A High-Order Direct Solver for Helmholtz Equations with Neumann Boundary Conditions

Xian-He Sun[*] *Yu Zhuang*

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

Abstract

In this study, a compact finite-difference discretization is first developed for Helmholtz equations on rectangular domains. Special treatments, then, are introduced for Neumann and Neumann-Dirichlet boundary conditions to achieve accuracy and separability. Finally, a Fast Fourier Transform (FFT) based technique is used to yield a fast direct solver. Analytical and experimental results show this newly proposed solver is comparable to the conventional second-order elliptic solver when accuracy is not a primary concern and is significantly faster than that of the conventional solver if a highly accurate solution is required. In addition, this newly proposed fourth order Helmholtz solver is parallel in nature. It is readily available for parallel and distributed computers. The compact scheme introduced in this study is likely extendible for sixth-order accurate algorithms and for more general elliptic equations.

[*]This research was supported in part by the National Aeronauticsand Space Administration under NASA contract No. NAS1-19480 whilethe second author was in residence at the Institute forComputer Applications in Science and Engineering (ICASE), NASALangley Research Center, Hampton, VA 23681-0001, and by NASAcontract No. NAS1-1672 and Louisiana Education Quality Support Fund.

1 Introduction

Obtaining a more accurate numerical solution, in general, means adding more discretization points and using smaller mesh sizes, which cost both computing time and storage space. The demand for more accurate solutions is a driving force for more powerful computers [15]. On the other hand, however, high performance computers require high-order accurate discretization methods to match their computation power and to explore the potential of high-performance computing. With the availability of high performance computers, the current barrier in utilizing existing hardware in many situations is the lack of high-order accurate numerical methods. In this study we introduce a high-order direct solver for Helmholtz equations with Neumann boundary conditions. This newly proposed solver achieves fourth-order accuracy with a computation count compatible with the best existing second-order algorithm. Of equal importance is its parallel nature and its readiness for parallel and distributed computers.

Solving Helmholtz equations is a key issue of scientific computing. Intensive research has been done in recent years in the field to develop efficient numerical methods [16]. In the late sixties, Hockney [4] and Bunemann [1] developed fast direct methods for elliptic equations on rectangular uniform meshs. These methods take advantage of the special block structure of the resulting system of finite-difference discretizations and reduce the number of computations considerably. While these methods have been highly recognized for their practical importance, they are based on second order approximations. By adopting a novel finite-difference discretization, in 1979 Houstis and Papatheodorou [6] proposed a direct Helmholtz-Dirichlet solver with fourth-order accuracy. This fourth-order solver combines the techniques of Fourier transform and cyclic reduction and is as fast as the second-order solvers developed by Hockney and Bunemann. The main drawback of Houstis and Papatheodorou's method is that it is designed for Dirichlet boundary conditions only. Although, analytically, solving a Neumann problem is equivalent to solving two Dirichlet problems for Helmhotz equations [7], due to the high transformation cost[1] Houstis and Papatheodorou's method is unacceptably slow for Helmholtz equations with Neumann or Neumann-Dirichlet conditions.

Finite-element schemes provide another alternative for high-order discretizations. By using Rayleigh-Ritz-Galerkin approach with tensor product B-splines, Kaufman and Warner [8, 9] developed a direct solver more recently for separable elliptic equations on rectangular domains. Their method permits high-order discretizations and works for both Dirichlet and Neumann boundary problems. While Kaufman and Warner's method is a more general elliptic solver, it has a computational complexity of $O(N^3)$ on a square domain of size $N \times N$.

In this paper, we propose a fourth-order direct solver for Helmholtz equations

$$P_{xx} + P_{yy} - \lambda P = R(x, y) \quad \text{in } \Omega, \tag{1}$$

[1] As shown in Appendix B, the complexity of computing the transformation influence matrix is at least $O(N^3 \log_2 \log_2 N)$ on a square domain of size $N \times N$.

where Ω is a rectangular computational domain with Neumann boundary conditions imposed on at least one dimension, $\lambda \geq 0$, R is a given function, P is the scalar-valued function to be solved for. When $\lambda = 0$, (1) becomes a Poisson equation. Mathematical analyses are conducted to prove our newly proposed solver is fourth order accurate and requires $N^2(5\log_2 N+17)+119N$ operations on a $N \times N$ domain. These analytical results are confirmed by experimental measurements. Performance comparison has been made against the best second-order algorithm available. Analytical and experimental results show this newly proposed high-order solver is significantly faster than existing algorithms for Helmholtz equations with Neumann conditions, especially for large applications.

The main mathematical tool used in this study is a newly-derived finite-difference discretization scheme, which is a generalization of the compact finite-difference scheme originally proposed by Kreiss and Oliger [10]. This paper is organized as follows. Compact schemes are presented in Section 2. A high order discretization is derived for Laplace operator. Our new discretization scheme for Helmhotz equations is introduced in Section 3. Based on the newly developed finite-difference scheme, a high-order fast solver is proposed in Section 4 for Helmholtz equations with Neumann conditions. The accuracy and efficiency of this algorithm are also analyzed. Extension of the fast solver to Neumann-Dirichlet conditions is discussed in Section 5. Finally, testing results are presented in Section 6. Section 7 gives the conclusion. To provide an appropriate comparison, accuracy analysis of the conventional second order solver and the influence matrix method are presented in Appendix A and B, respectively. A detailed error estimation of the newly proposed algorithm is conducted in Appendix C.

2 Compact Difference Schemes

Compact finite-difference schemes are derivative approximation methods that express a linear combination of derivatives in terms of a linear combination of function values [13]. As originally suggested by Kreiss and Oliger [10], and later discussed for fluid dynamics problems by Hirsh and Lele [3, 11], the first and second derivatives for compact differences can be approximated by

$$f'_n = \left(\frac{D_0}{1+\frac{1}{6}h^2 D_+ D_-}\right) f_n \qquad \text{with error } -\frac{h^4}{180} f^{(5)} \tag{2}$$

and

$$f''_n = \left(\frac{D_+ D_-}{1+\frac{1}{12}h^2 D_+ D_-}\right) f_n \qquad \text{with error } -\frac{h^4}{240} f^{(6)}, \tag{3}$$

where

$$D_0 f_n = \frac{1}{2h}(f_{n+1} - f_{n-1}), \qquad D_+ f_n = \frac{1}{h}(f_{n+1} - f_n), \qquad D_- f_n = \frac{1}{h}(f_n - f_{n-1}), \tag{4}$$

and h is the mesh size. Multiply (2) and (3) by their respective denominators yields

$$\frac{1}{6} f'_{n-1} + \frac{2}{3} f'_n + \frac{1}{6} f'_{n-1} = \frac{f_{n+1} - f_{n-1}}{2h},$$

and

$$\frac{1}{12}f''_{n-1} + \frac{5}{6}f''_n + \frac{1}{12}f''_{n-1} = \frac{f_{n+1} - 2f_n + f_{n-1}}{h^2}. \tag{5}$$

As Hirsh has shown, the above two compact schemes yield a smaller difference stencil and higher accuracy for approximating f' and f'' than the traditional fourth order central-difference.

Compact scheme has traditionally been used only for approximating derivatives with known function values. For instance, compact schemes have often been used to compute one dimensional time derivatives in the iterative solving of differential equations where the function values of the last iteration are known. To approximate high-dimensional spatial differential operators, however, using 1-D compact schemes to approximate corresponding 1-D components of Partial Differential Equations (PDE) does not necessarily lead to a solution. For example, to discretize the Poisson equation

$$P_{xx} + P_{yy} = R(x, y) \tag{6}$$

if we use (5) to approximate P_{xx} and P_{yy} we will end up with

$$\begin{aligned} &\tfrac{1}{12}P_{xx}^{i-1,j} + \tfrac{5}{6}P_{xx}^{i,j} + \tfrac{1}{12}P_{xx}^{i+1,j} + \tfrac{1}{12}P_{yy}^{i,j-1} + \tfrac{5}{6}P_{yy}^{i,j} + \tfrac{1}{12}P_{yy}^{i,j+1} \\ &\qquad = \tfrac{1}{h^2}\left(P^{i-1,j} + P^{i+1,j} + P^{i,j+1} + P^{i,j-1} - 4P^{i,j}\right). \end{aligned} \tag{7}$$

The left-hand side of (7) is not a linear combination of the Laplace operator $\triangle P = \frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2}$, hence relation (6) cannot be used to reduce (7) into a solvable linear equation. Using compact schemes for the solution of spatial differential operators is still state-of-the-art and very challenging. Before deriving a high-order solution for Helmholtz equations, we first need to investigate the applicability of compact schemes for two dimensional Laplace operators.

An example of compact finite-difference scheme for the 2-D Laplacian is given below when both x and y dimensions take the same uniform mesh size h:

$$\triangle P^{i,j} = \left(\frac{\frac{1}{6}(D_+^x D_+^y + D_+^x D_-^y + D_-^x D_+^y + D_-^x D_-^y) + D_+^x D_-^x + D_+^y D_-^y}{1 + \frac{1}{12}h^2(D_+^x D_-^x + D_+^y D_-^y)} \right) P^{i,j} \tag{8}$$

where

$$\begin{aligned} D_+^x P^{i,j} &= \tfrac{1}{h}(P^{i+1,j} - P^{i,j}), \qquad & D_-^x P^{i,j} &= \tfrac{1}{h}(P^{i,j} - P^{i-1,j}), \\ D_+^y P^{i,j} &= \tfrac{1}{h}(P^{i,j+1} - P^{i,j}), \qquad & D_-^y P^{i,j} &= \tfrac{1}{h}(P^{i,j} - P^{i,j-1}). \end{aligned} \tag{9}$$

The truncation error of (8) is

$$Error(\triangle P^{i,j}) = h^4 \left[\frac{1}{216}\left(\frac{\partial^6}{\partial x^4 \partial y^2} + \frac{\partial^6}{\partial x^2 \partial y^4}\right) - \frac{1}{360}\left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right) \right] P^{i,j}.$$

The difference stencils of (8) can be seen easily in the form of

$$-\begin{pmatrix} & \frac{1}{8} & \\ \frac{1}{8} & 1 & \frac{1}{8} \\ & \frac{1}{8} & \end{pmatrix} \triangle P^{i,j} = h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} P^{i,j}. \tag{10}$$

A more general form of the compact difference scheme for the Laplace operator is

$$\begin{array}{lll} \triangle P^{i,j} & + \alpha \left(\triangle P^{i-1,j} + \triangle P^{i+1,j} + \triangle P^{i,j-1} + \triangle P^{i,j+1} \right) \\ & + \beta \left(\triangle P^{i-1,j-1} + \triangle P^{i+1,j-1} + \triangle P^{i-1,j+1} + \triangle P^{i+1,j+1} \right) \\ = & a \, \frac{P^{i+1,j}+P^{i-1,j}+P^{i,j-1}+P^{i,j+1}-4P^{i,j}}{h^2} \\ & + b \, \frac{P^{i-1,j-1}+P^{i+1,j-1}+P^{i-1,j+1}+P^{i+1,j+1}-4P^{i,j}}{2h^2} \, . \end{array} \tag{11}$$

The values of the parameters a, b, and α, β can be derived by matching coefficients of the corresponding Taylor series of the Laplace operator for various orders of accuracy. The first unmatched coefficient determines the truncation error of the approximation (11). After some mathematical manipulation, direct matching leads to the following results:

$$1 + 4\alpha + 4\beta = a + b \qquad \text{(second order accuracy)},$$

$$\left. \begin{array}{l} \alpha + 2\beta = \frac{a+b}{12} \\ 2\alpha + 4\beta = \frac{b}{2} \end{array} \right\} \qquad \text{(fourth order accuracy)}.$$

The highest order (11) can reach is $O(h^4)$, and (10) is the optimal one with three non-zero coefficients, which is attained with β set to 0.

Compared with conventional fourth order finite-difference

$$\begin{array}{lll} \triangle P^{i,j} & = & \frac{4}{3} \cdot \frac{P^{i-1,j}+P^{i+1,j}+P^{i,j-1}+P^{i,j+1}-4P^{i,j}}{h^2} \\ & - & \frac{1}{3} \cdot \frac{P^{i-2,j}+P^{i+2,j}+P^{i,j-2}+P^{i,j+2}-4P^{i,j}}{4h^2} + O(h^4), \end{array}$$

though compact scheme (10) does not reduce the number of stencils, it has the advantage in handling grid points in the neighborhood of the boundaries (see [2] pp. 568). More importantly, its block tridiagonal structure makes fast direct solving possible.

Discretization (10) is for Laplace operators. To apply compact schemes to Helmholtz equation (1), we need modify discretization (2) and (8) accordingly. Two basic finite-difference formulas are needed for the modification. They are:

$$D^2 f_n = D_+ D_- f_n + O(h^2), \tag{12}$$

and

$$\triangle P^{i,j} = (D_+^x D_-^x + D_+^y D_-^y) P^{i,j} + O(h^2), \tag{13}$$

where D_+^x, D_-^x, D_+^y, and D_-^y, are defined in (9), D_+ and D_- in (4), and D^2 denotes the second derivative.

Both (2) and (8) are of order four. Replacing $D_+ D_-$ in the denominator of (2) by the second differential operator D^2, and $(D_+^x D_-^x + D_+^y D_-^y)$ in the denominator of (8) by the Laplace operator $\triangle$, yields

$$f_n' = \left(\frac{D_0}{1 + \frac{1}{6} h^2 \; D^2} \right) f_n$$

and

$$\triangle P^{i,j} = \left(\frac{\frac{1}{6}(D_+^x D_+^y + D_+^x D_-^y + D_-^x D_+^y + D_-^x D_-^y) + D_+^x D_-^x + D_+^y D_-^y}{1 + \frac{1}{12} h^2 \triangle} \right) P^{i,j}.$$

The order of truncation error of the two modified compact schemes remain the same. Multiplying each equation with its corresponding denominator, the explicit forms of the above two equations are given by

$$\frac{1}{2h}(f_{n+1} - f_{n-1}) = f_n' + \frac{1}{6} h^2 f_n''' + O(h^4) \tag{14}$$

and

$$h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} P^{i,j} = -\frac{3}{2} \triangle P^{i,j} - \frac{h^2}{8} \triangle^2 P^{i,j} + O(h^4), \tag{15}$$

respectively.

3 High Order Discretization for Helmholtz Equations

We assume equation (1) to be solved in a rectangular region $[0, x] \times [0, y]$. This rectangular domain is partitioned into uniform mesh size h in both dimensions, yielding grid points

$$(0,0),\ (h,0),\ ...,\ (Mh,0)\ ;\ ...\ ;\ (0,Nh),\ (h,Nh),\ ...,\ (Mh,Nh). \tag{16}$$

We apply the modified compact scheme (15) to discretize equation (1) with grid (16). Substitute $\triangle P^{i,j}$ by $\lambda P^{i,j} + R^{i,j}$, and $\triangle^2 P^{i,j}$ by $\lambda^2 P^{i,j} + \lambda R^{i,j} + \triangle R^{i,j}$, based on the equivalences derived from equation (1), equation (15) becomes

$$h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} P^{i,j} = -\frac{3}{2} \left(\lambda P^{i,j} + R^{i,j} \right) - \frac{h^2}{8} \left(\lambda^2 P^{i,j} + \lambda R^{i,j} + \triangle R^{i,j} \right) + O(h^4). \tag{17}$$

Move all the $P^{i,j}$ terms on the right-hand side of (17) to the left, yielding

$$h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 + \frac{\lambda h^2 + 12}{8} \lambda h^2 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} P^{i,j} = -\frac{3}{2} R^{i,j} - \frac{h^2}{8} \left(\lambda R^{i,j} + \triangle R^{i,j} \right) + O(h^4). \tag{18}$$

The approximation of $\triangle R$ in the above equation only needs to be second order accurate for the final solution P to remain fourth order accurate.

To provide (18) with an adequately accurate boundary treatment necessitates a fourth-order first derivative approximation for the boundary conditions. As stated in Section 1, equation (1) has a Neumann-Neumann or Neumann-Dirichlet boundary conditions. Since Neumann-Neumann boundary problems and Neumann-Dirichlet problems share many common characteristics, we discuss the solution of the Neumann-Neumann boundary problem in detail in this section and, then, extend the solution to Neumann-Dirichlet boundary problems in Section 5.

General Neumann-Neumann boundary conditions can be expressed mathematically as

$$\frac{dP}{dn} = b(x, y) \quad \text{on } \partial\Omega, \tag{19}$$

where $\partial\Omega$ is the boundary of the rectangular domain Ω, n is the normal vector of the boundary of the computational domain, and $\frac{dP}{dn}$ indicates the derivative normal to the boundary.

Applying equation (14) at grid point $(0, j)$, we have

$$\frac{h^2}{6} P^{0,j}_{xxx} + P^{0,j}_{x} = \frac{P^{1,j} - P^{-1,j}}{2h} + O(h^4).$$

Incorporating this result into boundary condition (19) yields

$$\frac{h^2}{6} b^{0,j}_{xx} + b^{0,j} = \frac{P^{1,j} - P^{-1,j}}{2h} + O(h^4). \tag{20}$$

Difficulty exists in using Equation (20), however. The boundary function $b(x, y)$ is usually only available at the boundaries, but not in a small neighborhood of the boundaries. Thus, the derivative of function $b(x, y)$ can be found only along the y-direction but not the x-direction for grid points $(0, j)$. To overcome this difficulty, we reformulate derivative $b^{0,j}_{xx}$ by incorporating information from the equation (1),

$$\begin{aligned} b^{0,j}_{xx} &= P^{0,j}_{xxx} = \tfrac{\partial}{\partial x}\left(\triangle P^{0,j} - P^{0,j}_{yy}\right) = \tfrac{\partial}{\partial x}\triangle P^{0,j} - \tfrac{\partial}{\partial x} P^{0,j}_{yy} \\ &= \tfrac{\partial}{\partial x}(R^{0,j} + \lambda P^{0,j}) - \tfrac{\partial^2}{\partial y^2} P^{0,j}_{x} \ = R^{0,j}_{x} + \lambda b^{0,j} - b^{0,j}_{yy}. \end{aligned}$$

Therefore, we have transformed (20) into

$$\frac{P^{1,j} - P^{-1,j}}{2h} = \frac{h^2}{6}(R_x - b^{0,j}_{yy}) + (1 + \frac{\lambda h^2}{6}) b^{0,j} + O(h^4). \tag{21}$$

The approximation of $b^{0,j}_{yy}$ in the above equation only needs to be second order for (21) to remain fourth order, and it is feasible to carry the y-direction derivative approximation at the boundary of x-direction.

The value $P^{-1,j}$ in (21) which lies outside the computational domain is eliminated when both equations (21) and (18) are applied to grid points $(0, j)$. Through the above discretization and derivation, the Helmholtz equation (1) and the boundary conditions (19) now can be incorporated into the following linear system:

$$\mathbf{A}\ P = -\frac{3}{2} h^2 R - \frac{h^4}{8}\left(\lambda R + \triangle R\right) + 2hU \ + O(h^5)\ . \tag{22}$$

The vector U results from the boundary conditions and vanishes at interior points. It has different

forms along the four boundaries and at the four corners and is given by

$$\begin{aligned} U^{0,j} &= -\frac{3P_x^{0,j}}{2} - \lambda h^2 \frac{(P_x^{0,j-1}+4P_x^{0,j}+P_x^{0,j+1})}{24} - h^2 \frac{(R_x^{0,j-1}+4R_x^{0,j}+R_x^{0,j+1})}{24}, \\ U^{m,j} &= \frac{3P_x^{m,j}}{2} + \lambda h^2 \frac{(P_x^{m,j-1}+4P_x^{m,j}+P_x^{m,j+1})}{24} + h^2 \frac{(R_x^{m,j-1}+4R_x^{m,j}+R_x^{m,j+1})}{24}, \end{aligned}$$

$$U^{i,0} = -\frac{3P_y^{i,0}}{2} - \lambda h^2 \frac{(P_y^{i-1,0}+4P_y^{i,0}+P_y^{i+1,0})}{24} - h^2 \frac{(R_x^{i-1,0}+4R_x^{i,0}+R_x^{i+1,0})}{24}, \tag{23}$$

$$U^{i,n} = \frac{3P_y^{i,n}}{2} + \lambda h^2 \frac{(P_y^{i-1,n}+4P_y^{i,n}+P_y^{i+1,n})}{24} + h^2 \frac{(R_x^{i-1,n}+4R_x^{i,n}+R_x^{i+1,n})}{24}, \tag{24}$$

for $1 \le i \le m-1, 1 \le j \le n-1$, and

$$\begin{aligned} U^{0,0} = & -(\tfrac{1}{4} + \tfrac{\lambda h^2}{24}) \left(5(P_x^{0,0} + P_y^{0,0}) + P_x^{0,1} + P_y^{1,0}\right) - h^2 \frac{5(R_x^{0,0}+R_y^{0,0})+R_x^{0,1}+R_y^{1,0}}{24} \\ & + \frac{5(P_x^{0,0}+P_y^{0,0})-12(P_x^{0,1}+P_y^{1,0})+9(P_x^{0,2}+P_y^{2,0})-2(P_x^{0,3}+P_y^{3,0})}{24}, \\ U^{m,0} = & (\tfrac{1}{4} + \tfrac{\lambda h^2}{24}) \left(5(P_x^{m,0} - P_y^{m,0}) + P_x^{m,1} - P_y^{m-1,0}\right) + h^2 \frac{5(R_x^{m,0}-R_y^{m,0})+R_x^{m,1}-R_y^{m-1,0}}{24} \\ & - \frac{5(P_x^{m,0}-P_y^{m,0})-12(P_x^{m,1}-P_y^{m-1,0})+9(P_x^{m,2}-P_y^{m-2,0})-2(P_x^{m,3}-P_y^{m-3,0})}{24}, \end{aligned}$$

$$\begin{aligned} U^{0,n} = & -(\tfrac{1}{4} + \tfrac{\lambda h^2}{24}) \left(5(P_x^{0,n} - P_y^{0,n}) + P_x^{0,n-1} - P_y^{1,n}\right) - h^2 \frac{5(R_x^{0,n}-R_y^{0,n})+R_x^{0,n-1}-R_y^{1,n}}{24} \\ & + \frac{5(P_x^{0,n}-P_y^{0,n})-12(P_x^{0,n-1}-P_y^{1,n})+9(P_x^{0,n-2}-P_y^{2,n})-2(P_x^{0,n-3}-P_y^{3,n})}{24}, \\ U^{m,n} = & (\tfrac{1}{4} + \tfrac{\lambda h^2}{24}) \left(5(P_x^{m,n} + P_y^{m,n}) + P_x^{m,n-1} + P_y^{m-1,n}\right) + h^2 \frac{5(R_x^{m,n}+R_y^{m,n})+R_x^{m,n-1}+R_y^{m-1,n}}{24} \\ & - \frac{5(P_x^{m,n}+P_y^{m,n})-12(P_x^{m,n-1}+P_y^{m-1,n})+9(P_x^{m,n-2}+P_y^{m-2,n})-2(P_x^{m,n-3}+P_y^{m-3,n})}{24}. \end{aligned}$$

The matrix $\mathbf{A}$ is an $(M+1)(N+1)$ by $(M+1)(N+1)$ matrix given by

$$\mathbf{A} = \begin{pmatrix} A_1 & -2A_2 & 0 & \cdot & 0 & 0 & 0 \\ -A_2 & A_1 & -A_2 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & -A_2 & A_1 & -A_2 \\ 0 & 0 & 0 & \cdot & 0 & -2A_2 & A_1 \end{pmatrix}, \tag{25}$$

with matrix $\mathbf{A}_1$ of order $(N+1)$ by $(N+1)$ being

$$\mathbf{A_1} = \begin{pmatrix} d & -2 & 0 & \cdots & 0 & 0 & 0 \\ -1 & d & -1 & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & -1 & d & -1 \\ 0 & 0 & 0 & \cdots & 0 & -2 & d \end{pmatrix},$$

where $d = 5 + \frac{\lambda h^2+12}{8}\lambda h^2$, and with matrix $\mathbf{A_2}$ of order $(N+1)$ by $(N+1)$ being

$$\mathbf{A_2} = \begin{pmatrix} 1 & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{4} & 1 & \frac{1}{4} & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{2} & 1 \end{pmatrix}.$$

The regular structure of matrix $\mathbf{A}$ allows it to be tridiagonalized by Fast Cosine Transform (FCT)

$$\mathbf{A} = \mathbf{F}\mathbf{\Lambda}\mathbf{F}^{-1}, \tag{26}$$

where $\mathbf{\Lambda}$ is the resulting matrix, and $\mathbf{F}$ is an $(M+1)(N+1)$ by $(M+1)(N+1)$ block diagonal matrix with $(N+1)$ by $(N+1)$ diagonal submatrix $\mathbf{F_1}$ given by

$$\mathbf{F_1} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdot & 1 \\ 1 & cos(\frac{\pi}{N}) & cos(\frac{2\pi}{N}) & cos(\frac{3\pi}{N}) & \cdot & cos(\frac{N\pi}{N}) \\ 1 & cos(\frac{2\pi}{N}) & cos(\frac{4\pi}{N}) & cos(\frac{6\pi}{N}) & \cdot & cos(\frac{2N\pi}{N}) \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & cos(\frac{N\pi}{N}) & cos(\frac{2N\pi}{N}) & cos(\frac{3N\pi}{N}) & \cdot & cos(\frac{N^2\pi}{N}) \end{pmatrix}.$$

After tridiagonalization, equation (22) becomes

$$h^{-2}\mathbf{F}\mathbf{\Lambda}\mathbf{F}^{-1}P = B\ , \tag{27}$$

where $B = -\frac{3}{2}R - \frac{h^2}{8}(\lambda R + \triangle R) + 2h^{-1}U$. Finally, the solution is

$$P = h^2\ \mathbf{F}\mathbf{\Lambda}^{-\mathbf{1}}\mathbf{F}^{-1}B. \tag{28}$$

4 The Fourth-Order Fast Solver

Based on the mathematical study given in the last section, a fourth-order algorithm can be carried out in the following steps:

1) Compute the cosine values to be used in the FCT.

2) Compute the value of each entry in matrix $\mathbf{\Lambda}$.

3) Compute the vector U in equation (22).

4) Compute the right-hand side of (27).
   This is done by adding the results from step 1) to $-\frac{3}{2}R - \frac{h^2}{8}(\lambda R + \triangle R)$ which is to be computed in this step.

5) Apply inverse FCT to the right-hand side of equation (27).
This is done by multiplying matrix $\mathbf{F}_1^{-1}$ to each block B_i in the matrix B, for $i = 0, 1, ..., M$, where $B_i = (b_{i0},\ b_{i1},\ \ldots,\ b_{iN})^T$ is the i-th N-component-long block of vector B.

6) Solve the tridiagonal system $\mathbf{\Lambda} Y = \mathbf{F}^{-1} B$ for Y, where $Y = \mathbf{F}^{-1} P$.
Because of the particular structure of matrix $\mathbf{\Lambda}$, we get N independent tridiagonal systems, each of size M. Reassemble the right-hand side $\mathbf{F}^{-1} B$ according to the structure of $\mathbf{\Lambda}$, and then solve the N tridiagonal systems.

7) Apply FCT to vector Y computed in step 6) to recover P.
This is done by multiplying matrix $\mathbf{F}_1$ to each of vector Y's N-component-long block Y_i for $i = 0, 1, ..., M$.

The operation count of each step of the fourth-order algorithm on a square domain of $N \times N$, from steps 1 to 7, is:

1. N multiplications and $0.5N$ additions;
2. $6N$ multiplications and $2N$ additions;
3. $36N$ multiplications and $48N$ additions;
4. $N^2 + 10N$ multiplications and $4N^2 + 12N$ additions;
5. $N^2 \log_2 N + N(\log_2 N - 4)$ multiplications and $N^2(1.5 \log_2 N + 1.5) - N \log_2 N$ additions;
6. $5N^2 + 6N$ multiplications and $3N^2 + 4N$ additions;
7. $N^2 \log_2 N + N(\log_2 N - 3)$ multiplications and $N^2(1.5 \log_2 N + 2.5) - N \log_2 N$ additions.

The total operation count of the algorithm is the sum of the work of the seven steps, which is

$$\begin{aligned} &N^2(6 + 2\log_2 N) + N(2\log_2 N + 52) \text{ multiplications and} \\ &N^2(3\log_2 N + 11) + N(-2\log_2 N + 67) \text{ additions} \\ &= N^2(5\log_2 N + 17) + 119N. \end{aligned} \tag{29}$$

It is $N^2 + 45N$ multiplications and $4N^2 + 53N$ additions more than the operation count of the conventional second-order fast Poisson solver (see (40) in Appendix A [4]).

Execution time in general is approximately proportional to the number of operations. To better understand the time-accuracy efficiency, the newly-proposed high-order direct solver, the conventional second order fast Poisson method [4], and the influence matrix method [7] are compared analytically in terms of operation counts, assuming all the three solvers are solving the same problem and to meet the same accuracy requirement.

The influence matrix method (see Appendix B) can be combined with the direct fourth-order Dirichlet solver [6] to solve Equation (1) with Neumann conditions. The analysis given in Appendix B shows that the influence matrix method has the same order of truncation error as our proposed

high-order solution. The final solution error depends on both the truncation error and the matrix of the discretized linear system, but the matrices of the two methods have eigenvalues with the same properties where solution errors are concerned. Therefore it is appropriate to conclude that the influence matrix method and the newly proposed high-order solver have the same accuracy. The operation count of the influence matrix method given by (44) is at least N times the operation count of our direct Neumann solver for problems of partition size no larger than $2^{20} \times 2^{20}$. So we can conclude that our method is approximately N times faster than the influence matrix method.

The comparison of the conventional fast Fourier method for Poisson equations needs more deliberation. For the sake of brevity, we restrict our discussion on the unit square domain $[0,1] \times [0,1]$. With slight modifications, the same analysis can be conducted for general rectangular domains. We introduce the following notations: $E(Mthd)$ denotes the difference between the exact solution and the numerical solution computed by method $Mthd$, $\varepsilon > 0$ is the required accuracy, i.e. the difference between the computed numerical solution and the exact solution must be less than or equal to ε. With these notations, now the error of our fourth-order method can be denoted by $E(order4)$, and the error of the second-order Poisson solver will be $E(order2)$. By analysis given in Appendix C for the high order direct method, the solution error in general satisfies

$$E(order4) = a \cdot h^4 \quad \text{for some problem dependent coefficient } a.$$

The error estimation for the conventional second order solution given in appendix A indicates that in general

$$E(order2) = b \cdot h^2 \quad \text{for some problem dependent coefficient } b.$$

To meet the accuracy requirement for both the order 2 fast Poisson solver and our order 4 method, the two solvers need to take different mesh sizes and partition sizes, say partition size N and mesh size h for our direct Neumann solution, and partition size N_2 and mesh size h_2 for the order 2 method. Then $E \leq \varepsilon$, implies that

$$a \cdot h^4 = \varepsilon \qquad \text{for the 4th order direct Neumann solver}$$

and

$$b \cdot h_2^2 = \varepsilon \qquad \text{for the second order method.}$$

Therefore

$$a \cdot h^4 = b \cdot h_2^2 \tag{30}$$

Since

$$h = 1/N \qquad \text{and} \qquad h_2 = 1/N_2,$$

(30) is equivalent to

$$\frac{a}{N^4} = \frac{b}{N_2^2}.$$

A relation between N and N_2 is

$$N_2 \;=\; \sqrt{\frac{b}{a}}\, N^2 \;=\; C\, N^2, \tag{31}$$

where

$$C = \sqrt{\frac{b}{a}}\,. \tag{32}$$

Thus, if our fourth order solver can satisfy the error tolerance by taking a partition of size N, then it requires the second order solver to take a partition size of CN^2 to achieve the same accuracy.

The parameter C in general could vary largely from problem to problem. For a Poisson equation with polynomial solution of degree four, the new fourth-order method gives the exact solution while the second order fast solver does not (see Table 5). In these cases, C is close to infinity, which means the second order solver will never get the same accuracy as our method no matter what partition size it takes. For Poisson equations which have only twice differentiable solutions, the number C will be close to $\frac{1}{N}$, which means the two methods have almost the same accuracy when taking the same partition size. Fortunately, for equations having at least fifth derivatives, the variation range of the coefficient C is not very large.

Obviously C depends on the matrices of the two discretized systems for the two methods and also on the truncation errors. Since the eigenvalues of the two methods are almost the same, we can mainly use their truncation errors to compare the solution errors of the two methods. From the analysis given in appendix A and appendix C, we can see that the truncation errors are influenced by many factors such as the coefficients, the derivatives of the solution of the first five orders, the number of terms, and the number λ in the equation (1). So the parameter C is heavily problem dependent. But the highest order terms of truncation error for both the fourth and second order solution clearly occur at the four corners of the computational domain. For problems whose derivatives of different orders differ within a factor of 100, and the coefficient λ is not greater than 400, substituting the ratio of a and b in (32) by the ratio of the truncation errors of the fourth and second solutions at the four corners, we estimate that C will fall into the interval of $[\frac{1}{16},\ 8]$. For "well behaved" equations, e.g. the derivatives of different orders differ within a factor of 5 and λ is moderately small (see the examples given in Tables 1 to 5 and in Tables 7 to 10), the parameter C assumes a range of $[\frac{1}{4},\ 1]$.

Let T_4 and T_2 denote the time needed by the order 4 and order 2 methods respectively to solve a problem within a given error tolerance. According to equations (40), (29), and (31),

$$T_2 : T_4 = \frac{C^2 N^4 (5 \log_2 CN^2 + 12) + 21 C^2 N}{N^2 (5 \log_2 N + 17) + 119 N}. \tag{33}$$

For equations where C ranges from $\frac{1}{4}$ to 1, the time ratio $T_2 : T_1$ will fall between $\frac{1}{10}N^2$ and $2N^2$ by formula (33). In other words, the traditional order 2 solver will take roughly $\frac{1}{10}N^2$ to $2N^2$ times more computation than that of the newly-proposed fourth order solver to reach the same accuracy.

The difference is huge. The speed of the newly proposed algorithm is unmatched by that of other existing direct solvers when a fourth order accurate solution is wanted.

5 The Modified Fast Solver for Neumann-Dirichlet Conditions

In this section we discuss the fourth order solution of problems with Neumann boundary conditions imposed on y-dimension, and Dirichlet conditions on x-dimension. By symmetry, this solution can be extended to problems with Neumann boundary conditions imposed on x-dimension, and Dirichlet conditions on y-dimension as well. These boundary conditions can be expressed mathematically as:

$$\begin{cases} \frac{dP(x,y)}{dy} = b(x,y) & \text{along y-dimension boundaries} \\ P(x,y) = f(x,y) & \text{along x-dimension boundaries} \end{cases} \tag{34}$$

For these Neumann-Dirichlet problems, we apply the compact scheme (15) at interior grid-points of the problem domain, yielding (18). Similarly, we apply the 1-D compact scheme (14) to discretize the boundary conditions of y-dimensional boundaries.

The discretized equation system now has the form

$$\mathbf{A}\ P = -\frac{3}{2}h^2 R - \frac{h^4}{8}\left(\lambda R + \triangle R\right) + U_x + 2hU_y\ + O(h^5), \tag{35}$$

where U_y, which depends on the Neumann conditions at the y-dimensional boundaries, is defined the same way as that of (23) and (24). U_x depends on the Dirichlet conditions at the x-dimensional boundaries and is given by

$$\begin{aligned} U_x^{1,j} &= \tfrac{1}{4}P^{0,j-1} + P^{0,j} + \tfrac{1}{4}P^{0,j+1}, \\ U_x^{m-1,j} &= \tfrac{1}{4}P^{m,j-1} + P^{m,j} + \tfrac{1}{4}P^{m,j+1}, \end{aligned}$$

for $j = 1, 2, ..., n-1$, and

$$\begin{aligned} U_x^{1,0} &= P^{0,0} + \tfrac{1}{2}P^{0,1}, & U_x^{m-1,0} &= P^{m,0} + \tfrac{1}{2}P^{m,1}, \\ U_x^{1,n} &= P^{0,n} + \tfrac{1}{2}P^{0,n-1}, & U_x^{m-1,n} &= P^{m,n} + \tfrac{1}{2}P^{m,n-1}. \end{aligned}$$

Equation (35) can be reduced to a sequence of independent tridiagonal systems either by Fast Cosine Transform (FCT) or by Fast Sine Transform (FST). Which transformation to use is a choice of efficiency. When $N \geq M$, FCT will lead to a better efficiency. Otherwise $M < N$, FST will be a good choice. If FCT is the choice, the vector P then is stored in the form of

$$P = (p_{1,0},\ p_{1,1},\ \cdots,\ p_{1,N},\ \cdots\ ,\ p_{M-1,0},\ p_{M-1,1},\ \cdots,\ p_{M-1,N})^T.$$

Matrix $\mathbf{A}$ in equation (35), then, is an $(M-1)(N+1)$ by $(M-1)(N+1)$ matrix given by

$$\mathbf{A} = \begin{pmatrix} A_1 & -A_2 & 0 & \cdot & 0 & 0 & 0 \\ -A_2 & A_1 & -A_2 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & -A_2 & A_1 & -A_2 \\ 0 & 0 & 0 & \cdot & 0 & -A_2 & A_1 \end{pmatrix}$$

with A_1, A_2 being the same as that in (25). The FCT used here is the same as (26) except that now $\mathbf{F}$ is of order of $(M-1)(N+1)$ by $(M-1)(N+1)$.

The FST is given by

$$\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1},$$

where $\mathbf{S}$ is an $(N+1)(M-1)$ by $(N+1)(M-1)$ block diagonal matrix with diagonal submatrix $\mathbf{S_1}$ of order $(M-1)$ by $(M-1)$ being

$$\mathbf{S}_1 = \begin{pmatrix} sin(\frac{\pi}{M}) & sin(\frac{2\pi}{M}) & \cdots & sin(\frac{(M-1)\pi}{M}) \\ sin(\frac{2\pi}{M}) & sin(\frac{4\pi}{M}) & \cdots & sin(\frac{2(M-1)\pi}{M}) \\ sin(\frac{3\pi}{M}) & sin(\frac{6\pi}{M}) & \cdots & sin(\frac{3(M-1)\pi}{M}) \\ \cdot & \cdot & \cdots & \cdot \\ sin(\frac{(M-1)\pi}{M}) & sin(\frac{2(M-1)\pi}{M}) & \cdots & sin(\frac{(M-1)^2\pi}{M}) \end{pmatrix}.$$

To apply FST, we need to rearrange equation (35) in such a way that the vector P is stored in the form of

$$P = (p_{1,0},\ p_{2,0},\ \cdots,\ p_{M-1,0},\ \cdots\ ,\ p_{1,N},\ p_{2,N},\ \cdots,\ p_{M-1,N})^T.$$

After the rearrangement, the matrix $\mathbf{A}$ in equation (35) takes the same form as equation (25) but with different submatrices, where matrix $\mathbf{A}_1$ is an $(N+1)$ by $(N+1)$ Teoplitz tridiagonal matrix,

$$\mathbf{A_1} = \begin{pmatrix} d & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & d & -1 & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & -1 & d & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & d \end{pmatrix},$$

$d = 5 + \frac{\lambda h^2+12}{8}\lambda h^2$, and matrix $\mathbf{A_2}$ is also an $(N+1)$ by $(N+1)$ Teoplitz tridiagonal matrix,

$$\mathbf{A_2} = \begin{pmatrix} 1 & \frac{1}{4} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{4} & 1 & \frac{1}{4} & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{4} & 1 \end{pmatrix}.$$

By equation (35), the solution algorithm for Neumann-Dirichlet problems consists of the fol-

lowing steps:

0) Determine whether FCT or FST should be used and store the data accordingly.

1) Compute the cosine or sine values to be used in the FCT or FST.

2) Compute the values of each entry in matrix $\mathbf{\Lambda}$.

3) Compute the vector U_x and U_y in equation (35).

4) Compute the right-hand side of equation (35).

5) Apply the inverse of FCT or FST to the right-hand side of equation (35).

6) Solve the sequence of tridiagonal systems that resulted from the fast transform.

7) Apply FCT or FST to the solution of each tridiagonal system computed in step 6) to recover P.

Step 0) takes no computation. For equations on a square domain with partition size $N \times N$, the operation count of this algorithm is almost the same as that of the Neumann-Neumann boundary problem. Its operation count is approximately

$$N^2(2\log_2 N + 6) \text{ multiplications and } N^2(3\log_2 N + 11) \text{ additions}$$
$$= N^2(5\log_2 N + 17) \text{ operations.}$$

6 Experiment Results

Theoretical analyses given in Appendix C show that the newly proposed Helmholtz solver is approximately fourth order accurate and is highly efficient. By definition, a numerical method is fourth order accurate if and only if when the number of grid points doubles the discretization error will decrease at a rate of $\left(\frac{1}{2}\right)^4$. Five Helmholtz equations with known exact solutions have been chosen as test problems to verify the analytical results and to illustrate the performance gain of the high order method. Among the five test problems, two have polynomial solutions, with one having a fractional degree. The other three are with solution of sine-exponential function, polynomial-cosine function, and two dimensional cosine function, respectively. They represent a large class of practical Helmholtz equations. Experimental tests have been conducted on a DEC Station 5000 to measure the numerical results and execution time. As listed in Tables 1 to 10, experimental results match analytical results closely. Measured performance confirms that the newly proposed algorithm is highly accurate and efficient.

Two performance metrics are used in the measurement. The metric **Order**, defined by

$$\text{Order(n, n+1)} = \log_2 \frac{MaxE(n)}{MaxE(n+1)},$$

measures the order of accuracy of numerical solutions. The term **Relative error** is defined as

$$Relative\ error = \frac{\|P - \widehat{P}\|_1}{\|P\|_1},$$

Table 1. Solving $P_{xx} + P_{yy} - 11P = R$ on Unit Square with $P = e^x sin(y)$

Fourth Order							
N =	8	16	32	64	128	256	512
Maximal error	1.10E-04	7.30E-06	4.67E-07	2.95E-08	1.85E-09	1.16E-10	9.40E-12
Relative error	4.22E-05	2.57E-06	1.58E-07	9.81E-09	6.11E-10	3.80E-11	3.22E-12
Order		3.9	4.0	4.0	4.0	4.0	3.6
Time(seconds)	0.004	0.016	0.066	0.27	1.20	5.3	24
Second Order							
N =	8	16	32	64	128	256	512
Maximal error	1.94E-03	4.94E-04	1.24E-04	3.10E-05	7.76E-06	1.94E-06	4.85E-07
Relative error	7.22E-04	1.69E-04	4.06E-05	9.93E-06	2.46E-06	6.11E-07	1.52E-07
Order		2.0	2.0	2.0	2.0	2.0	2.0
Time(seconds)	0.004	0.012	0.066	0.26	1.13	5.2	24

Table 2. Solving $P_{xx} + P_{yy} = R$ on Unit Square with $P = cos(xy)$

Fourth Order							
N =	8	16	32	64	128	256	512
Maximal error	5.26E-05	2.90E-06	1.66E-07	9.88E-09	5.99E-10	3.70E-11	1.49E-12
Relative error	9.32E-06	5.00E-07	2.95E-08	1.79E-09	1.11E-10	6.91E-12	1.71E-13
Order		4.2	4.1	4.1	4.0	4.0	4.7
Time(seconds)	0.004	0.012	0.062	0.28	1.20	5.18	23
Second Order							
N =	8	16	32	64	128	256	512
Maximal error	8.15E-04	2.01E-04	5.00E-05	1.25E-05	3.12E-06	7.80E-07	1.95E-07
Relative error	1.11E-04	2.53E-05	6.05E-06	1.48E-06	3.67E-07	9.12E-08	2.27E-08
Order		2.0	2.0	2.0	2.0	2.0	2.0
Time(seconds)	0.004	0.016	0.062	0.28	1.13	5.04	23

where P and $\widehat{P}$ denote the exact solution and computed solution respectively, and $\|\cdot\|_1$ is the l_1 norm. All the testing is conducted on the unit square domain $[0,1] \times [0,1]$ with the same uniform mesh size h on each dimension. $N = 1/h$ is the number of grid points on each x- and y-dimension.

Tables 1 to 4 present the time-accurate comparison between the new fourth-order fast solver and the traditional second-order fast solver (see Appendix A), for the first four testing problems with Neumann-Neumann boundary conditions. Measured experimental results show our new method is indeed fourth order accurate and achieves the high order accurate solution without increasing execution time, as compared with the conventional second-order fast solver.

Table 5 lists measured experimental results for a special Poisson equation whose solution is a polynomial of degree four. The truncation error given in Appendix C has only derivatives of fifth order or higher when $\lambda = 0$, which implies that our high order solver for the Poisson equation

Table 3. Solving $P_{xx} + P_{yy} - P = R$ on Unit Square with $P = (x^3 - x^2)cosy$

Fourth Order							
N =	8	16	32	64	128	256	512
Maximal error	7.73E-05	4.90E-06	3.07E-07	1.92E-08	1.20E-09	7.50E-11	4.64E-12
Relative error	5.72E-04	3.12E-05	1.83E-06	1.10E-07	6.79E-09	4.21E-10	2.63E-11
Order		4.0	4.0	4.0	4.0	4.0	4.0
Time(seconds)	0.008	0.016	0.086	0.269	1.24	5.30	24
Second Order							
N =	8	16	32	64	128	256	512
Maximal error	6.86E-03	1.72E-03	4.31E-04	1.08E-04	2.69E-05	6.73E-06	1.68E-06
Relative error	5.46E-02	1.21E-02	2.83E-03	6.85E-04	1.68E-04	4.18E-05	1.04E-05
Order		2.0	2.0	2.0	2.0	2.0	2.0
Time(seconds)	0.004	0.012	0.059	0.262	1.14	5.02	24

will give an exact solution for polynomials of degree four or lower. This implication is verified by the testing results. The table shows that the fourth-order direct Neumann solver gives the exact solution while the second order method cannot reach high accuracy even with enlarged problem size and with extended execution time.

Table 6 compares the measured execution time of the conventional second-order faster Poisson solver and the newly-proposed fourth-order solver. The first four testing problems are solved by the fourth-order algorithm. Then, the same problems are solved with the conventional second order method to match the achieved accuracy with increased number of grid points and execution time. The execution times of the fourth order algorithm and the second order algorithm are listed side-by-side in Table 6 for each of the testing problems. Table 6 shows that the new method is 300 to 1500 times faster, as indicated by the column of time ratio for the two solvers. Notice that the performance gain increases largely when the problem size increase with the problem domain. This time ratio increase is no surprise. It is around $N^2/4$ and is well predicted by the range $[\frac{1}{10}N^2, 2N^2]$ given in Section 4. The new algorithm is well suitable for scalable computing where problem size increases with the computational power.

Tables 7 to 10 list the experimental results of the first four testing problems with corresponding Neumann-Dirichlet boundary conditions. As confirmed by the measured results, solutions of Neumann-Dirichlet problems are also of fourth order accurate. In addition, they even have a smaller error than that of the Neumann-Neumann boundary problem, since there is no discretization error arising along x-dimensional Dirichlet boundary conditions. This feature is also well matched by our experiment results.

Table 4. Solving $P_{xx} + P_{yy} - 24.75P = R$ on Unit Square with $P = x^{5.5} + y^{5.5}$

Fourth Order							
N =	16	32	64	128	256	512	1024
Maximal error	4.75E-04	3.46E-05	2.32E-06	1.50E-07	9.55E-09	6.01E-10	3.84E-11
Relative error	3.45E-04	2.50E-05	1.67E-06	1.07E-07	6.81E-09	4.28E-10	2.67E-11
Order		3.8	3.9	4.0	4.0	4.0	4.0
Time(seconds)	0.016	0.059	0.285	1.24	5.30	23	102
Second Order							
N =	16	32	64	128	256	512	1024
Maximal error	1.82E-02	4.61E-03	1.15E-03	2.89E-04	7.22E-05	1.81E-05	4.51E-06
Relative error	8.34E-03	2.09E-03	5.20E-04	1.30E-04	3.24E-05	8.09E-06	2.02E-06
Order		2.0	2.0	2.0	2.0	2.0	2.0
Time(seconds)	0.016	0.055	0.281	1.13	5.05	25	97

Table 5. Solving $P_{xx} + P_{yy} = R$ on Unit Square with $P = x^4 + y^4$

Method	Fourth Order	Second Order	Second Order
N =	8	256	1024
Maximal error	6.66E-16	3.05E-05	1.91E-06
Relative error	1.09E-15	2.53E-05	1.59E-06
Time(seconds)	0.004	5.04	100

7 Conclusion

Solving Helmholtz equations is a fundamental problem of scientific computing. Since Hockney first proposed the so-called "fast Poisson solver" in 1965, intensive research has been done in the field to develop fast direct solvers. However, while significant progress has been made during the years, fourth-order fast solvers are only currently available for Helmholtz equations with Dirichlet boundary conditions. Based on a novel compact finite-difference scheme, a fourth-order fast direct solver is proposed for Helmholtz equations with Neumann-Neumann and Neumann-Dirichlet boundary conditions on a rectangular domain in this study. Accuracy and efficiency of the fourth order algorithm are carefully examined. Theoretical and experimental results show that the newly proposed algorithm is highly efficient. It can obtain fourth order solution as fast as the conventional method achieving second order solutions, and could be thousands of times faster than that of the conventional method if accurate solutions are required.

The search for a better parallel solver is the original motivation behind this research. The proposed algorithm is parallel in nature. Although parallel implementation is not presented in this study, since the algorithm has a similar data structure as that of the conventional direct method, its parallelization is straight forward and high speedup is expected [13, 14]. The same uniform mesh

Table 6. Computation time of the two methods for the same accuracy

Problem	Method	N	Maximal error	Time(seconds)	Time ratio
1	Order 4	32	4.67E-07	0.066	
	Order 2	512	4.85E-07	24	364
1	Order 4	64	2.95E-08	0.27	
	Order 2	2048	3.03E-08	423	1570
2	Order 4	32	1.66E-07	0.062	
	Order 2	512	1.95E-07	24	389
2	Order 4	64	9.88E-09	0.28	
	Order 2	2048	1.22E-08	423	1510
3	Order 4	16	4.90E-06	0.016	
	Order 2	256	6.73E-06	5.02	314
3	Order 4	32	3.07E-07	0.062	
	Order 2	1024	4.21E-07	97	1560
4	Order 4	32	3.46E-05	0.059	
	Order 2	512	1.81E-05	25	424
4	Order 4	64	2.32E-06	0.285	
	Order 2	2048	1.13E-06	422	1480

Table 7. Neumann-Dirichlet problem: $P_{xx} + P_{yy} - 11P = R$ on unit square with $P = e^x sin(y)$

By FCT							
N =	8	16	32	64	128	256	512
Maximal error	6.16E-05	4.02E-06	2.54E-07	1.59E-08	9.97E-10	6.25E-11	5.17E-12
Relative error	2.28E-05	1.21E-06	6.83E-08	4.04E-09	2.46E-10	1.51E-11	1.65E-12
Order		3.9	4.0	4.0	4.0	4.0	3.6
Time(seconds)	0.004	0.016	0.066	0.27	1.20	5.3	24

size is used in both x- and y-dimension in our discussion. The restriction of the same uniform mesh size is for the sack of brevity. Different uniform mesh sizes can be applied to x- and y-dimension respectively. Similar to the conventional second order solver, Fourier transform is used in the newly introduced method to hasten computation. As observed by Hockney, the Fourier transform based approach is a special case of the FACR(l) method with $l = 0$ [5]. By combining l steps of cyclic reduction with the newly proposed algorithm, an FACR(l)-like algorithm could be formed to further improve current results for optimal solution. Also, with appropriate modifications, the fourth order algorithm is likely extendible to six order solutions and to more general boundary and domain conditions. Many issues have opened for future work.

Table 8. Neumann-Dirichlet problem: $P_{xx} + P_{yy} = R$ on unit square with $P = cos(xy)$

By FCT							
N =	8	16	32	64	128	256	512
Maximal error	1.08E-05	7.59E-07	4.97E-08	3.17E-09	2.00E-10	1.26E-11	5.39E-13
Relative error	2.10E-06	1.10E-07	6.34E-09	3.85E-10	2.38E-11	1.50E-12	7.38E-14
Order		3.8	3.9	4.0	4.0	4.0	4.5
Time(seconds)	0.004	0.019	0.066	0.26	1.23	6.1	23.7

Table 9. Neumann-Dirichlet problem: $P_{xx} + P_{yy} - P = R$ on unit square with $P = (x^3 - x^2)cosy$

By FST							
N =	4	8	16	32	64	128	256
Maximal error	6.98E-05	5.98E-06	4.16E-07	2.72E-08	1.73E-09	1.09E-10	6.89E-12
Relative error	2.23E-04	1.34E-05	8.25E-07	5.20E-08	3.30E-09	2.08E-10	1.40E-11
Order		3.5	3.8	3.9	4.0	4.0	4.0
Time(seconds)	0.001	0.008	0.016	0.086	0.269	1.24	5.30

Appendices

A Direct Second Order Solution

Using traditional finite-difference scheme, the second order approximation of the Laplacian is given by

$$h^{-2} \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix} P^{i,j} = \triangle P^{i,j} + O(h^2), \tag{36}$$

and the discretization of the Neumann boundary conditions is given by

$$\frac{P^{-1,j} - P^{1,j}}{2h} = P_x^{0,j} + O(h^2). \tag{37}$$

Table 10. Neumann-Dirichlet problem: $P_{xx} + P_{yy} - 24.75P = R$ on unit square with $P = x^{5.5} + y^{5.5}$

By FST							
N =	8	16	32	64	128	256	512
Maximal error	2.54E-03	2.26E-04	1.65E-05	1.10E-06	7.13E-08	4.53E-09	2.85E-10
Relative error	2.02E-03	1.46E-04	9.46E-06	5.96E-07	3.73E-08	2.33E-09	1.44E-10
Order		3.5	3.8	3.9	3.9	4.0	4.0
Time(seconds)	0.004	0.016	0.059	0.285	1.24	5.3	22

Applying (36) and (37) to discretize equation (1) and boundary condition (19) respectively, yielding

$$\mathbf{D}\ P = -h^2\ R + 2hU + O(h^3)\ , \tag{38}$$

where the vector U stores the linear combination of the function values of $b(x, y)$ in (19) at different boundary points, and matrix $\mathbf{D}$ has the same structure as matrix $\mathbf{A}$ given by (25) except that the submatrix $\mathbf{A_2}$ is the $N \times N$ identity matrix, and $\mathbf{A_1}$ is given by

$$\mathbf{A_1} = \begin{pmatrix} 4+\lambda h^2 & -2 & 0 & \cdot & 0 & 0 & 0 \\ -1 & 4+\lambda h^2 & -1 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & -1 & 4+\lambda h^2 & -1 \\ 0 & 0 & 0 & \cdot & 0 & -2 & 4+\lambda h^2 \end{pmatrix}.$$

The truncation error of (38) is

$$\begin{aligned} T_2(i,j) &= -\tfrac{h^4}{12}\left(\tfrac{\partial^4}{\partial x^4} + \tfrac{\partial^4}{\partial y^4}\right) P^{i,j} && \text{for interior point } (i,j), \\ T_2(0,j) &= -\tfrac{h^4}{12}\left(\tfrac{\partial^4}{\partial x^4} + \tfrac{\partial^4}{\partial y^4}\right) P^{0,j} - \tfrac{h^3}{3} P^{0,j}_{xxx} && \text{for boundary point } (0,j), \\ T_2(0,0) &= -\tfrac{h^4}{12}\left(\tfrac{\partial^4}{\partial x^4} + \tfrac{\partial^4}{\partial y^4}\right) P^{0,0} - \tfrac{h^3}{3}\left(\tfrac{\partial^3}{\partial x^3} + \tfrac{\partial^3}{\partial y^3}\right) P^{0,0} && \text{for boundary point } (0,0). \end{aligned}$$

Since the solution error depends not only on the truncation error, but also on the matrix of the discretized system, a study of matrix $\mathbf{D}$ is needed.

Matrix $\mathbf{D}$ has the same eigenvectors as that of matrix $\mathbf{A}$ in equation (25). Eigenvalues of matrix $\mathbf{D}$ are

$$d_{k,l} = 4 + \lambda h^2 - 2cos(\frac{k\pi}{M}) - 2cos(\frac{l\pi}{N})\ , \tag{39}$$

for $k = 0, 1, ..., M$ and $l = 0, 1, ..., N$. Compared with the eigenvalues of matrix $\mathbf{A}$ given in (46), we can see that

$$d_{k,l} = \lambda_{k,l} - \left(1 + \frac{\lambda h^2 + 4}{8}\lambda h^2 - cos(\frac{k\pi}{M})cos(\frac{l\pi}{N})\right) < \lambda_{k,l}\ .$$

Like the eigenvalues of matrix $\mathbf{A}$, $d_{k,l}$ are all positive, and satisfy the monotone property

$$d_{k-1,l} < d_{k,l}, \quad \text{and} \quad d_{k,l-1} < d_{k,l},$$

and $d_{0,0} \in O(h^2)$ and $d_{k,l} \in O(1)$ for (k, l) such that $k \geq \frac{M}{10}$ or $l \geq \frac{N}{10}$. Following a similar analysis to that of the fourth order method given in appendix C, we can conclude that the solution error of the second order method ranges from $O(h)$ to $O(h^3)$ when considering all possible extreme situations, and in general, will be around $O(h^2)$.

The conventional second-order direct solver, which is first proposed by Hockney [4] and modified and extended by many since then [12], is a Fourier transform based algorithm. It goes through the same seven steps as the fourth order solution listed in Section 4, but with different computations in steps 2), 3) and 4) due to the difference in discretization. The conventional second order method

has $3N$ multiplications and N additions for step 2), $4N$ multiplications and $4N$ additions for step 3), and $4N$ additions for step 4). Therefore, the total operation count of the second order fast solver is

$$\begin{aligned} &N^2(2\log_2 N+5)+N(2\log_2 N+7) \text{ multiplications and} \\ &N^2(3\log_2 N+7)+N(-2\log_2 N+14) \text{ additions} \\ &=N^2(5\log_2 N+12)+21N \quad \text{operations.} \end{aligned} \tag{40}$$

B The Influence Matrix Method

A Poisson-Neumann problem can be solved by solving two Poisson-Dirichlet problems through the influence matrix method. Here we briefly describe the influence matrix method based on [7].

To solve the Neumann problem (1) with boundary condition $\frac{dP}{dn}=0$, a sequence of solutions to the following problem is first determined:

$$\begin{aligned} &\triangle q^i-\lambda q^i=0 \\ &q^i=\delta_{i,j} \end{aligned} \tag{41}$$

for each discrete boundary grid point $\bar{x}_j$. So there are $4N$ equations on a square of partition size $N\times N$. The Dirac delta function $\delta^{i,j}$ is defined as $\delta^{i,j}=1$ for $i=j$ and $\delta^{i,j}=0$ for $i\neq j$. Upon computation of the vectors of normal gradients $\frac{dq^i}{dn}$ at all the boundary points, these vectors are then stored in columns to yield a matrix $\mathbf{I}_{NF}$ that is referred to as the influence matrix.

The Neumann problem then is equivalent to the following solution of two Dirichlet problems. First, solve

$$\begin{aligned} &\triangle P_1-\lambda P_1=R \quad \text{in } \Omega \\ &P_1=0 \qquad \text{on } \partial\Omega\ . \end{aligned} \tag{42}$$

Again, compute the gradients normal to the boundary and store them in vector G. Then, solve

$$\begin{aligned} &\triangle P_2-\lambda P_2=0 \quad \text{in } \Omega \\ &P_2=\mathbf{I}_{NF}^{-1}\,G \qquad \text{on } \partial\Omega\ . \end{aligned} \tag{43}$$

The final solution that satisfies the original equation (1) and boundary condition (19) is P_1-P_2.

The influence matrix method can be combined with the direct fourth-order Dirichlet solver [6] to solve Equation (1) with Neumann conditions. Solving (41) and (42) by the fourth order method of [6] results in two linear systems with truncation error of order $O(h^6)$, with a same symmetric matrix with eigenvectors

$$S_{k,l}=\left(S_{1,1}^{k,l},S_{1,2}^{k,l},...,S_{1,N-1}^{k,l}\ ;\ ...\ ;\ S_{M-1,1}^{k,l},S_{M-1,2}^{k,l},...,S_{M-1,N-1}^{k,l}\right)^T,$$

where $S_{i,j}^{k,l}=sin(\frac{ik\pi}{M})sin(\frac{jl\pi}{N})$ with respective eigenvalues $\lambda_{k,l}$ being the same as the eigenvalues given by (46, for $k=1,2,...,M-1$, and $l=1,2,...,N-1$.

Note that the eigenvalues $\lambda_{k,l}$ are the same as those of the proposed high order direct solver except for $k=0,M$ or $l=0,N$. We also have that $\lambda_{1,1}$ is of $O(h^2)$, and $\lambda_{M-1,N-1}$ is of $O(1)$. So

based on the error analysis given in appendix C, solving (41) and (42) results in solutions with error ranging from $O(h^4)$ to $O(h^6)$. Suppose they achieve $O(h^6)$. Then after computing the influence matrix $\mathbf{I}_{NF}$ and the vector G, the truncation error of (43) increases to at least $O(h^5)$, because first order differentiation reduces the order by one. Also the discrete system of (43) has the same matrix as (41) and (42), which in the sense of l_2 norm is not better than matrix $\mathbf{A}$ of (22).

The operation count of the influence matrix method is mainly due to the computation of the influence matrix, which requires solving a sequence of $4N$ Helmholtz-Dirichlet equations given by (41) on a square domain of size $N \times N$. To solve each equation in the sequence (41) using cyclic reduction and Fourier analysis based method has an asymptotic operation count of $3N^2 \log_2(\log_2 N)$ multiplications and $3N^2 \log_2(\log_2 N)$ additions as given in the review paper [13]. Therefore the computation count of the influence matrix is

$$12N^3 \log_2(\log_2 N) \text{ multiplications and } \; 12N^3 \log_2(\log_2 N) \text{ additions.} \tag{44}$$

C Error Estimation of the Fourth-Order Neumann Solver

In this section, we give the error analysis of the newly proposed high-order compact finite-difference discretization. First we derive the truncation error of the discretized linear system, and then an eigenvalue analysis of the matrix is presented, and finally we give a global solution error estimation by using the eigenvalue properties of the matrix and the truncation error derived.

The truncation error T_x of the boundary condition (21) at the boundaries of x-dimension when all differential operators replaced by their respective discrete counterparts is:

$$\begin{aligned} T_x(0,j) &= \tfrac{h^4}{120}\tfrac{\partial^5}{\partial x^5}P^{0,j} + \tfrac{h^4}{18}\tfrac{\partial^3}{\partial x^3}\left(\triangle P^{0,j} + \lambda P^{0,j}\right) + \tfrac{h^4}{72}\tfrac{\partial^5}{\partial x \partial y^4}P^{0,j}, \\ T_x(0,0) &= \tfrac{h^4}{120}\tfrac{\partial^5}{\partial x^5}P^{0,0} + \tfrac{h^4}{18}\tfrac{\partial^3}{\partial x^3}\left(\triangle P^{0,0} + \lambda P^{0,0}\right) + \tfrac{11h^4}{72}\tfrac{\partial^5}{\partial x \partial y^4}P^{0,0}. \end{aligned}$$

The truncation error T_y of the boundary condition at the boundaries of y-dimension has similar formula as T_x, and is therefore omitted.

To eliminate outside point $(-1,-1)$ of (18) at point $(0,0)$, both the x- and y- dimensional boundary conditions are needed. And the truncation error of both the x- and y- dimensional boundary condition at point $(0,0)$ is given by

$$T_{xy}(0,0) = \left[\tfrac{h^4}{120}\left(\tfrac{\partial}{\partial x} + \tfrac{\partial}{\partial x}\right)^5 + \tfrac{h^4}{18}\left(\tfrac{\partial^3}{\partial x^3} + \tfrac{\partial^3}{\partial y^3}\right)(\triangle + \lambda) + \tfrac{11h^4}{36}\left(\tfrac{\partial^5}{\partial x^4 \partial y} + \tfrac{\partial^5}{\partial x \partial y^4}\right)\right] P^{0,0}.$$

After substituting the Laplacian in (18) by its discrete version, the truncation error for (18) is

given by

$$\begin{aligned}
T_1(i,j) &= h^4\left[\tfrac{1}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{1}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})\right]P^{i,j}\\
&+ \tfrac{h^4}{96}\left(\tfrac{\partial^4}{\partial y^4}+\tfrac{\partial^4}{\partial x^4}\right)(\triangle P^{i,j}+\lambda P^{i,j}),\\
T_1(0,j) &= h^4\left[\tfrac{1}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{1}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})\right]P^{0,j}\\
&+ \tfrac{h^4}{96}\left(\tfrac{\partial^4}{\partial y^4}+11\tfrac{\partial^4}{\partial x^4}\right)(\triangle P^{0,j}+\lambda P^{0,j}),\\
T_1(0,0) &= h^4\left[\tfrac{1}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{1}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})\right]P^{0,0}\\
&+ \tfrac{11h^4}{96}\left(\tfrac{\partial^4}{\partial y^4}+\tfrac{\partial^4}{\partial x^4}\right)(\triangle P^{0,0}+\lambda P^{0,0}).
\end{aligned}$$

So the truncation error of (22) when all differential operators replaced by their respective discrete versions is

$$\begin{aligned}
T(i,j) &= h^2T_1(i,j),\\
T(0,j) &= h^2T_1(0,j)-2hT_x(0,j)-\tfrac{h}{2}\left(T_x(0,j-1)+T_x(0,j+1)\right),\\
T(0,0) &= h^2T_1(0,0)-2h\left(T_x(0,0)+T_y(0,0)\right)\tfrac{h}{2}\left(T_x(0,1)+T_y(1,0)+T_{xy}(0,0)\right),
\end{aligned}$$

or

$$\begin{aligned}
T(i,j) &= \left[\tfrac{h^6}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{h^6}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})+\tfrac{h^6}{96}\left(\tfrac{\partial^4}{\partial y^4}+\tfrac{\partial^4}{\partial x^4}\right)(\triangle+\lambda)\right]P^{i,j},\\
T(0,j) &= \left[\tfrac{h^6}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{h^6}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})+\tfrac{h^6}{96}(\tfrac{\partial^4}{\partial y^4}+11\tfrac{\partial^4}{\partial x^4})(\triangle+\lambda)\right]P^{0,j}\\
&- h^5\left[\tfrac{1}{40}\tfrac{\partial^5}{\partial x^5}+\tfrac{1}{6}\tfrac{\partial^3}{\partial x^3}(\triangle+\lambda)+\tfrac{1}{24}\tfrac{\partial^5}{\partial x\partial y^4}\right]P^{0,j},
\end{aligned}$$

$$\begin{aligned}
T(0,0) = {} & h^6\left[\tfrac{1}{144}(\tfrac{\partial^6}{\partial x^4\partial y^2}+\tfrac{\partial^6}{\partial x^2\partial y^4})-\tfrac{1}{240}(\tfrac{\partial^6}{\partial x^6}+\tfrac{\partial^6}{\partial y^6})+\tfrac{11}{96}\left(\tfrac{\partial^4}{\partial y^4}+\tfrac{\partial^4}{\partial x^4}\right)(\triangle+\lambda)\right]P^{0,0}\\
& -h^5\left[\tfrac{1}{240}(\tfrac{\partial}{\partial x}+\tfrac{\partial}{\partial y})^5+\tfrac{1}{60}(\tfrac{\partial^5}{\partial x^5}+\tfrac{\partial^5}{\partial y^5})+\tfrac{5}{36}(\tfrac{\partial^3}{\partial x^3}+\tfrac{\partial^3}{\partial y^3})(\triangle+\lambda)+\tfrac{11}{24}(\tfrac{\partial^5}{\partial x\partial y^4}+\tfrac{\partial^5}{\partial x^4\partial y})\right]P^{0,0}\\
& -h^5\left(\tfrac{1}{240}\tfrac{\partial^5}{\partial x^5}+\tfrac{1}{36}\tfrac{\partial^3}{\partial x^3}(\triangle+\lambda)+\tfrac{1}{144}\tfrac{\partial^5}{\partial x\partial y^4}\right)P^{0,1}\\
& -h^5\left(\tfrac{1}{240}\tfrac{\partial^5}{\partial y^5}+\tfrac{1}{36}\tfrac{\partial^3}{\partial y^3}(\triangle+\lambda)+\tfrac{1}{144}\tfrac{\partial^5}{\partial x^4\partial y}\right)P^{1,0}.
\end{aligned}$$

The solution error E, i.e. the difference between the exact solution and the computed solution, is not only dependent on the truncation error T but also on the matrix $\mathbf{A}$. In fact,

$$E = \mathbf{A}^{-1}T. \tag{45}$$

To see how the matrix $\mathbf{A}$ influences the solution error E, we look at its eigenvectors and eigenvalues first.

Matrix $\mathbf{A}$ has eigenvectors

$$V'_{k,l} = \left(V^{k,l}_{0,0}, V^{k,l}_{0,1}, \ \dots\ , V^{k,l}_{0,N}\ ;\ \dots\ ;\ V^{k,l}_{M,0}, V^{k,l}_{M,1}, \dots, V^{k,l}_{M,N}\right)^T,$$

where $V^{k,l}_{i,j} = cos(\frac{ik\pi}{M})cos(\frac{jl\pi}{N})$, with respective eigenvalues

$$\lambda_{k,l} = d - 2cos(\frac{k\pi}{M}) - 2cos(\frac{l\pi}{N}) - cos(\frac{k\pi}{M})cos(\frac{l\pi}{N}) \tag{46}$$

for $k = 0, 1, \dots, M$, and $l = 0, 1, \dots, N$, and $d = 5 + \frac{\lambda h^2+12}{8}\lambda h^2$.

Matrix $\mathbf{A}^{-1}$ has the same eigenvectors as $\mathbf{A}$ with corresponding eigenvalues $1/\lambda_{kl}$. Since there are $(M+1)(N+1)$ distinct eigenvalues in the $(M+1)(N+1)$ dimensional discrete space, the $(M+1)(N+1)$ eigenvectors are linear independent and thus span the $(M+1)(N+1)$ dimensional vector space in which we are solving the equation. Therefore the truncation error T can be expended in terms of the spanning eigenvectors V_{kl} normalized from V'_{kl} for $k = 0, 1, ..., M$ and $l = 0, 1, ..., N$ as

$$T = \sum_{k=0}^{M} \sum_{l=0}^{N} C_{kl} V_{kl} \ . \tag{47}$$

Thus by (45)

$$\|E\|_2 = \|\mathbf{A}^{-1} T\|_2 = \sqrt{\sum_{k=0}^{M} \sum_{l=0}^{N} \frac{C_{kl}^2}{\lambda_{kl}^2}}. \tag{48}$$

Since in any finite dimensional space l_2 norm $\|\cdot\|_2$ and infinity norm $\|\cdot\|$ are equivalent, we use (48) to estimate the error of the proposed high order direct solution. Since the truncation error T is of $O(h^5)$, (47) means that $\sum_{k,l} C_{kl} V_{kl}$ is of $O(h^5)$. The eigenvalues of $\mathbf{A}$ are all positive and satisfy

$$\lambda_{k-1,l} < \lambda_{k,l}, \quad \text{and} \quad \lambda_{k,l-1} < \lambda_{k,l}.$$

Therefore

$$\sqrt{\sum_{k=0}^{M} \sum_{l=0}^{N} \frac{C_{kl}^2}{\lambda_{M,N}^2}} \leq \|E\|_2 \leq \sqrt{\sum_{k=0}^{M} \sum_{l=0}^{N} \frac{C_{kl}^2}{\lambda_{0,0}^2}}.$$

But $\lambda_{0,0}$ is of $O(h^2)$, $\lambda_{0,\sqrt{N}}$ and $\lambda_{\sqrt{M},0}$ are of $O(h)$, and $\lambda_{k,l}$ is of $O(1)$, for all (k,l) pairs such that $k \geq \frac{M}{10}$ or $l \geq \frac{N}{10}$, which implies that the order of E ranges from $O(h^3)$ to $O(h^5)$. For the solution to be of $O(h^3)$, the truncation error T must concentrate on the near-zero low frequence (i.e. all coefficients $C_{k,l}$ are almost equal to 0 for k, l not near zero) in the expansion (47). Assuming uniform distribution for the coefficients $C_{k,l}$'s, the probability of E being $O(h^3)$ is close to zero when M and N are large. In general, this high order method is approximately fourth order.

References

[1] Buneman, O. A compact non-iterative poisson solver. Report 294, 1969.

[2] Collatz, L. *The numerical treatment of differential equations.* Springer, 1960.

[3] Hirsh, R. Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique. *J. Comput. Phys. 19* (1975), 90–109.

[4] Hockney, R. A fast direct solution of Poisson's equation using Fourier analysis. *J. ACM 12* (1965), 95–113.

[5] Hockney, R. The potential calculation and some applications. *Meth. in Comput. Phys. 9* (1970), 135–211.

[6] Houstis, E. N., and Papatheodorou, T. S. High-order fast elliptic equation solver. *ACM Trans. on Math. Software 5*, 4 (Dec. 1979), 431 – 441.

[7] Joslin, R., Streett, C., and Chang, C.-L. Validation of three-dimensional incompressible spatial direct numerical simulation code. NASA Technical Report, TP-3025, NASA Langley Research Center, July 1992.

[8] Kaufman, L., and Warner, D. High-order, fast-direct methods for separable elliptic equations. *SIAM J. Numer. Anal. 21*, 4 (1984), 672 – 694.

[9] Kaufman, L., and Warner, D. A program for solving separable elliptic equations. *ACM Trans. Math. Software 16*, 4 (1990), 325 – 351.

[10] Kreiss, H., and Oliger, J. Methods for the approximate solution of time dependent problems. GARP Report No 10, 1973.

[11] Lele, S. Compact finite difference schemes with spectral-like resolution. *J. Comp. Phys. 103*, 1 (1992), 16–42.

[12] Strang, G. *Introduction to Applied Mathematics.* Wellesley-Combridge Press, 1986.

[13] Sun, X.-H. Application and accuracy of the parallel diagonal dominant algorithm. *Parallel Computing* (Aug. 1995), 1241–1267.

[14] Sun, X.-H., and Joslin, R. A simple parallel prefix algorithm for almost Toeplitz tridiagonal systems. *International Journal of High Speed Computing* (Dec. 1995).

[15] Sun, X.-H., and Rover, D. Scalability of parallel algorithm-machine combinations. *IEEE Transactions on Parallel and Distributed Systems* (June 1994), 599–613.

[16] Swarztrauber, P. N. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review 19*, 3 (1977), 490–501.