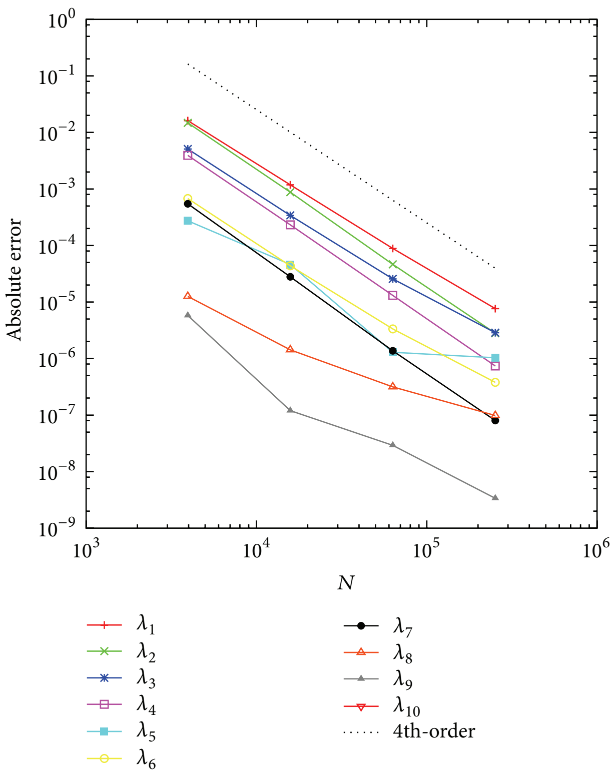
(a)



(b)



(c)



(d)

FIGURE 3: Convergence in the first ten eigenvalues for each of the four cases in Section 4.1: (a) unit square and fixed edges; (b) unit square and one free edge; (c) unit disk and fixed boundary; (d) unit disk and free boundary (with zero eigenvalues omitted). In all cases, the dashed line for fourth-order convergence is shown for comparison. N denotes the number of points used to discretize the overall two-dimensional geometry in each case.

Table 1: Computational time required for evaluation of the differential operator Lu for four test cases, with breakdown of computing time per Lu calculation, and the number of Lu calculations needed for the eigenvalue solver to converge to the required tolerance. All computations were performed on a 2.67 GHz Intel Xeon processor. (a) Square with fixed edges; (b) square with one free edge; (c) disk with fixed boundary; (d) disk with free boundary.

(a)

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
1525	0.001691	728	1.2313
6100	0.006096	2179	13.2835
24400	0.020961	6003	125.8296
97600	0.103420	22861	2364.2878

(b)

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
1525	0.001891	923	1.7452
6100	0.005459	2702	14.7507
24400	0.019813	8328	165.0070
97600	0.094591	23440	2217.2179

(c)

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
3950	0.006348	1065	6.7605
15800	0.017500	2037	35.6459
63200	0.072966	4981	363.4445
252800	0.257892	12338	3181.8739

(d)

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
3950	0.006638	1584	10.5144
15800	0.012105	3336	40.3828
63200	0.061148	7027	429.6900
252800	0.222666	14820	3299.9156

Table 2: Number of iterations and computational time required by the FC solver and the finite difference solver to compute λ_1 to a fixed accuracy for the unit square example (fixed boundaries). (The first 10 eigenvalues are calculated to the tolerance of 10^{-7}, but we only show the error in the first eigenvalue for conciseness.) For each given accuracy, the FC(Gram) algorithm outperforms the finite difference algorithm by a large margin. All computations were performed on a 2.67 GHz Intel Xeon processor. (a) Results using centered finite difference; (b) results using FC(Gram) with 5th-order basis.

(a)

Absolute error	N	Time per Lu operation (s)	Number of Lu operations
3.00×10^{-4}	74725	0.009501	9693
7.57×10^{-5}	298800	0.05130	37760

(b)

Absolute error	N	Time per Lu operation (s)	Number of Lu operations
3.53×10^{-4}	1525	0.001235	728
1.41×10^{-5}	6100	0.004211	2245

section, we use the examples of the previous section to show that the choice of FC(Gram) is justified when compared to a standard finite difference scheme and that the advantages of FC(Gram) are quite apparent. (It is worth noting the recent contribution in [20], which relies on finite differences and Richardson extrapolation, can provide highly accurate approximations of Laplace eigenvalues provided that the domain can be discretized by means of grids which exactly sample the boundary of the domain.)

In these tests, we compare the number of iterations, time taken, and number of grid points needed by the two methods to compute the principal eigenvalue of the Laplacian to a fixed amount of error. We consider both the unit disk and the unit square with Dirichlet boundary conditions, and a second-order accurate centered finite difference scheme is used in the comparison. As the results in Table 2 indicate, the FC(Gram) algorithm requires significantly fewer iterations and grid points than the finite difference algorithm to reach the desired accuracy. This also results in a much lower overall run time, even though the time per operation is smaller for the finite difference scheme. In addition, the FC(Gram) algorithm scales well to higher accuracies, while the cost of the finite difference algorithm grows prohibitively large for the finer discretizations. These results are even more pronounced in the case of the unit disk. Finally, simple tests using higher order Gram bases for the FC(Gram) algorithm demonstrate that the order of accuracy can be increased without much change in the computational cost. At the same time, there is no significant alteration to the FC(Gram) algorithm at high orders, while high-order finite difference schemes must be specially reformulated at boundaries to take into account the extended stencils. Thus, FC(Gram) seems to be well suited to the elliptic eigenvalue problems we consider here and competitive with other methods.

4.3. Smoothed L-Shaped Membrane. Collocation methods were used to calculate the eigenvalues of the L-shaped membrane with Dirichlet boundary conditions in order to show how polygonal boundaries, including those with geometric singularities, could be handled. Improvements to Fox et al.'s work [4] on the method of particular solutions have yielded the first several eigenvalues to at least 10 digits of accuracy [5]. In its present form the FC method does not handle geometries containing corner singularities, but it can be applied to approximations thereof containing sharp but smooth corners. (In the presence of corners in the boundary of the domain, PDE solutions and eigenfunctions are singular and therefore cannot be approximated with high accuracy by means of the present version of the FC method. An extension of this method could be envisioned which takes into account such singularities explicitly, but the development of such approaches lies beyond the scope of this paper.) We thus consider the eigenvalue problem for an L-shaped membrane with a smoothed corner, as discussed in what follows.

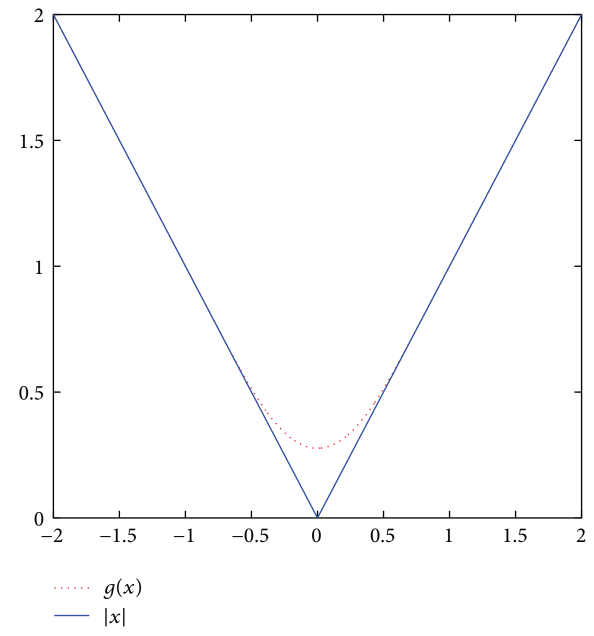


FIGURE 4: The smoothed corner coincides exactly with the absolute value function outside of $[-1, 1]$.

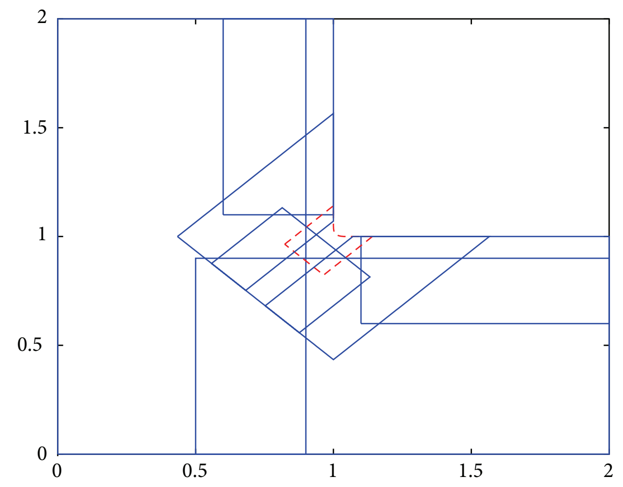


FIGURE 5: Domain decomposition for the smoothed L-shaped membrane. The corner patch ($d = 0.05$) is drawn using a dashed line for emphasis.

To decompose the membrane, we smooth the corner by locally approximating the boundary curve with C^∞ curve that is sampled at a fixed set of points. Evaluation at an arbitrary location is produced by means of Neville's algorithm for cubic interpolation on a sufficiently fine mesh to ensure that approximation errors are below the tolerances otherwise required. The smooth curve that replaces the corner is given (up to rigid transformations) by

$$g(x) = \int_{-\infty}^{x} \int_{-\infty}^{y} \phi(z)\, dz\, dy, \tag{9}$$

where the function $\phi(z)$ is a smooth bump function on $\mathbb{R}$ with support on $[-1, 1]$:

$$\phi(z) = \begin{cases} 0, & \text{if } z \notin [-1, 1], \\ \dfrac{e^{-1/(1-z)}}{e^{-1/(1-z)} + e^{-1/z}}, & \text{if } z \in [-1, 1]. \end{cases} \tag{10}$$

Thus, outside the interval $[-1, 1]$, $g(x) = |x|$ and matches the rest of the boundary of the L-shaped membrane near the corner. Within it, we have a smooth curve that deviates from the corner, as shown in Figure 4. (This can be thought of as a way of approximating the Dirac delta function, which is the second "derivative" of the absolute value function, with a smooth function that has a peak of finite height and nonzero width.)

Integral (9) is obtained numerically using Fourier techniques that take advantage of the fact that $\phi(z)$ is C^∞ periodic function on $[-1, 1]$. We sample ϕ at 1024 equally spaced points on $[-1, 1]$ and compute the FFT of the discrete data; at higher frequencies, the Fourier coefficients are effectively zero. Integration in the frequency domain amounts to multiplying each coefficient by a constant. Thus, taking the inverse FFT after multiplication yields the integral of ϕ on $[-1, 1]$ up to a constant of integration. The second integral is computed the same way, with the constant term integrated analytically. This yields a highly accurate representation of $g(x)$ and its derivatives, which are necessary for the calculation of the Laplacian. Note that this method can be generalized for smoothing reentrant corners of any angle by choosing the first integration constant or adjusting the height of the peak in ϕ accordingly.

Using this description of the approximate boundary curve we parametrize the region of the L-shaped membrane near the corner by

$$\begin{aligned} \tilde{x}(s,t) &= (2+2l)\, s - (l+1), \\ \tilde{y}(s,t) &= \frac{t}{b} g\left((2+2l)\, s - (l+1)\right), \\ \begin{pmatrix} x \\ y \end{pmatrix} &= d \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \\ &\quad + h \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix}, \end{aligned} \tag{11}$$

for $s \in [0,1]$ and $t \in [0,b]$. Here $\theta = -\pi/4$, $(x_0, y_0) = (1,1)$, and l and h are chosen to be of the same order as d. The decomposition of the smoothed L-shaped membrane is shown in Figure 5. This particular parametrization is chosen so that as $d \to 0$ the curve approximates the corner of the L-shaped membrane arbitrarily well.

Figure 7 indicates that 4th-order convergence is achieved, just as in the tests for the simpler unit square and unit disk geometries; run time information is given in Table 3. Of equal importance is the fact that the computed eigenfunctions (Figure 6) closely resemble those of the true L-shaped membrane [2] and the corresponding eigenvalues agree to anywhere from two to four digits, as one might expect. In practice, we have found that the eigenvalues converge linearly as the rounding radius of curvature tends to zero.

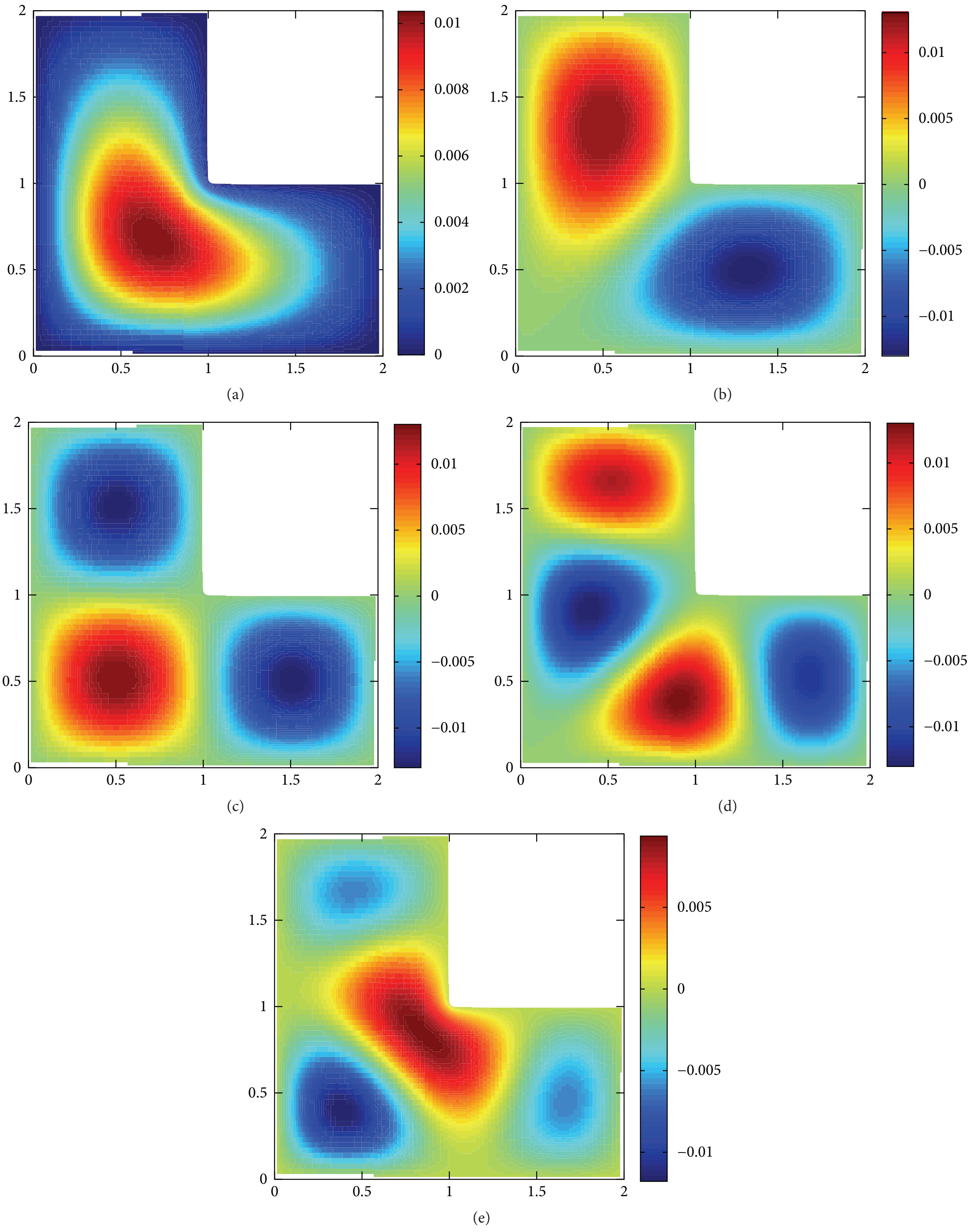


Figure 6: The first 5 eigenfunctions (modes of vibration) of the smoothed L-shaped membrane (cf. Figure 1 in [2]): (a) $\lambda_1 = -9.6013$; (b) $\lambda_2 = -15.1969$; (c) $\lambda_3 = -19.7392$; (d) $\lambda_4 = -29.5214$; (e) $\lambda_5 = -31.8179$.

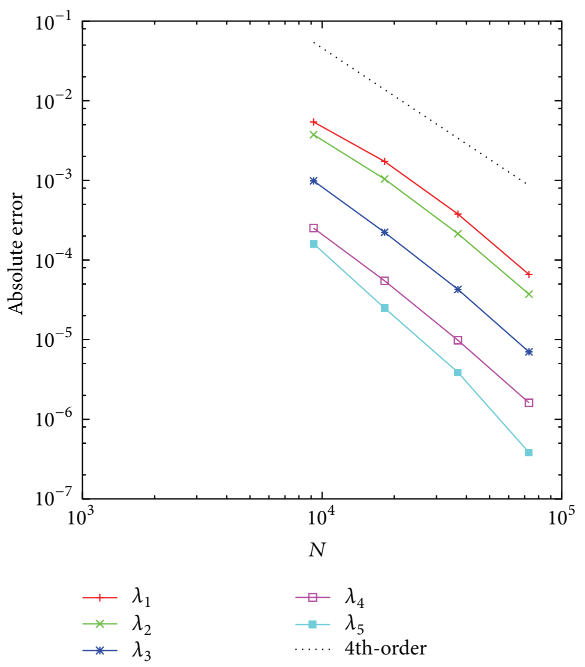


FIGURE 7: Convergence in the first 5 eigenvalues for the smoothed L-shaped membrane ($d = 0.05$) is roughly 4th order when compared to the finest discretization tested ($N = 147456$). Note that the eigenvalues do not necessarily converge to those of the true L-shaped membrane with a sharp corner due to the approximation of the corner geometry. For example, here $\lambda_1 = 9.6013$ when $N = 147456$, even though $\lambda_1 = 9.63972$ (e.g., [2, 4]) for the membrane with the sharp corner.

TABLE 3: Computational time required for evaluation of the Laplacian for the smoothed L-shaped membrane. All computations were performed on a 2.67 GHz Intel Xeon processor.

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
9216	0.012912	6820	88.0598
18225	0.018062	14736	266.1676
36864	0.031643	21469	679.3593
72990	0.049035	44212	2167.9487
147456	0.149842	88070	13196.5794

TABLE 4: Computational times and number of iterations required for evaluation of the Laplacian on the thinly tethered membrane. All computations were performed on a 2.67 GHz Intel Xeon processor to a tolerance of 10^{-4}.

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
56320	0.061338	464727	28505.4983
79420	0.072279	1758160	127079.4402
116380	0.115435	2679346	309291.7615
166045	0.167378	2793136	467510.6446

4.4. Thinly Tethered Membrane. In this section we explore a more physically relevant membrane geometry inspired by quantum optics experiments. The particular geometry explored here can be described as a roughly square-shaped membrane (with side length on the order of 100 μm) attached by thin tethers coming off of the four corners to a square frame (which has side length on the order of 600 μm). The edges of the membrane that are not attached to the frame are left free. In addition, the tethers smoothly continue into the boundaries of the inner square and also taper smoothly at the corners of the frame. (The purpose of this geometry is to decrease stress on the main part of the membrane, which would theoretically drive down the fundamental frequency.) For this experiment, we take the stress across the membrane to be constant in order to obtain a first approximation to the mechanics of such a complex membrane. Under this approximation, the operator in (2) is once again the Laplace operator (up to a constant factor).

The decomposition developed for this membrane is shown in Figure 8. Nearly all of the patches used are rectangles, but the smooth boundary curves at the corner of the membrane and the frame provide a challenge. To address this issue, we use the methodology previously developed for smoothing corners to parametrize these regions; we treat the segments of the boundary shown in Figures 8(b) and 8(c) as having smoothed reentrant corners with angles of 90° and 135°, respectively. The dimensions of the membrane are scaled so that the frame is the square of side length 2 centered at the origin.

As in all the previous cases, we consider several successively finer discretizations for the geometry. Because this problem is significantly larger than those considered previously in this paper, we only increase the number of points in each direction by a factor of 1.2 at each level. Convergence rates for the first 5 eigenvalues are shown in Figure 9 and timings are given in Table 4. In addition, plots of the

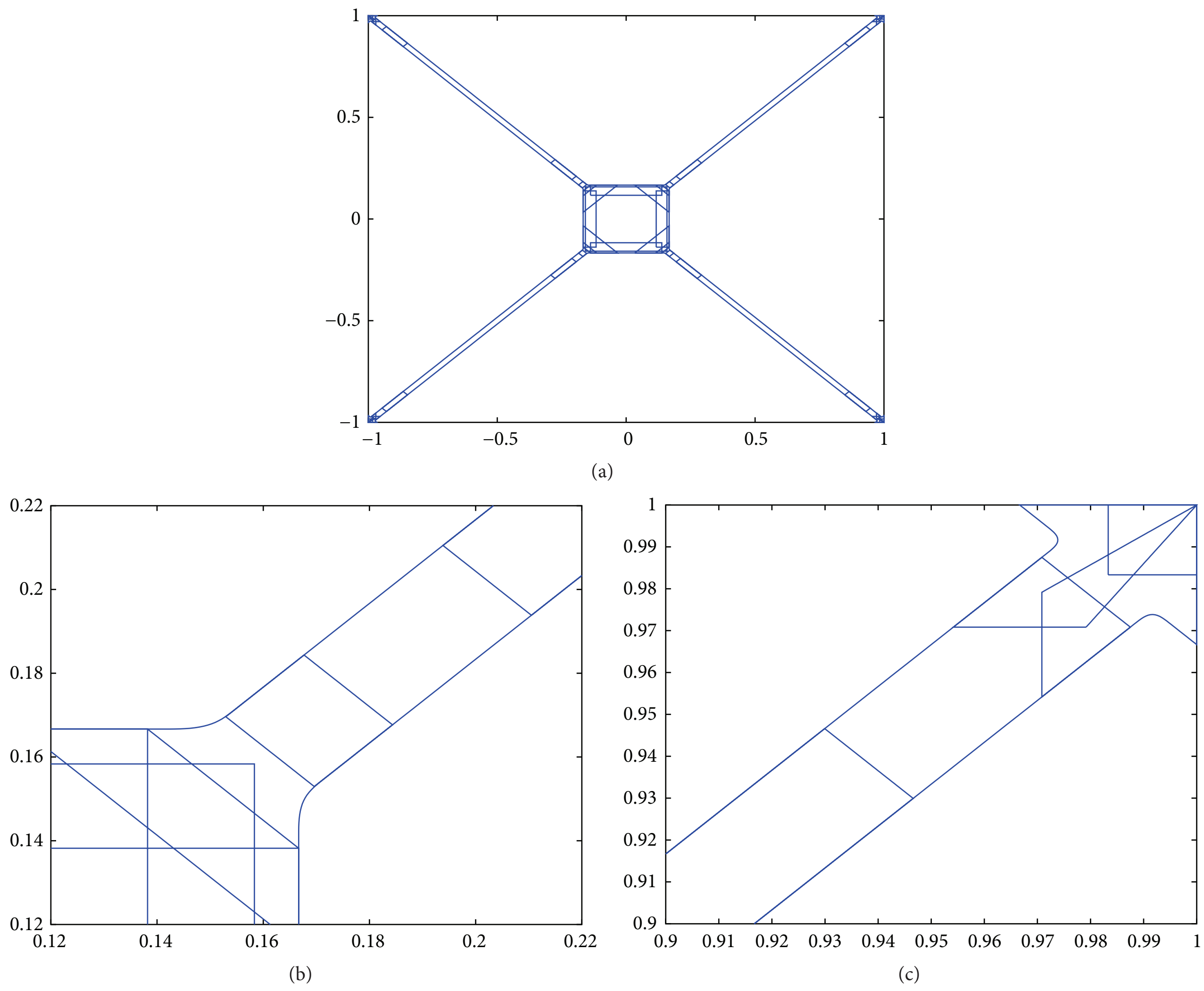


Figure 8: Domain decomposition for the thinly tethered membrane: (a) full decomposition (37 patches); (b) curving boundary near point where tether attaches to center square; (c) curving boundary near corner of frame.

eigenfunctions corresponding to these computed eigenvalues are shown in Figure 10, which show interesting oscillation modes.

4.5. Variable Coefficients. In this section we consider a more general eigenvalue problem with nonconstant coefficients, which generalizes the eigenvalue problem for the Laplacian:

$$\begin{gathered}\nabla\cdot(\beta(x,y)\nabla u)=\lambda u,\\ u|_{\partial\Omega_1}=0,\\ \left.\frac{\partial u}{\partial\widehat{\mathbf{n}}}\right|_{\partial\Omega_2}=0.\end{gathered}\tag{12}$$

We take β to be a scalar function of position. While the Laplacian eigenvalue problem allows us to study the fundamental modes of vibration for a homogeneous membrane, these more general operators allow for the simulation of membranes with spatially varying tension, for example, which are of great physical relevance.

For our tests, we solve (12) for the unit disk with fixed boundary and $\beta(x,y)=x^2+y^2+1$, using the same decomposition and set of discretizations as used previously. The absolute error is computed with respect to finest discretization. Results here, shown in Figure 11 and Table 5, are similar to those seen in the case of the Laplacian eigenvalue problem on the unit disk. We observe, most notably, that the convergence rate for the computed eigenvalues (with respect to the discretization) is roughly fourth order again. Similarly, the time spent computing a single realization of the operator and the number of calls to the operator scale sublinearly with respect to N, just as in the constant coefficient case. However, the time per Lu calculation is slightly larger here since the gradient and divergence operators are computed in succession.

4.6. Parallelization. The differential operator L can be evaluated independently on each patch P_i, so the algorithm presented here can be parallelized by distributing this step to multiple processors. This is because the only step that requires

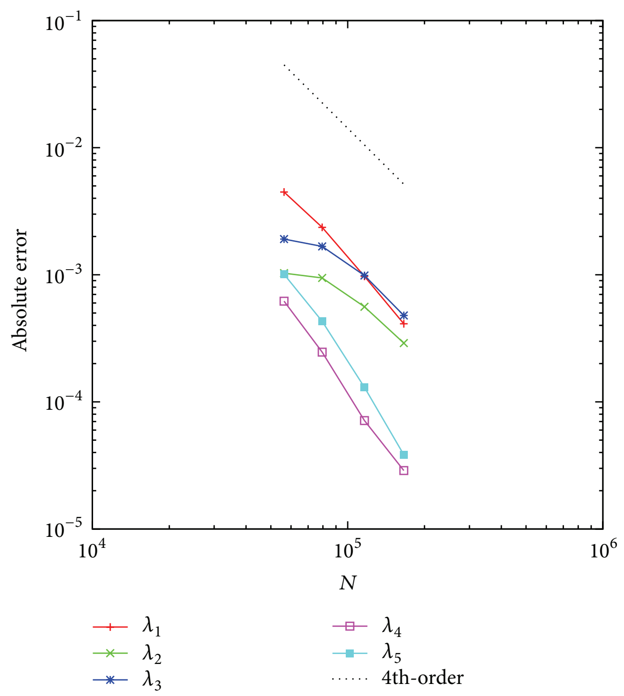


FIGURE 9: Convergence in the first 5 eigenvalues for the thinly tethered membrane is roughly 4th order (when compared to the finest discretization tested; $N = 239580$).

TABLE 5: Computational times and number of iterations required for evaluation of the variable coefficient elliptic operator (12) with $\beta(x, y) = x^2 + y^2 + 1$. All computations were performed on a 2.67 GHz Intel Xeon processor.

N	Time per Lu operation (s)	Number of Lu operations	Total time in Lu (s)
3950	0.009051	1242	11.2415
15800	0.023501	3072	72.1960
63200	0.109634	7610	834.3157
252800	0.370401	20630	7641.3635

communication between different patches is the setting of continuity conditions (via interpolation from the interiors of overlapping patches). To implement parallelization across p processors, we use a simple load balancing algorithm for $n \geq p$ (cf. Algorithm 2a in [21]) that assigns a given task to the processor with the least amount of work. The task in this case is computing the restriction of Lu to P_i, and the amount of work for this task is given by the total number of points in P_i.

To test the efficiency of the parallelization, we consider again the thinly tethered membrane geometry shown in Figure 8 ($n = 37$), using a discretization with $N = 3604480$. The parallel efficiency, defined by

$$E_p = \frac{T_1}{pT_p}, \tag{13}$$

where T_1 is the time taken to apply Lu using a single processor and T_p is the time taken to apply Lu after distributing the tasks to p processors, is shown in Figure 12 with respect to increasing p. Sharp jumps in the efficiency are most likely because of the prime number of patches, which makes distribution of tasks in as even a manner as possible somewhat difficult under this framework. Our parallel efficiency tests included runs up to twelve processors; the parallel efficiency decreased slowly as the number of processors increased to a level around 80%.

5. Conclusions

We have presented a high-order method for evaluation of the eigenvalues of elliptic operators. Our approach combines the FC method for evaluation of accurate Fourier approximations of nonperiodic functions, domain decomposition, and Arnoldi iteration. Our domain decomposition strategy enables consideration of general geometries. Additionally, the decomposition strategy leads easily to efficient parallel computations. The resulting eigenvalue algorithm converges at a roughly fourth-order rate in all cases considered. Furthermore, a methodology to deal with corner singularities is suggested by the results from the smoothed L-shaped membrane. Comparisons to solutions provided by a finite difference scheme demonstrate the advantages provided by the proposed algorithm as well as the viability of our Fourier continuation algorithm for evaluation of eigenvalues for complex geometries and general variable-coefficient operators.

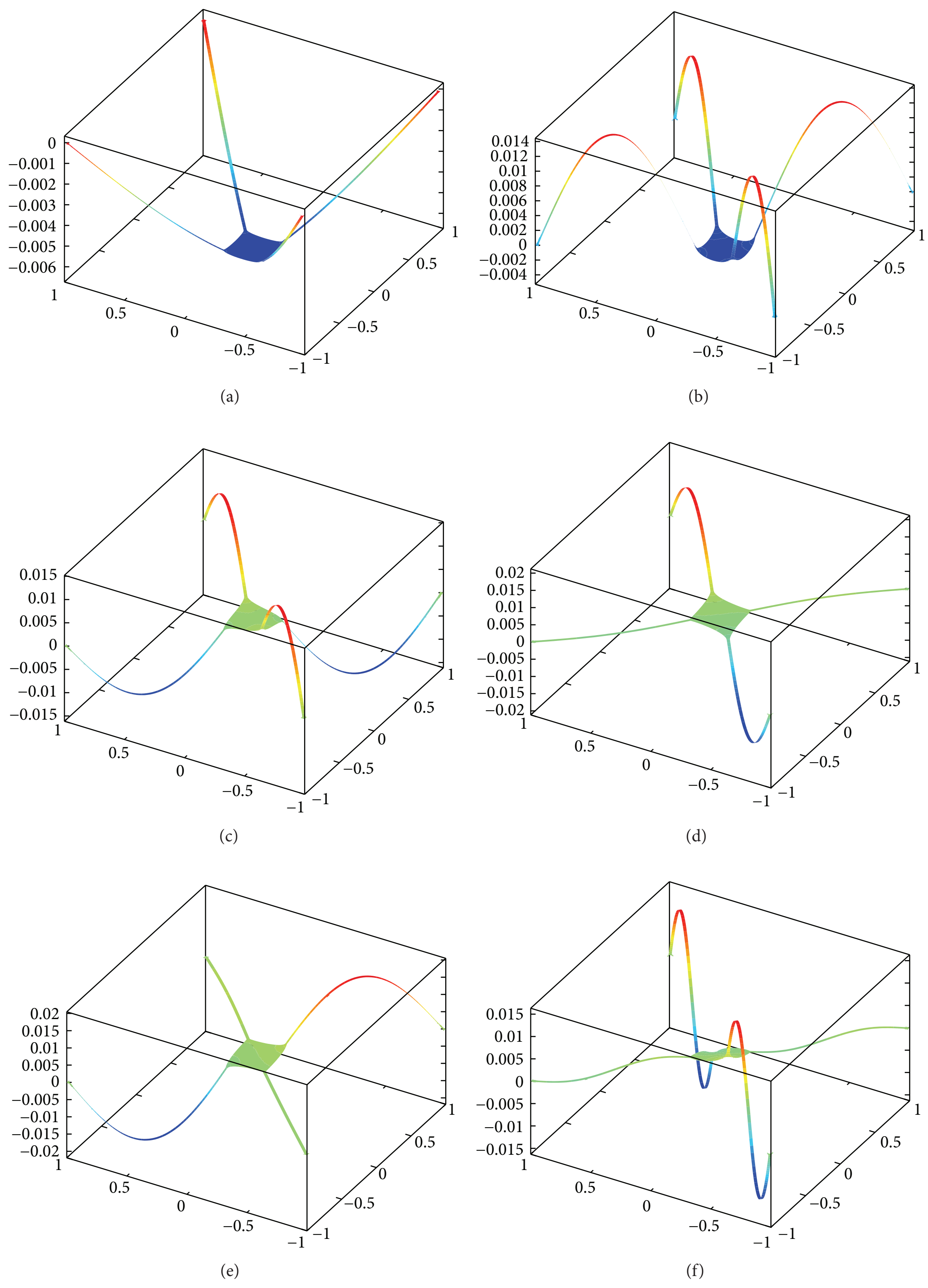


FIGURE 10: The first 6 eigenfunctions (modes of vibration) of the thinly tethered membrane: (a) u_1; (b) u_2; (c) u_3; (d) u_4; (e) u_5; (f) u_6.

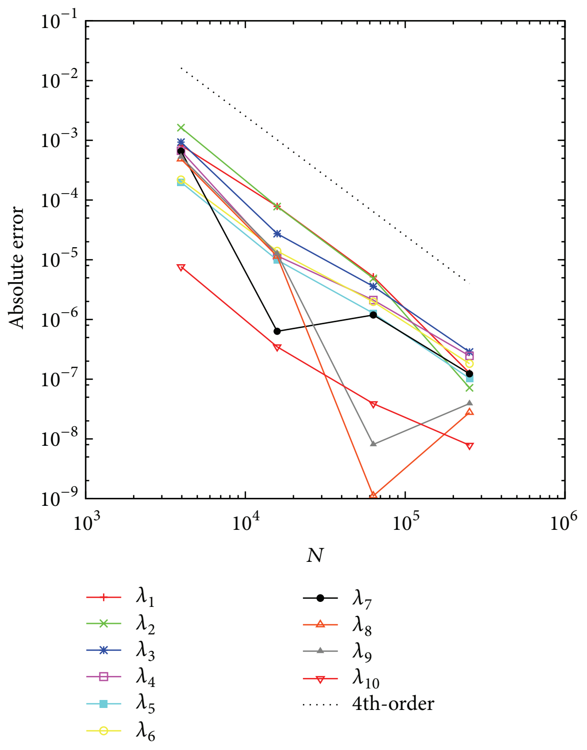


FIGURE 11: Convergence in the first 10 eigenvalues for the unit disk with the variable coefficient $\beta(x, y) = x^2 + y^2 + 1$ is shown with respect to the finest discretization; $N = 252800$.

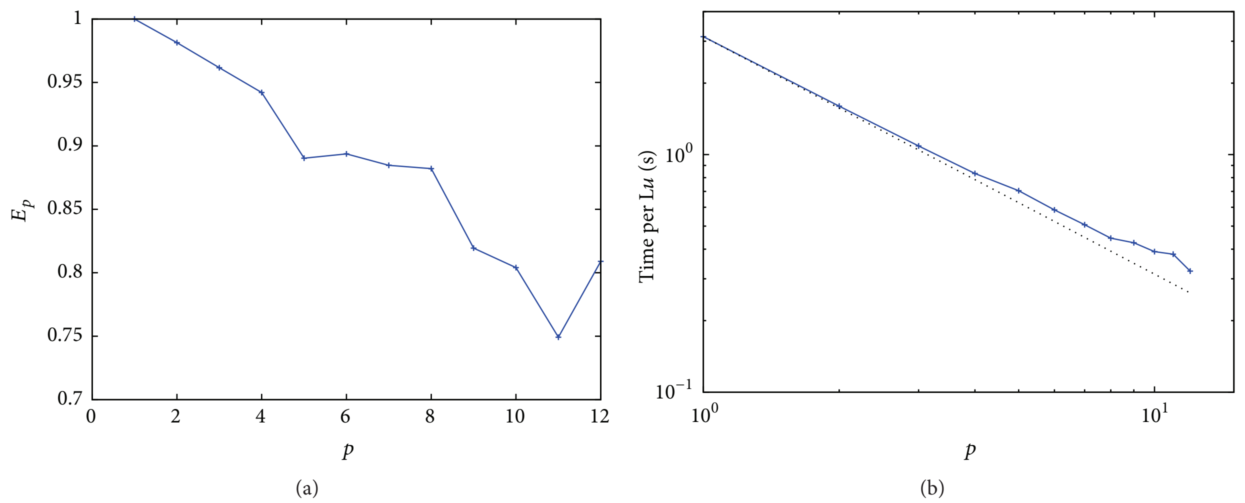


FIGURE 12: Parallel efficiency tests performed on 12 Intel Xeon 2.66 GHz processor computer: (a) parallel efficiency (E_p) decreases with the number of processors to approximately 80%; (b) time per call to Lu is shown versus the increasing number of processors. The dashed line denotes perfect efficiency (i.e., $E_p = 1$, or $pT_p = T_1$).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge support from NSF and AFOSR.

References

[1] O. P. Bruno and M. Lyon, “High-order unconditionally stable FC-AD solvers for general smooth domains. I. Basic elements,” *Journal of Computational Physics*, vol. 229, no. 6, pp. 2009–2033, 2010.

[2] L. N. Trefethen and T. Betcke, “Computed eigenmodes of planar regions,” in *Recent Advances in Differential Equations and Mathematical Physics*, vol. 412 of *Contemporary Mathematics*,

pp. 297–314, American Mathematical Society, Providence, RI, USA, 2006.

[3] F. Scheben and I. G. Graham, “Iterative methods for neutron transport eigenvalue problems,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2785–2804, 2011.

[4] L. Fox, P. Henrici, and C. Moler, “Approximations and bounds for eigenvalues of elliptic operators,” *SIAM Journal on Numerical Analysis*, vol. 4, no. 1, pp. 89–102, 1967.

[5] T. Betcke and L. N. Trefethen, “Reviving the method of particular solutions,” *SIAM Review*, vol. 47, no. 3, pp. 469–491, 2005.

[6] P. Amore, “Solving the Helmholtz equation for membranes of arbitrary shape: numerical results,” *Journal of Physics A: Mathematical and Theoretical*, vol. 41, no. 26, Article ID 265206, 2008.

[7] G. Sathej and R. Adhikari, “The eigenspectra of Indian musical drums,” *Journal of the Acoustical Society of America*, vol. 125, no. 2, pp. 831–838, 2009.

[8] L. Banjai, “Eigenfrequencies of fractal drums,” *Journal of Computational and Applied Mathematics*, vol. 198, no. 1, pp. 1–18, 2007.

[9] R. H. Hoppe, H. Wu, and Z. Zhang, “Adaptive finite element methods for the Laplace eigenvalue problem,” *Journal of Numerical Mathematics*, vol. 18, no. 4, pp. 281–302, 2010.

[10] M. S. Min and D. Gottlieb, “Domain decomposition spectral approximations for an eigenvalue problem with a piecewise constant coefficient,” *SIAM Journal on Numerical Analysis*, vol. 43, no. 2, pp. 502–520, 2005.

[11] R. Rannacher, A. Westenberger, and W. Wollner, “Adaptive finite element solution of eigenvalue problems: balancing of discretization and iteration error,” *Journal of Numerical Mathematics*, vol. 18, no. 4, pp. 303–327, 2010.

[12] T. A. Driscoll, “Eigenmodes of isospectral drums,” *SIAM Review*, vol. 39, no. 1, pp. 1–17, 1997.

[13] D. Boffi, “Finite element approximation of eigenvalue problems,” *Acta Numerica*, vol. 19, pp. 1–120, 2010.

[14] J. Berland, C. Bogey, O. Marsden, and C. Bailly, “High-order, low dispersive and low dissipative explicit schemes for multiple-scale and boundary problems,” *Journal of Computational Physics*, vol. 224, no. 2, pp. 637–662, 2007.

[15] W. D. Henshaw and D. W. Schwendeman, “Parallel computation of three-dimensional flows using overlapping grids with adaptive mesh refinement,” *Journal of Computational Physics*, vol. 227, no. 16, pp. 7469–7502, 2008.

[16] N. Albin and O. P. Bruno, “A spectral FC solver for the compressible Navier-Stokes equations in general domains I: explicit time-stepping,” *Journal of Computational Physics*, vol. 230, no. 16, pp. 6248–6270, 2011.

[17] M. Frigo and S. G. Johnson, “The design and implementation of FFTW3,” *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[18] M. Lyon and O. P. Bruno, “High-order unconditionally stable FC-AD solvers for general smooth domains II. Elliptic, parabolic and hyperbolic PDEs; theoretical considerations,” *Journal of Computational Physics*, vol. 229, no. 9, pp. 3358–3381, 2010.

[19] F. M. Gomes and D. C. Sorenson, “*ARPACK++: A C++* implementation of *ARPACK* eigenvalue package,” Tech. Rep. CRPC-TR97729, Center for Research on Parallel Computation, 1997.

[20] P. Amore, J. P. Boyd, F. M. Fernandez, and B. Rösler, “High order eigenvalues for the Helmholtz equation in complicated non-tensor domains through Richardson Extrapolation of second order finitedifferences,” http://arxiv.org/abs/1509.02795.

[21] R. Blikberg and T. Sørevik, “Load balancing and OpenMP implementation of nested parallelism,” *Parallel Computing*, vol. 31, no. 10-12, pp. 984–998, 2005.