A fast adaptive numerical solver for nonseparable elliptic partial differential equations

June-Yub Lee

Dept of Math, Ewha Womans Univ,
Seoul 120-750, KOREA
jylee@math.ewha.ac.kr

Abstract

We describe a fast numerical method for non-separable elliptic equations in self-adjoin form on irregular adaptive domains. One of the most successful results in numerical PDE is developing rapid elliptic solvers for separable EPDEs, for example, Fourier transformation methods for Poisson problem on a square, however, it is known that there is no rapid elliptic solvers capable of solving a general non-separable problems [8]. It is the purpose of this paper to present an iterative solver for linear EPDEs in self-adjoint form. The scheme discussed in this paper solves a given non-separable equation using a sequence of solutions of Poisson equations, therefore, the most important key for such a method is having a good Poison solver. High performance is achieved by using a fast high-order adaptive Poisson solver which requires only about 500 floating point operations per gridpoint in order to obtain machine precision for both the computed solution and its partial derivatives. A few numerical examples have been presented.

1 Introduction

Many problems in science and engineering require the solutions of second order linear elliptic partial differential equations (EPDEs). Thus, developing numerical solvers for these problems has been one of the central areas of numerical analysis. In sixties, the early era of numerical PDE, Finite Difference Method (FDM) had been taken an important role in this kind of effort in general. At the same time, many fast algorithms had been studied to get the best computational performance for certain class of problems, for example, Poisson equations on rectangle-like domain [9, 25, 26, 24]. The resulting

[1991] *Mathematics Subject Classification by AMS :* 65N, 65R20
Key words and phrases : Fast multipole method, Fast algorithm, Rapid elliptic solver, Helmholtz decomposition, Layer potential

performance was outstanding and fast numerical methods have been still studied since then, however, such tools are limited, in many cases, to the separable problems with constant coefficients on regular grids. Flexibility in discretizing a problem and handling a complex geometry becomes more and more important for practical applications and it is one of the main reasons why Finite Element Method (FEM) and its offsprings such as Domain Decomposition (DD) or Multi-Grid (MG) methods become more and more popular these days [1, 3, 6].

In this paper, we describe a fast numerical method for non-separable elliptic equations in self-adjoin form on irregular adaptive domains and provide some computational examples. The basic idea of the numerical method is that non-separable equations could be iteratively solved using the solutions of simpler separable constant coefficient equations. Such ideas have been proposed by many researchers [2, 4, 7, 8, 21] in eighties and nineties. In order to obtain good performance for this iterative method, a fast elliptic solver for separable equations is a key component and the ratio of computational cost for the target problem to that for the separable problem should be bigger that the number of iteration.

In section 2, we describe an iterative scheme to solve non-separable EPDE in self-adjoin form. The iterative scheme used in this paper has been already proposed by Brackbill and Forslund [4] and is one of many possible choices to develop a fast algorithm. The most salient feature of our implementation is that we use a new fast high-order adaptive Poisson solver proposed by Greengard and Lee [11, 17]. The solver described in section 3 requires only about 500 floating point operations per grid-point which is only a few time more expensive than a second order nonadaptive method based on Fourier method or cyclic reduction, yet it can easily obtain machine precision for both the computed solution and its partial derivatives. In section 4, we present a few numerical examples to demonstrate the usefulness of the Poisson solver for more general elliptic equations.

2 An iterative numerical procedure

In this section, we describe an iterative procedure, to solve a self-adjoint nonseparable generalization of Poisson's equation

$$\frac{\partial}{\partial x}\left[A(x,y)\frac{\partial}{\partial x}u(x,y)\right]+\frac{\partial}{\partial y}\left[B(x,y)\frac{\partial}{\partial y}u(x,y)\right]=S(x,y), \tag{1}$$

where A and B are nonzero functions with the same sign in a two dimensional domain Ω. We assume that S satisfy, if necessary, certain compatibility conditions to solve the equation (1), for example, the integral of S over the domain matches total flux in case of Poisson Neumann problem. The procedure which will be described below in details has been proposed in [2, 4] and a successful implementation of the algorithm depends on fast and accurate Poisson solver for the given computational domain.

The vector field generated by the solution of (1) can be split into the curl-free part $\nabla\phi$ of (Au_x, Bu_y) and the divergence-free part $\nabla \times \psi$ of (u_x, u_y) as follows:

$$\begin{aligned} Au_x &= \phi_x + A\psi_y \quad (2)\\ Bu_y &= \phi_y - B\psi_x. \quad (3)\end{aligned}$$

Equation (1) can be rewritten, by substitution, as a system of two Poisson equations

$$\begin{aligned} \Delta\phi &= S(x,y) - (A\psi_y)_x + (B\psi_x)_y \quad (4)\\ \Delta\psi &= \left(\frac{1}{B}\phi_y\right)_x - \left(\frac{1}{A}\phi_x\right)_y \quad (5)\end{aligned}$$

with the compatible boundary conditions

$$\begin{aligned} (d_x, d_y)\cdot\nabla\phi &= (d_x, d_y)\cdot(Au_x, Bu_y) \quad (6)\\ (d_x, d_y)\cdot\nabla\psi &= (0,0) \quad (7)\end{aligned}$$

where (d_x, d_y) denotes normal (or tangential) vector to the boundary when Neumann (or Dirichlet) boundary condition is given. (This kind of boundary type allows solutions with constant differences and can be uniquely determined with an additional boundary data.)

The numerical solution of the system (4),(5) can be computed iteratively as follows:

$$\begin{aligned} \Delta\phi^{k+1} &= S(x,y) - \left(A\psi_y^k\right)_x + \left(B\psi_x^k\right)_y \quad (8)\\ \Delta\psi^{k+1} &= \left(\frac{1}{B}\phi_y^{k+1}\right)_x - \left(\frac{1}{A}\phi_x^{k+1}\right)_y \quad (9)\end{aligned}$$

with initial guess $\phi^0 = 0, \psi^0 = 0$ for $k = 0$. At each step in the iteration, new estimate for u is given by

$$\left(u_x^k, u_y^k\right) = \left(\frac{1}{A}\phi_x^k + \psi_y^k, \frac{1}{B}\phi_y^k - \psi_x^k\right). \quad (10)$$

And this process is continued until some desired convergence are achieved, for example, the relative iteration convergence error is smaller than given tolerance ϵ.

$$||\nabla u^k - \nabla u^{k-1}||_\Omega^2 < \epsilon^2 ||\nabla u^k||_\Omega^2 \quad (11)$$

where $||\cdot||_\Omega$ denotes for L^2 norm in the domain Ω.

3 A fast Poisson solver using FMM

In this section, we briefly describe a direct, adaptive numerical method for the Poisson equation in order to solve (8) and (9) numerically. The solutions of Poisson problems,

in general, have to satisfy not only the Poisson equation in the domain $\Omega \subset R^2$ but also some linear boundary conditions on $\partial\Omega$. Such equations can be solved easily using a decomposition method, widely used for constant coefficient linear differential equations, which first determines the solution of the equation in the absence of physical boundaries and then solves an auxiliary equation later to enforce the boundary condition.

Therefore, we will concentrate our attention, in the first part of this section, only to the solution of the Poisson equation in the absence of physical boundaries for the source distribution f whose support is bounded in unit square D in R^2,

$$\Delta u_v = f \quad \text{in} \quad R^2. \tag{12}$$

And then we will later describe the auxiliary Laplace equation to match the boundary conditions for $u = u_v + u_h$,

$$\Delta u_h = 0 \ \text{ in } \ \Omega\,, \qquad \begin{array}{rcl} u_h & = & g - u_v \\ \frac{\partial u_h}{\partial n} & = & h - \frac{\partial u_v}{\partial n} \end{array} \quad \text{on } \ \partial\Omega \tag{13}$$

where g or h is the Dirichlet or the Neumann data on the boundary and $\frac{\partial}{\partial n}$ denotes out normal derivative on the boundary $\partial\Omega$. And we will also explain how to numerically solve the equations efficiently using a boundary integral method.

3.1 A volume integral method for the Poisson problem

There are many approaches to solve the Poisson problem (12). A few special techniques exist especially for this problem under special circumstances [9], for examples, spectral methods using fast Fourier transformation or cyclic reduction methods on rectangular-like or tensor-product meshes. However, its usage is very restricted for the problems with more complex discretizations. Many of finite difference and finite element methods are well studied and they take advantage of the fact that the Laplacian Δ is operator local which makes it possible to develop efficient numerical methods.

Unfortunately, these standard methods are not completely robust when the source distribution has a complex structure, the grid is highly non-uniform, and high accuracy of its derivatives is required. And a great deal of researches are still going on in this field. In order to overcome these difficulties, the Poisson problems are solved using a rather new approach which gives the exact solution u in the form of a volume integral

$$u(\mathbf{x}) = \frac{1}{2\pi} \int_{\mathbf{R}^2} \log|\mathbf{x} - \mathbf{y}| f(\mathbf{y})\, d\mathbf{y}. \tag{14}$$

There are many advantages to this approach and readers interested in complete discussion of the algorithm are referred to the paper by Greengard and Lee [11].

We now briefly outline mathematical results to this volume integral approach. Assume that the source distribution f is supported inside a square domain D embedded in a quad-tree structure with M leaf nodes D_i and f is smooth on the scale of each such small square D_i. The main result can be summarized in the following theorem :

Theorem 3.1 *Let the source distribution f be given as a K-th order Chebyshev polynomial f_i for each leaf node D_i for $i = 1, \ldots, M$ of the quad tree embedded on D. Then, for $\mathbf{x} \in D_i$, the solution to the Poisson equation (12) is given by*

$$u(\mathbf{x}) = u_i^s(\mathbf{x}) + \sum_{j=1}^{M} u_j^h(\mathbf{x}). \tag{15}$$

where $u_i^s(\mathbf{x})$ is a polynomial satisfying $\Delta u_i^s = f_i$ locally (inside D_i) and $u_j^h(\mathbf{x})$ is a harmonic function in D_i defined in terms of single and double layer potentials generated by the boundary values of $u_j^s(\mathbf{x})$ and $\frac{\partial}{\partial n} u_j^s(\mathbf{x})$ along the interfaces of subdomain D_j.

While $u_i^s(\mathbf{x})$ depends only on f_i and the computation cost linearly depends on number of boxes M, the evaluation of the harmonic patches by direct summation over M boxes requires order $O(MN)$ work for N target points, which is very expensive. The algorithm described below uses the Fast Multipole Method (FMM) to reduce the computation cost to order $O(N)$. We briefly summarize the implementation of the theorem which consists of four steps.

Suppose a square box D contains the support of the right hand side f. Starting from $S_{0,0} = D$, a quad-tree structure is obtained by dividing a square subdomain $S_{l,k}$ into four equal size subdomains $S_{l+1,4k+d}$, for $d = 0, 1, 2, 3$. In order to achieve adaptivity, this process continues until the source term f is locally smooth enough on each of the leaf nodes $S_{l,k}$ aliased as D_i. To describe the algorithm, we define two concepts: the neighbors and the interaction list for each square $S_{l,k}$. The neighbors $\mathcal{N}_{l,k}$ consist of those squares at the same (or coarser, if none) refinement level with which it shares a boundary point and the interaction list $\mathcal{I}_{l,k}$ consists of those squares at the same (or coarser, if none) refinement level in the area covered by the neighbors of $S_{l,k}$'s parent, excluding the neighbors of $S_{l,k}$.

1. **First local solve:** Given any 2-dimensional Chebyshev polynomial $f_j(\mathbf{x})$, find a polynomial $u_j^s(\mathbf{x})$ such that $\Delta u_j^s(\mathbf{x}) = f_j(\mathbf{x})$ and then compute a multipole expansion $\Phi_j(\mathbf{x})$ at the center $\mathbf{y}_j$ representing the harmonic patch $u_j^h(\mathbf{x})$ for each of leaf node boxes D_j, $j = 1, \cdots, M$.

$$\Phi_j(\mathbf{x}) = Re\left(\frac{1}{2\pi i} \int_{\partial D_j} \frac{\partial u_j^s}{\partial n}(w) \log(w - \mathbf{x})\, dl_w + \frac{1}{2\pi i} \int_{\partial D_j} \frac{u_j^s(w)}{w - \mathbf{x}}\, dw \right) \tag{16}$$

2. **FMM upward pass:** Once the multipole expansion Φ_j for each leaf node is obtained, multipole expansions $\Phi_{l,k}$ for internal nodes $S_{l,k}$ can be computed by collecting the information from their four children $S_{l+1,4k+d}$, $d = 0, 1, 2, 3$. The multipole for $S_{l,k}$ represents $\sum u_j^h(\mathbf{x})$ of all D_j inside of $S_{l,k}$ for $\mathbf{x} \notin \mathcal{N}_{l,k}$.

$$\Phi_{l,k}(\mathbf{x}) := \sum_{D_j \subset S_{l,k}} u_j^h(\mathbf{x}) = \sum_{d=0}^{3} \Phi_{l+1,4k+d}(\mathbf{x}) \text{ for } \mathbf{x} \in \mathcal{N}_{l,k}^c. \tag{17}$$

3. FMM downward pass: The local expansion $\Psi_{l,k}(\mathbf{x})$ at $\mathbf{x} \in S_{l,k}$ is a combination of its parent's $\Psi_{l-1,floor(k/4)}(\mathbf{x})$ and $\Phi_j(\mathbf{x})$ for all $D_j \in \mathcal{I}_{l,k}$.

$$\Psi_{l,k}(\mathbf{x}) := \sum_{D_j \subset \mathcal{N}^c_{l,k}} u^h_j(\mathbf{x}) = \Psi_{l-1,floor(k/4)}(\mathbf{x}) + \sum_{S_{l,k} \subset \mathcal{I}_{l,k}} \Phi_{l,k}(\mathbf{x}) \text{ for } \mathbf{x} \in S_{l,k} \quad (18)$$

since $\mathcal{N}^c_{l,k} = \mathcal{N}^c_{l-1,floor(k/4)} \cup \mathcal{I}_{l,k}$. The local expansion $\Psi_{0,0}(\mathbf{x})$ for the root node is zero by definition since $\mathcal{N}^c_{0,0}$ is empty and the hierarchical quad-tree data structure allows a recursive procedure of the summation from top to bottom.

4. Final local solve: Once the $\Psi_i(\mathbf{x})$ for all leaf nodes D_i are computed, the harmonic patches $\sum_{j=1}^M u^h_j(\mathbf{x})$ of the leaf node D_i is the sum of $\Psi_i(\mathbf{x})$, $u^h_i(\mathbf{x})$, and $\sum u^h_j(\mathbf{x})$ for $D_j \in \mathcal{N}_i$.

$$\sum_{j=1}^{M} u^h_j(\mathbf{x}) = \Psi_{l,k}(\mathbf{x}) + u^h_i(\mathbf{x}) + \sum_{D_j \subset \mathcal{N}_{l,k}} u^h_j(\mathbf{x}) \text{ for } \mathbf{x} \in D_i = S_{l,k} \quad (19)$$

To save computational time, instead of evaluating u^s_i and the harmonic patches at all of the desired points $\mathbf{x}$, we just evaluate them at the boundary points of D_i containing $\mathbf{x}$ and then solve the local Poisson equation again, but with the correct boundary data.

We end this section by estimating the CPU time required. Letting M be the number of leaf nodes and K be the desired order of accuracy, we construct a (scaled) $K \times K$ Chebyshev mesh on each leaf node D_i for $i = 1, \ldots, M$. The total number of discretization points is given by $N = MK^2$. The computational cost of the Poisson solver, to get the solution $u(x)$ at the $N = MK^2$ grid points on M leaf nodes with $K \times K$ grid points each, is of order $N\left(4K + \frac{27p^2}{K^2} + K^2\right)$ where p is the number of terms in the multipole expansion (around 20 for single and 40 for double precision computation). The most salient feature of the present algorithm is its speed. The sixteenth order $K = 16$ accurate implementation requires about 500 floating point operations per grid point. It is only a few times more expensive than a second order nonadaptive method based on Fourier analysis or cyclic reduction, yet it can easily obtain machine precision for both the computed solution and its partial derivatives.

3.2 A layer potential method for the Laplace equation

The classical solution of the interior Dirichlet problem (13) is obtained by representing the solution as a double-layer potential

$$u(\mathbf{x}) = \int_\Gamma \frac{\partial}{\partial \nu_{\mathbf{y}(s)}} G(\mathbf{x}, \mathbf{y}(s)) \mu(s)\, ds \text{ for } \mathbf{y}(s) \in \Gamma = \partial\Omega \quad (20)$$

where $G(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi}\log|\mathbf{x}-\mathbf{y}|$ is the fundamental solution of the Laplace equation in two space dimension. Since the representation automatically satisfies the Laplace equation, it remains to determine the density function μ to match the the boundary condition. If we take the limit as x approaches to a boundary point $x_0 \in \Gamma$, μ satisfies the jump relation

$$g(x_0) = \frac{1}{2}\mu(x_0) + \int_\Gamma \frac{\partial}{\partial \nu_{\mathbf{y}(s)}} G(\mathbf{x},\mathbf{y}(s))\mu(s)\,ds, \tag{21}$$

or equivalently, $(\frac{1}{2}I + K)\mu = g$ where g is a given Dirichlet boundary data and K is the double layer potential operator with $\frac{\partial G}{\partial \nu}$ as a kernel.

Similarly, the solution of the interior Neumann problem (13) is represented via a single layer potential,

$$u(\mathbf{x}) = \int_\Gamma G(\mathbf{x},\mathbf{y}(s))\rho(s)\,ds \tag{22}$$

whose jump relation provides

$$h(\mathbf{x}_0) = -\frac{1}{2}\rho(\mathbf{x}_0) + \int_\Gamma \frac{\partial}{\partial \nu_{\mathbf{x}_0}} G(\mathbf{x}_0,\mathbf{y}(s))\rho(s)\,ds, \tag{23}$$

or equivalently, $(-\frac{1}{2}I + K^*)\rho = h$ where h is the given Neumann boundary data and K^* denotes the normal derivative operator of the single layer potential. It is easy to verify that K^* is the adjoint of double layer potential operator K.

This potential theory based approach is one of the early results in the history of partial differential equation theory and it has many well-known advantages. First, the Laplace equation is reformulated on lower dimensional set thus the problem solving procedure deals with only the boundary points. Second, the formulation is indifferent to geometry, therefore, there is no difference between bounded interior problems and unbounded exterior problems. However, less well emphasized feature of the approach is that it leads to well-conditioned linear systems independent to discretization scheme and order of accuracy which makes it possible to get a stable, high order of accuracy numerical method. Despite of the advantages, the integral equation methods can be useful only with special numerical tools since the integral kernel $G(x,y)$ is non-smooth near the boundary and the integral operator K or K^* is not local. We now describe a fast method [22] to solve the Neumann boundary value problem (23) using the mathematical tools above.

Suppose Γ consists of M disjoint components $\Gamma_k, k = 1,\cdots,M$ (Note that $M = 1$ if the domain is simply connected) and we select N_k boundary points on the k-th boundary component which are equispaced in arclength. Let ρ_i^k is charge density on x_i^k then the trapezoidal rule for (23) gives,

$$-\frac{1}{2}\rho_i^l + \frac{1}{2\pi}\sum_{k=1}^{M}\frac{|\Gamma_k|}{N_k}\sum_{j=1}^{N_k}\frac{\partial}{\partial \nu_{x_i^l}}\ln|x_j^k - x_i^l|\,\rho_j^k = h(x_i^l). \tag{24}$$

Care must be taken when $x_j^k = x_i^l$ to use the appropriate limit $\frac{1}{2}\kappa(x_j^k)$ in place of $\frac{\partial}{\partial \nu_{x_i^l}} \ln|x_j^k - x_i^l|$ where κ denotes the curvature of the boundary curve. It is well known that the trapezoidal rule with equispaced points on simply connected smooth boundary curves provides superalgebraically converging solution ρ as N.

The linear system (24) for ρ_i^l is solved with a conjugate gradient-type algorithm such as Generalized Minimum Residual method (GMRES) and it requires only a few iterations (about 10 or less iteration with 10 complicate boundary curves) to get full precision of accuracy and the number of iteration is independent of the number of boundary points N. What we need to have for the kind of computation is a fast numerical tool for dense matrix-vector multiply in order to compute the field due to a collection of N charge sources at the source locations themselves. It can be performed in $O(N)$ operations instead of $O(N^2)$ direct summation using the Fast Multipole Method (FMM) [5, 12] which is already described in subsection 3.1. Once integral equation is solved, we need to evaluate the solution at many interior points. For this purpose, Mayo's Method presented in [19] may accelerate the computation time in optimal order with small constant though we did not use the method for our current implementation.

4 Experimental Results

The numerical method described in the previous sections has been implemented in GNU C++ and GNU Fortran, C++ for adaptive data structure management and Fortran for numerical computation. In this section, we present performance results obtained from experiments using a SparcStation 2.

Example 1 (Decomposition of the solution). The first example we start with is a simple problem with a prescribed solution $u(x, y)$ which decays exponentially fast near the boundary and the corresponding source term $S(x, y)$.

$$u(x,y) = e^{-\frac{w}{2}(x^2+4y^2)} \tag{25}$$

$$S(x,y) = w[A(wx^2-1) + 4B(4wy^2-1)]\, e^{-\frac{w}{2}(x^2+4y^2)} \tag{26}$$

where $w = 256, A = 1, B = 4$.

Since the smoothness of the elliptic solution solely depends on the smoothness of the coefficient and the source functions, it is possible to automatically generate a computational domain by making the local truncation errors of $A(x, y)$, $B(x, y)$, and $S(x, y)$ in K-th order Chebyshev series on each of leaves D_i to be less than a user provided constant tol_{Cheby}. The order of local Chebyshev expansion and the truncation error bound for this example are set to be $K = 16$ and $tol_{Cheby} = 10^{-6}$. Figure 1 shows the given source function $S(x, y)$ and the computed solution $u(x, y)$ on the computational domain $[-0.5, 0.5] \times [-0.5, 0.5]$ on which 64 boxes or equivalently $16 * 16 * 64 = 16384$ grid points are allocated with three levels of adaptivity.

The plotted solution and $\phi(x,y), \psi(x,y)$ is obtained after 12 times of iteration which takes 108 seconds using a SparcStation II and the estimated $L^2(\Omega)$ converges error $\frac{||\nabla u^k - \nabla u^{k-1}||}{||\nabla u^k||}$ is about $3.4 \cdot 10^{-4}$. The $\nabla\phi(x,y)$ is the cure-free part of the stretched gradient field (Au_x, Bu_y) and $\nabla \times \psi(x,y)$ is the divergence-free part of (u_x, u_y) as shown in Figure 1.

Figure 1: The source function $S(x,y)$, the solution $u(x,y)$ and $\phi(x,y), \psi(x,y)$ defined in (4), (5) of Example 1. The three numbers printed on the lower right corners show the minimum, the step, and the maximum values of the contour lines.

Example 2 (Constant coefficient case). We now examine the convergence of the iterative method for the constant coefficient problems with the prescribed solution $u(x,y) = e^{-200(x^2+y^2)}$ which makes no boundary effect and the corresponding source function $S(x,y)$, in order to simplify the discussion. Figure 2 shows the numerical

results of the convergence error for the constant coefficient problem

$$Au_{xx}(x,y) + Bu_{yy}(x,y) = S(x,y) \tag{27}$$

with various choice of constant B for fixed $A = 1$. For the experiment, we use adaptive domains with 136 boxes for $K = 8$ and 28 boxes for $K = 16$ to make the local truncation errors for $S(x,y)$ to be less than 10^{-5}.

Figure 2: The leftmost figure shows the convergence results with 136 boxes at $K = 8$ and the rightmost figure draws those with 28 boxes at $K = 16$ for Example 2.

Note that the convergence depends only on the ratio of A to B and the role of A and B are interchangeable in the algorithm, therefore, the result for $A = 1, B = 2$ represents those for $A = 100, B = 200$, and $A = 2, B = 1$, so on. And $A = 1, B = 1$ represents the Poisson problem, thus we can get the desired precision with single iteration.

To analize the convergence result, we rewrite the functions of the iterative method in terms of Fourier components $\cos(m\pi x)\cos(n\pi y)$, then (8), (9) becomes

$$\phi_{mn}^{k+1} = \frac{1}{m^2+n^2} f_{mn} + (B-A)\frac{mn}{m^2+n^2}\psi_{mn}^{k} \tag{28}$$

$$\psi_{mn}^{k+1} = \left(\frac{1}{B} - \frac{1}{A}\right)\frac{mn}{m^2+n^2}\phi_{mn}^{k+1}. \tag{29}$$

Therefore,

$$\phi_{mn}^{k+1} = \frac{1}{m^2+n^2} f_{mn} - \left(\sqrt{\frac{B}{A}} - \sqrt{\frac{A}{B}}\right)^2 \left(\frac{mn}{m^2+n^2}\right)^2 \phi_{mn}^{k} \tag{30}$$

which shows the iterative method for constant coefficients has linear convergence with asymptotic convergence constant λ,

$$\lambda = \left(\sqrt{\frac{B}{A}} - \sqrt{\frac{A}{B}}\right)^2 \left(\frac{mn}{m^2+n^2}\right)^2 \leq \frac{1}{4}\left(\sqrt{\frac{B}{A}} - \sqrt{\frac{A}{B}}\right)^2 \quad \text{since} \quad \frac{mn}{m^2+n^2} \leq \frac{1}{2} \tag{31}$$

and the iterative sequence is guaranteed to converge when $3 - 2\sqrt{2} \leq \frac{A}{B} \leq 3 + 2\sqrt{2}$. From this example, we can conclude that linear transformation defined below may accelerate the convergence significantly for the constant or slowly varying coefficient problems.

$$\frac{\partial}{\partial x'}\left[\frac{A(x',y')}{A'}\frac{\partial}{\partial x'}u(x',y')\right] + \frac{\partial}{\partial y'}\left[\frac{B(x',y')}{B'}\frac{\partial}{\partial y'}u(x',y')\right] = S(x',y') \tag{32}$$

where $A' = ||A(x,y)||_{L_1(\Omega)}$, $B' = ||B(x,y)||_{L_1(\Omega)}$, $x = \sqrt{A'}x'$, and $y = \sqrt{B'}y'$.

Example 3 (Scalar conductivity problem). We consider an example where $A(x,y)$ and $B(x,y)$ are identical functions representing scalar conductivity,

$$\nabla \cdot [a(x,y)\nabla u(x,y)] = S(x,y) \tag{33}$$

for $a(x,y) = (1+\frac{1}{\kappa}) + (1-\frac{1}{\kappa})\cos(2\pi(x-y))$, $S(x,y) = xy(1-4x^2)^4(1-4y^2)^4$. The convergence of iterative methods for such a problem strongly depends on the ratio, which is κ for this example, of the maximum value to the minimum value of the conductivity $a(x,y)$ over the domain $[7,10]$. Table 1 summarizes the convergence result with various choices of κ. The result demonstrates that the numerical scheme works very well even problems with wide variance of conductivity $16 < \kappa < 64$.

κ	# of	convergence error after k-th of iteration					
	boxes	k=2	k=4	k=6	k=8	k=10	k=12
2	136	$1.7{\cdot}10^{-4}$	$4.4{\cdot}10^{-8}$	N.A.	N.A.	N.A.	N.A.
4	256	$3.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-5}$	N.A	N.A.	N.A.	N.A.
8	280	$2.0{\cdot}10^{-2}$	$4.1{\cdot}10^{-4}$	$9.0{\cdot}10^{-6}$	$2.1{\cdot}10^{-7}$	N.A.	N.A.
16	298	$8.3{\cdot}10^{-2}$	$6.0{\cdot}10^{-3}$	$4.9{\cdot}10^{-4}$	$4.3{\cdot}10^{-5}$	$4.0{\cdot}10^{-6}$	$3.8{\cdot}10^{-7}$
32	301	$2.7{\cdot}10^{-1}$	$6.0{\cdot}10^{-2}$	$1.5{\cdot}10^{-2}$	$4.1{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$3.5{\cdot}10^{-4}$
64	307	$6.8{\cdot}10^{-1}$	$4.2{\cdot}10^{-1}$	$3.0{\cdot}10^{-1}$	$2.3{\cdot}10^{-1}$	$1.9{\cdot}10^{-1}$	$1.5{\cdot}10^{-1}$
128	310	1.14	1.25	1.43	1.56	1.63	1.67

Table 1: Convergence results for Example 3 with respect to κ. # of boxes denotes the total number of boxes on the adaptive tree to get $tol_{Chevby} = 10^{-4}$ and the number of points, for example, in 256 boxes is 256 $K^2 = 65536$ for $K = 16$. N.A. means that the computation has been stopped after the desired accuracy 10^{-6} is achieved.

ACKNOWLEDGEMENTS: This work was supported by the STEPI under project number 97N6-0101-A4 and by the Ewha womans university.

References

[1] C. Anderson, Domain decomposition techniques and the solution of poisson's equation in infinite domains, In *the Second International Symposium on Domain Decomposition methods* pages 129–139, (1987).

[2] Paul A. Bernhardt and J. U. Brackbill, Solution of Elliptic Equations Using Fast Poisson Solvers, *J. Comput. Phys.*, **53**, 382-394, (1984).

[3] A. Brandt, Multi-level adaptive solutions to boundary value problems, *Math. Comp.* **31** 330–390, (1977).

[4] J. U. Brackbill and D.W. Forslund, An implicit method for electromagnetic plasma simulation in two dimensions, *J. Comput. Phys.* **46** (2), 271-308, (1982).

[5] J. Carrier, L. Greegard, and V. Rokhlin, A fast adaptive multipole algorithm for particle simulation, *SIAM J. Sci. Stat. Comput.* **9**(4) 669-686 (1987).

[6] T. F. Chan and D. C. Resasco, A domain-decomposed fast poisson solver on a rectangle, *SIAM J. Sci. Stat. Comput.* **8**(1) S14–26, (1987).

[7] P. Concus and G. H. Golub, Use of Fast Methods for the Efficient numerical solution of Nonseperable Elliptic Equations, *SIAM J. Numer. Anal.*, **10**, No. 6, 1103-1120, (1973).

[8] E. Detyna, Rapid Elliptic Solver in " Sparse Matrices and Their Uses" (I. S. Duff, Ed.), *Academic Press, London/New York*, (1981).

[9] Fred W. Dorr, The Direct Solution of the Discrete Poisson Equation on a Rectangle, *SIAM Review*, **12**, No. 2, pp248-263 (1970).

[10] A. Greenbaum, Diagonal scalings of the Laplacian as preconditioners for other elliptic differential operators. *SIAM J. Matrix Anal. Appl.*, **13** no. 3, 826–846, (1992).

[11] L. Greengard and J.-Y. Lee, A Direct Adaptive Poisson Solver of Arbitrary Order Accuracy, *J. Comput. Phys.*, **125**, 415-424, (1996).

[12] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **73** 325–348, (1987).

[13] M. Griebel, Parallel domain-oriented multilevel methods, *SIAM J. Sci. Comput.* **16**(5) 1105–1125, September (1995).

[14] Dale B. Haidvogel, The Accurate Solution of Poisson's by Expansion in Chebyshev Polinomials, *J. Comput. Phys.*, **30**, 167-180, (1979).

[15] S. Kim, Parallel multidomain iterative algorithms for the helmbholtz wave equation, *Appl. Numer. Math.* **17** 411–429, (1995).

[16] D. S. Lee, Fast parallel solution of the poisson equation on irregular domains, *Numer. Algorithms* **8**(2-4) 347–362, (1994).

[17] J.-Y. Lee and K. Jeong, A parallel Poisson solver using the fast multipole method on networks of workstations, *Comp. and Math. with applications* **35** (1998).

[18] R. E. Lynch and J. R. Rice, High accurate finite difference approximations to solutions of elliptic partial differential Equations, *Proc. Nat. Acad. Sci.*, **75**, 2541-2544, (1979).

[19] A. Mayo Fast high order accurate solution of Laplace's equation on irregular regions *SIAM J. Sci. Stat. Comput.*, **6** 144-157, (1985).

[20] B. E. Mcdonald, The Chebychev Method for Solving Nonself-Adjoint Elliptic Equations on a Vector Computer, *J. Comput. Phys.*, **35**, 147-168, (1980).

[21] A. Pares-Sierra and G. K. Vallis, A Fast Semi-direct Method for the Numerical Solution of Non-separable Elliptic Equations in Irregular Domains, *J. Comput. Phys.*, **82**, 398-412, (1989).

[22] V. Rokhlin, Rapid solution of integral equations of classical potential theory, *J. Comput. Phys.*, **60**, 187-207 (1985).

[23] U. Schumann and M. Strietzel, Parallel solution of tridiagonal systems for the poisson equation, *J. Sci. Comput.* **10**(2) 181–190, (1995).

[24] P. N. Swarztrauber, Approximate Cyclic Reduction for solving Poisson Equation, *SIAM J. Sci. Stat. Comput.*, **8**, No. 3, 199-209, (1987).

[25] P. N. Swarztrauber and R. A. Sweet, Vector and parallel methods for the direct solution of poisson's equation, *J. Comput. Appl. Math.* **27**(1-2) 241–263, (1989).

[26] C. Temperton, Direct Method for the Solution of the Discrete Poisson Equation: Some Comparisons, *J. Comput. Phys.*, **31**, 1-20, (1979).

[27] L. Vozovoi, M. Israeli and A. Averbuch, High Order Fast Elliptic Solver in Rectangular Regions, *Technical Report # 846*, (1995).