Fast Poisson Solvers

Oluwapelumi Stephen Awolude (oluwapelumi@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Nick Hale
Stellenbosch University, South Africa

23 May 2019

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

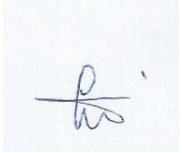
Abstract

The Poisson equation is an elliptic partial differential equation with many applications in physics and engineering. In real-world applications, obtaining the exact solution can be difficult and this leads us to employing numerical methods to obtain an approximate solution. We desire the approximate solution to be accurate, and the method of computing it to be efficient. In this essay, we consider two procedures for discretizating the Poisson equation on a square domain. These procedures include the central difference and the recently developed spectral methods with quasi-optimal complexity, i.e., $\mathcal{O}(n^2 P(\log(n)))$. There are several methods for solving the resulting linear equations some of which include LU or Cholesky factorization of the Kronecker form, eigendecomposition, fast Fourier transform, and alternating direction implicit. Numerical experiments are presented for all the methods considered and computations were implemented using Python 3.

Keywords: finite difference method, alternating direction implicit method, fast Fourier transform, spectral methods, ultraspherical polynomials.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.



Oluwapelumi Stephen Awolude, 23 May 2019

Contents

1. Introduction

The Poisson equation

$$\nabla^2 u = f, \tag{1.0.1}$$

is a second order elliptic partial differential equation (Burden and Faires, 2011; Iserles, 2009), and in two dimensions, the Laplacian operator is

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

Typically, the forcing function $f(x, y)$ is given and we seek the solution $u(x, y)$ in a specified domain Ω. If the function $f(x, y)$ is a zero, i.e., $\nabla^2 u = 0$, this is known as the Laplace equation (Chapra and Canales, 2010; Randall J, 2007). Typical boundary conditions for the Poisson equation are either Dirichlet or Neumann, where the value of $f(x, y)$ is defined on the boundary or the value of the normal derivative is defined on the boundary. In other words, for us to safeguard uniqueness of the solution, we impose boundary conditions which are described as

$$u = v_d \quad \text{or} \quad \frac{\partial u}{\partial m} \equiv \hat{m} \cdot \nabla u = v_M \quad \text{on} \quad \partial\Omega. \tag{1.0.2}$$

Both the Laplace and Poisson equations have a broad area of applications in engineering and physics (Fuka, 2015). The Poisson equation (1.0.1) can be applied in the evaluation of gravitational and electrostatic potentials. This is done by conducting pressure correction when finding solution to the incompressible Navier-Stokes equations (Fuka, 2015). We also encounter the Poisson equation in the study of isotropic bodies with internal forces whose steady-state temperature is to be determined (Feynman et al., 2005). The Poisson equation is frequently solved in physics problems relating to nano- and macro-electronic devices (Datta, 2005), electronic transport, and electro-chemistry problems in form of the Poisson–Boltzmann equation (Fogolari et al., 2002).

Like many other PDEs, exact solutions are difficult to get and it is therefore imperative for us to find fast numerical methods to obtain accurate approximations to the solutions of (1.0.1) and (1.0.2).

In order to obtain the solution of (1.0.1) using a numerical approach, it is necessary that we discretize (1.0.1). One way of doing this is by employing the finite difference approximation (FD). FD is used to obtain a discrete approximation to a function that satisfies a particular connection linking function values and derivatives on some specified domain of time or space, alongside conditions on the boundary of this domain. This method is applied by replacing the derivatives in the differential equations with FD approximations. After discretizing the Poisson equation (1.0.1) using the FD, one arrives at a system of algebraic linear equations, which must be solved.

FD methods are often favoured for simplicity, but are known to have slow convergence (Fletcher, 1984). In order to attain fast convergence and high accuracy, spectral methods are used. However, these typically lead to dense matrices that are computationally expensive to invert. However, following Fortunato and Townsend (2017), we shall derive a fast spectral method for the Poisson equation.

1.1 Aim and Objectives

The aim of this project is to use numerical methods to solve the Poisson equation (1.0.1) on a square domain with zero homogeneous Dirichlet boundary conditions. In particular, we will focus on the finite

difference and spectral methods. To solve the resulting linear systems, we will consider methods such as the linear system of the Kronecker form, eigendecomposition and fast Fourier transforms (FFT), alternating direction implicit (ADI) and the fast spectral method with quasi-optimal complexity, i.e., $\mathcal{O}(n^2 P(\log(n)))$.

1.2 Outline

Chapter 2 is devoted to discussing finite difference methods and how they can be used in discretizing the 2D Poisson equation (1.0.1). Following this, the Kronecker, eigendecomposition and FFT approaches for solving (1.0.1) are discussed, and we perform numerical experiments using these methods. Chapter 3 focuses on the alternating direction implicit (ADI) method and how it can be used to solve the Poisson equation (1.0.1). A numerical experiment is also performed using the ADI method with the finite difference discretization. In Chapter 4, the optimal complexity spectral method which uses the ADI method in solving (1.0.1) is discussed and we also perform a numerical experiment. Conclusion and discussion of possible future work are devoted into Chapter 5.

2. Finite Difference Methods

Approximating the numerical solution of the Poisson equation (1.0.1) involves the use of various discretization methods some of which include FD, finite volume (FV), finite element (FE), or pseudo-spectral methods (Fuka, 2015; Jumat Sulaimon and Khatim, 2007; Komla Domelevo, 2005).

In our study, we first investigate finite difference methods, which are one of the most classical techniques of solving PDEs such as (1.0.1).

2.1 Finite Difference For Univariate Functions

Suppose n is a positive integer and U is a function which is differentiable m-times on the closed interval $[b, x]$ and again differentiable $(m+1)$-times on the open interval (b, x). Then, by Taylor's theorem,

$$U(x) = U(b) + (x-b)U'(b) + \frac{(x-b)^2}{2!}U''(b) + \ldots + \frac{(x-b)^m}{m!}U^{(m)}(b) + R_m(x), \tag{2.1.1}$$

where

$$R_m(x) = \frac{(x-b)^{m+1}}{(m+1)!}U^{(m+1)}(c),$$

is the remainder and $U^{(k)}(b)$ is the k-th derivative of U evaluated at b where c lies between x and b.

Let $x \to x+h$ and $b = x$ in (2.1.1), then we have

$$U(x+h) = U(x) + hU'(x) + \frac{h^2}{2!}U''(x) + \ldots + \frac{h^m}{m!}U^{(m)}(x) + R_m(x), \tag{2.1.2}$$

and if $x \to x-h$, then we have

$$U(x-h) = U(x) - hU'(x) + \frac{h^2}{2!}U''(x) + \ldots + \frac{(-h)^m}{m!}U^{(m)}(x) + R_m(x). \tag{2.1.3}$$

Subtracting (2.1.3) from (2.1.2) gives

$$U(x+h) - U(x-h) = 2U'(x)h + \frac{h^3}{3}u'''(x) + \mathcal{O}(h^5),$$

and therefore

$$U'(x) \approx \frac{U(x+h) - U(x-h)}{2h} - \frac{h^2}{6}u'''(x) + \mathcal{O}(h^4), \tag{2.1.4}$$

which is the first order *central difference approximation*.

Adding (2.1.2) and (2.1.3), we have

$$U(x+h) + U(x-h) = 2U(x) + h^2U''(x) + \frac{h^4}{12}u^{(4)}(x) + \mathcal{O}(h^6),$$

which upon rearrangement gives

$$U''(x) \approx \frac{U(x+h) - 2U(x) + U(x-h)}{h^2} - \frac{h^2}{12}u^{(4)}(x) + \mathcal{O}(h^4), \tag{2.1.5}$$

which is the second order *central difference approximation* of order $\mathcal{O}(h^2)$.

For higher order central difference approximations, we set $h \to 2h$ in (2.1.5) and we obtain

$$U''(x) \approx \frac{U(x+2h) - 2U(x) + U(x-2h)}{4h^2} + \mathcal{O}(4h^2). \tag{2.1.6}$$

Multiplying (2.1.4) by 4 and subtracting (2.1.6) to eliminate the h^2 term, from the result gives

$$3u''(x) = \frac{-u(x-2h) + 16u(x-h) - 30u(x) + 16u(x+h) - u(x+2h)}{4h^2} + \mathcal{O}(h^4),$$

thus,

$$u''(x) = \frac{-u(x-2h) + 16u(x-h) - 30u(x) + 16u(x+h) - u(x+2h)}{12h^2} + \mathcal{O}(h^4), \tag{2.1.7}$$

which is a second order *central difference approximation* of order $\mathcal{O}(h^4)$.

2.1.1 Order of Convergence of the Second Order Central Difference Approximation. In this section, we perform a numerical experiment to confirm the order of convergence of the second order central difference approximation (2.1.5).

Consider a function e^x with known derivatives and then approximate the error at $x = 1$ using step-size $h = 10^{-1}, \ldots, 10^{-6}$.

We expect the error term, $E(h)$, for (2.1.5) will have a part due to round-off error and a part due to truncation error, i.e.,

$$\begin{aligned} u''(x)_{\text{exact}} - u''(x)_{\text{machine}} &= u''(x) - \frac{\widetilde{u}(x+h) - 2\widetilde{u} + \widetilde{u}(x-h)}{h^2} \\ &= u''(x) - \frac{u(x+h) + \varepsilon_0 - 2u(x) - 2\varepsilon_1 + u(x-h) + \varepsilon_2}{h^2} \\ &= u''(x) - \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + \frac{\varepsilon_0 - 2\varepsilon_1 + \varepsilon_2}{h^2} \\ &= (u''(x)_{\text{exact}} - u''(x)_{\text{approx}}) + \text{error}_{\text{round-off}}. \end{aligned}$$

Therefore

$$E(h) = \frac{\varepsilon_0 - 2\varepsilon_1 + \varepsilon_2}{h^2} - \frac{h^2 u^{(4)}(c)}{12}. \tag{2.1.8}$$

Suppose we assume that each error ε_k is of the magnitude ϵ, with signs that amass error, such that $|e^x| \leq M$, then we have the error bound:

$$|E(h)| \leq \frac{4\epsilon}{h^2} + \frac{Mh^2}{12}. \tag{2.1.9}$$

If h is small, then the term $\dfrac{4\epsilon}{h^2}$, which is due to the round-off error, is large. When h is large, the term $\dfrac{Mh^2}{12}$ is large and therefore, the optimal step size h will minimize the quantity

$$f(h) = \frac{4\epsilon}{h^2} + \frac{Mh^2}{12}. \tag{2.1.10}$$

Taking the derivative of (2.1.10) and setting it to zero gives $\frac{Mh}{6} = \frac{8\epsilon}{h^3}$ which on simplification yields

$$h^4 = \frac{48\epsilon}{M},$$

from which we get the optimal value

$$h = \left(\frac{48\epsilon}{M}\right)^{1/4}. \tag{2.1.11}$$

Since the portion of the error generated by the round off is inversely proportional to the square of h, it follows that this error increases as h reduces. A partial way to solve this problem is to make use of higher order formula (like (2.1.7)) so that a better accuracy is achieved when a large value of h is used.

The error term for (2.1.7) has the form

$$E(h) = \frac{16\varepsilon}{3h^2} + \frac{h^4 f^{(6)}(c)}{90}, \tag{2.1.12}$$

where $c \in [x-2h, x+2h]$. a bound for $|E(h)|$ is

$$|E(h)| \leq \frac{16\varepsilon}{3h^2} + \frac{h^4 M}{90}, \tag{2.1.13}$$

where $|f^{(6)}(c)| \leq M$. The optimal value for h is given by the formula

$$h = \left(\frac{240\varepsilon}{M}\right)^{\frac{1}{6}}. \tag{2.1.14}$$

When we plot the $\log\log$ graph (i.e., $\log(h)$ on x-axis and $\log(abs\_err$[1]), we have Figure 2.1.

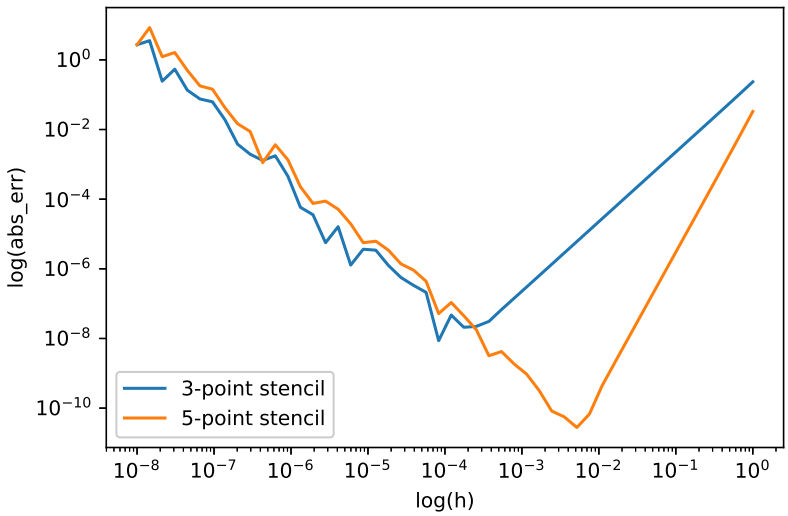


Figure 2.1: *A plot of* $\log(h)$ *against* $\log(abs\_err)$ *of the derivatives of the function* e^x *with* $x = 1$ *using the 3-point and 5-point second order central difference methods. In this figure, the absolute error initially decreases but as* $h \to 0$ *the error starts increasing. This shows that the rounding error dominates as predicted by (2.1.9). The* $\mathcal{O}(h^4)$ *of (2.1.7) performs better than* $\mathcal{O}(h^2)$ *of (2.1.5).*

[1]The absolute error defined by $abs\_err = |\text{approx} - e|$.

From Figure 2.1, the graph shows the logarithm of absolute error as a function of logarithm of step size for the test problem mentioned above. We observe that as h tends to zero, the error was reducing then it started increasing. We also observed from the plotted points that it is linear with slope 2. Hence, we say that the order of convergence of the second order central difference approximation (2.1.5) is $\mathcal{O}(n^2)$.[2]

2.2 Finite Difference Discretization of the 2D Poisson Equation

Consider the 2D Poisson's equation on a square domain with zero homogeneous Dirichlet conditions

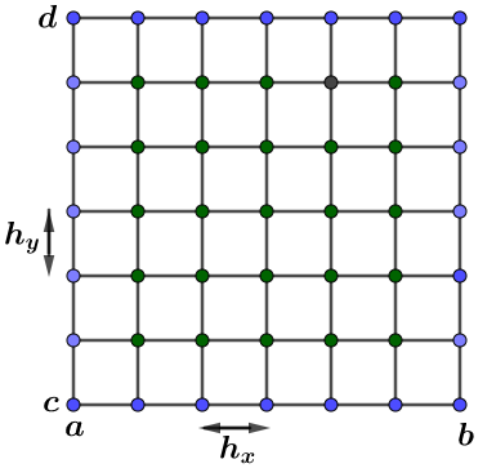
$$u_{xx} + u_{yy} = f, \quad (x, y) \in [-1, 1]^2, \quad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0, \tag{2.2.1}$$

where f is a known continuous function and $u = u(x, y)$ is our desired solution.

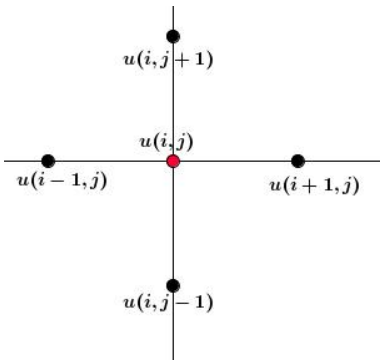
We seek to find its solution using the FD method with a five point stencil on an $(n + 1) \times (n + 1)$ equispaced grid. This is done by discretizing (2.2.1) at the value of the function on the grid points using uniform Cartesian grid.

A uniform Cartesian grid can be generated as

$$\begin{cases} x_i = a + ih_x, & i = 0, 1, \ldots, n + 1, \quad h_x = \dfrac{b - a}{n + 1}, \\ y_j = c + jh_y, & j = 0, 1, \ldots, n + 1, \quad h_y = \dfrac{d - c}{n + 1}. \end{cases}$$



(a) *A uniform Cartesian grid*



(b) *A 5 point stencil*

Figure 2.2: (a) *A Cartesian grid where h_x is the distance between two adjacent x grid points and h_y is the distance between two adjacent y grid points*; (b) *A stencil showing values of u at five grid points.*

The solutions at the boundary points of the grid are known from the boundary conditions and the solution at the internal grid points are to be approximated.

[2]Since the approximated errors are proportional to powers of h, then the slope of $\log - \log$ plot gives the order of accuracy of the scheme.

We approximate the internal grid points by approximating (2.2.1) using the second order central difference approximation (2.1.5) to get

$$-\frac{1}{h_x^2}[-u_{i-1,j}+2u_{i,j}-u_{i-1,j}]-\frac{1}{h_y^2}[-u_{i,j-1}+2u_{i,j}-u_{i,j+1}]=f_{i,j}, \tag{2.2.2}$$

where h_x is the distance between two adjacent x grid points and h_y is the distance between two adjacent y grid points. Equation (2.2.2) is a linear combination of 5 different values of u at 5 grid points, and is equal to $f_{i,j}$.

From equation (2.2.2), the order of accuracy of this scheme is second order in h_x and h_y.

Assuming a uniform spatial discretization, i.e., $h_x = h_y = h$ in (2.2.2), we get

$$-\frac{1}{h^2}[-u_{i-1,j}+2u_{i,j}-u_{i+1,j}]-\frac{1}{h^2}[-u_{i,j-1}+2u_{i,j}-u_{i,j+1}]=f_{i,j}, \tag{2.2.3}$$

where $i, j = 1, 2, \ldots, n$.

The order of accuracy of the second order central difference scheme is $\mathcal{O}(h^2)$.

2.2.1 FD Discretization of 2D Poisson Equation to Linear System Form. In the arrangement of the solution vector, we prefer to arrange the solution vector using the natural ordering and after removing boundary elements, we have

$$\vec{X} = [u_{1,1}, u_{2,1}, \ldots, u_{n,1}, u_{1,2}, u_{2,2} \ldots, u_{n,2}, \ldots, u_{1,n}, u_{2,n} \ldots, u_{n,n}]^T .$$

This results in an $n^2 \times n^2$ linear system is of the form

$$A\vec{X} = \vec{F} \tag{2.2.4}$$

where

$$A = \begin{bmatrix} D & -I & 0 & 0 & 0 & \ldots & 0 \\ -I & D & -I & 0 & 0 & \ldots & 0 \\ 0 & -I & D & -I & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & -I & D & -I & 0 \\ 0 & \ldots & \ldots & 0 & -I & D & -I \\ 0 & \ldots & \ldots & \ldots & 0 & -I & D \end{bmatrix},$$

I is the $n \times n$ identity matrix, D is the $n \times n$ matrix given by

$$D = \begin{bmatrix} 4 & -1 & 0 & 0 & 0 & \ldots & 0 \\ -1 & 4 & -1 & 0 & 0 & \ldots & 0 \\ 0 & -1 & 4 & -1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & -1 & 4 & -1 & 0 \\ 0 & \ldots & \ldots & 0 & -1 & 4 & -1 \\ 0 & \ldots & \ldots & \ldots & 0 & -1 & 4 \end{bmatrix},$$

and $\vec{F}$ is defined by

$$\vec{F} = h^2 \left[f_{1,1}, f_{2,1}, \ldots, f_{n,1}, f_{1,2}, f_{2,2} \ldots, f_{n,2}, \ldots, f_{1,n}, f_{2,n} \ldots, f_{n,n}\right]^T .$$

We call the formula in 2.2.4 the Kronecker form of the discretization.

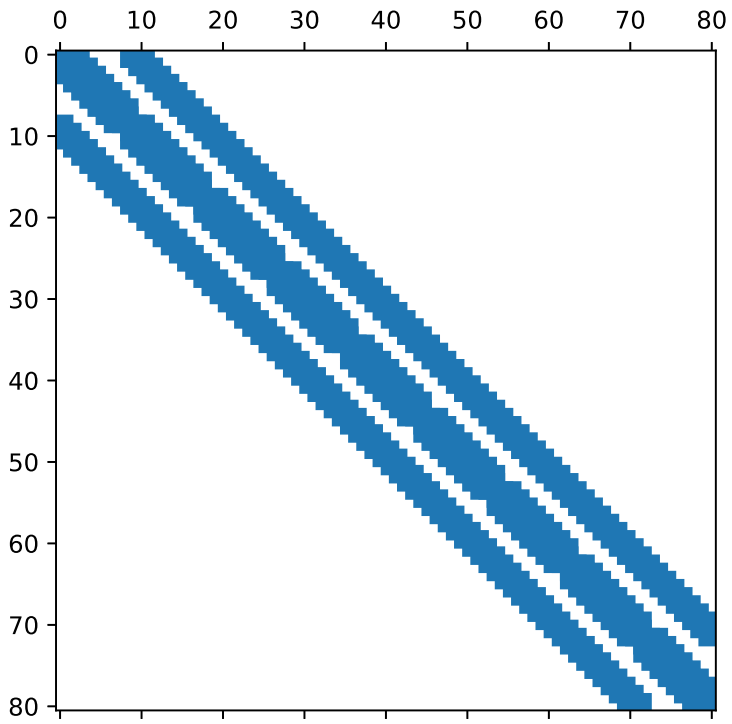


Figure 2.3: *A useful tool for visualizing the sparsity pattern of a matrix is the sparse plot (spy plot for short). Illustration for* $n = 9$ *where matrix* A *is a* $n^2 \times n^2$ *matrix.*

From above, we deduce that A contains n block columns of n. It is crucial to note that stipulated values of u (usually lying on the boundary) would have their corresponding elements detached from matrices I and D. For the usual instance that all the nodes on the boundary are set, we have $1 \leq i, j \leq n$ and the system would have the dimensions $n^2 \times n^2$ where D and I would have dimensions $n \times n$.

There are a variety of methods that can be employed for solving the linear system (2.2.4). We have the direct and iterative methods. The direct methods include the Gaussian elimination and LU or Cholesky decomposition methods. While the iterative methods include the Mesh relaxation methods (Jacobi, Gauss-Siedel, and successive over relaxation (SOR) method), Matrix methods (Thomas tridiagonal form, Sparse matrix method, preconditioned conjugate gradient (PCG) method, and Multi-grid method) and Alternating direction implicit (ADI) methods (Vasileska, 2010), which we will focus on later on in this literature.

The direct methods are based on triangularization techniques. These techniques are used to get rid of unknowns in (2.2.4) in a systematic way such that we obtain a triangular system in the end that can be easily solved. For a non-singular matrix A in (2.2.4), the rows can be reordered so that an LU decomposition exists. The LU decomposition method involves decomposing the matrix A in (2.2.4) into upper and lower triangular matrix $A = LU$ which requires approximately $\frac{2}{3}n^6$ flops. The system (2.2.4)

is then identical to $LU\vec{X} = \vec{F}$, and this decomposes into two triangular systems $L\vec{Y} = \vec{F}$ and $U\vec{X} = \vec{F}$ requiring n^4 flops each. These decomposition can be repeatedly used to directly obtain the solution vector $\vec{X}$. The disadvantage with this method is its complexity that is in total $\mathcal{O}\left(\frac{2}{3}n^6\right)$. Applying the LU decomposition on the banded sparse matrix A in (2.2.4) increases the number of non-zero entries in A. This is done in such a way that some zero entries in A become non-zero (fill-in) in the upper and lower triangular matrices L and U which consequently decreases the sparsity of these matrices.

The iterative methods begin with the first approximation which is consecutively enhanced by the repeated implementation of the same algorithm, till an adequate accuracy is reached. The iterative methods are often used when we have huge sparse system of equations like (2.2.4) and when a good approximation of the solution is known. In addition, error analysis and convergence rate are two pivotal characteristics of the theory of iterative methods.

2.2.2 FD Discretization of 2D Poisson Equation to Sylvester Equation Form. Now, instead of the linear system (2.2.4), we consider the discretization (2.2.3) in a manner that leads to a linear system of the form of the *Sylvester equation*

$$KU + UK = F, \tag{2.2.5}$$

where

$$K = -\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 2 & -1 & 0 \\ 0 & \dots & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & \dots & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2.2.6}$$

is a tridiagonal matrix with $h = \dfrac{2}{n+1}$. Notice that K is precisely the matrix form of the central difference formula (2.1.5).

The matrix U in (2.2.5) given by

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ u_{2,1} & u_{2,2} & \dots & u_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{n,1} & u_{n,2} & \dots & u_{n,n} \end{bmatrix} \in \mathbb{R}^{n\times n}$$

represents the values of the solution on the interior nodes of the $(n+2)\times(n+2)$ equispaced grid and matrix

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,n} \\ f_{2,1} & f_{2,2} & \dots & f_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & \dots & f_{n,n} \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

Note that KU applies the finite difference formula to the columns, and hence acts as the derivative with respect to y. Also, $UK^T = UK$ applies the finite difference formula to the rows of U and hence differentiates with respect to x.

Equation (2.2.5) can be solved either directly or iteratively. One of the direct approaches to solving (2.2.5) is the Bartels-Stewart method (Bartels, 1972). The idea of this approach is using the real Schur decompositions (Golub and Loan, 1996; Trefethen and Bau III, 1997) of K to transform (2.2.5) into a triangular system that can be properly solved using backward or forward substitutions.

In the case of a real symmetric matrix, a Schur decomposition becomes an eigenvalue decomposition and we discuss this below.

2.2.3 Eigenvalue Decomposition of Matrix K. Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of matrix K and $y_1, \ldots, y_n$ the corresponding eigenvectors. If matrix K is diagonalizable, then it can be written as

$$K = S\Lambda S^{-1}, \tag{2.2.7}$$

where S is the matrix of orthonormal eigenvectors, with the eigenvectors $y_1, \ldots, y_n$ of K in its columns and Λ is the eigenvalue matrix, i.e., the diagonal matrix of eigenvalues λ's,

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}.$$

If K is normal, i.e., $K^TK = KK^T$ then (2.2.7) can be re-written as

$$K = S\Lambda S^T. \tag{2.2.8}$$

Substituting (2.2.7) into (2.2.3), we have

$$S\Lambda S^TU + US\Lambda S^T = F. \tag{2.2.9}$$

We proceed to simplify (2.2.9) by performing left and right multiplications by S^T and S respectively, giving

$$S^T(S\Lambda S^TU)S + S^T(US\Lambda S^T)S = S^TFS$$

and

$$\Lambda(S^TUS) + (S^TUS)\Lambda = S^TFS.$$

Putting $S^TFS = G$ and $S^TUS = V$, we have

$$\Lambda V + V\Lambda = G. \tag{2.2.10}$$

Although this is another Sylvester equation, notice that Λ is now diagonal. Solving for V in (2.2.10) leads to

$$\begin{bmatrix} v_{1,1}(\lambda_1+\lambda_1) & v_{1,2}(\lambda_1+\lambda_2) & \ldots & v_{1,n}(\lambda_1+\lambda_n) \\ v_{2,1}(\lambda_2+\lambda_1) & v_{2,2}(\lambda_2+\lambda_2) & \ldots & v_{2,n}(\lambda_2+\lambda_n) \\ \vdots & \vdots & \vdots & \vdots \\ v_{n,1}(\lambda_n+\lambda_1) & v_{n,2}(\lambda_n+\lambda_2) & \ldots & v_{n,n}(\lambda_n+\lambda_n) \end{bmatrix} = \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n,1} & g_{m-1,2} & \cdots & g_{n,n} \end{bmatrix}, \tag{2.2.11}$$

where $v_{i,j}, \lambda_i$, and $g_{i,j}$ are elements of matrices V, Λ and G respectively.

Equating the corresponding entries of the matrices in (2.2.11) yields

$$V = C \circ G, \qquad \text{where } C_{ij} = \frac{1}{\lambda_i + \lambda_j},$$

where '$\circ$' is the Hadamard matrix product.[3]

Since $S^T U S = V$ then $U = SVS^T$ and this leads to

$$U = S(C \circ G)S^T. \tag{2.2.12}$$

Algorithm 1 The Eigendecomposition method to solve $KU + UK = F$

Input: $K \in \mathbb{R}^{n \times n}$ which is symmetric
Ouput: $U \in \mathbb{R}^{n \times n}$

1: **procedure** USING EIGENDECOMPOSITION METHOD
2: Compute the eigenvalues and eigenvectors of K, i.e., eigendecomposition $K = S\Lambda S^T$
3: Compute $G = S^T F S$
4: Compute the Hadamard matrix product of matrix C in (2.2.12) and G, i.e., $H = C \circ G$
5: Compute $U = SHS^T$
6: **return** U

Figure 2.4: *Pseudocode for the Eigendecomposition method for solving (2.2.5).*

The complexity of computing the eigenvalues and eigenvectors is $\mathcal{O}(n^3)$ and the complexity of multiplying by matrices S and S^T is $\mathcal{O}(n^3)$ (Rønquist, 2007). To obtain a fast method, we need to improve these two complexities.

Note that $\mathcal{O}(n^3)$ is much better than $\mathcal{O}(n^6)$, but we desire a better complexity.

2.2.4 Derivation of Eigenvalues and Eigenvectors of Matrix K. Consider the matrix K in (2.2.5). We can write its eigenvalue equation as

$$Ku = \lambda u. \tag{2.2.13}$$

Using (2.2.3), we can write the left side of (2.2.13) in the component form

$$2u_k - u_{k-1} - u_{k+1} = \lambda u_k, \quad k = 1, 2, \ldots, n. \tag{2.2.14}$$

The solution u of (2.2.13) can be written in the Fourier mode $u_k = e^{ik\varphi}$ as suggested in Von Neumann stability analysis (Dedner, 2011) where i the imaginary number $\sqrt{-1}$.

Substituting $u_k = e^{ik\varphi}$ in (2.2.14) gives

$$2e^{ik\varphi} - e^{i(k-1)\varphi} - e^{i(k+1)\varphi} = \lambda e^{ik\varphi},$$

[3]The Hadamard product of two $m \times n$ matrices is the $m \times n$ matrix of elementwise products (Styan, 1973).

which on dividing through by $e^{ik\varphi}$ yields

$$2 - e^{-i\varphi} - e^{i\varphi} = \lambda.$$

Since $e^{i\varphi} = \cos\varphi + i\sin\varphi$, we have

$$2 - 2\cos\varphi = \lambda. \tag{2.2.15}$$

Using a solution $u_k = e^{-ik\varphi}$ similarly yields the same result in (2.2.15).

Therefore, we say u_k can be written as a linear combination

$$u_k = \tau e^{ik\varphi} + \rho e^{-ik\varphi}. \tag{2.2.16}$$

Applying the boundary conditions $u_0 = 0$ and $u_{n+1} = 0$ on (2.2.16) yields

$$\begin{aligned} u_0 &= \tau + \rho = 0 \quad \text{i.e. } \rho = -\tau \\ u_{n+1} &= \tau e^{i(n+1)\varphi} + \rho e^{i(n+1)\varphi} = \tau(e^{i(n+1)\varphi} - e^{i(n+1)\varphi}) \\ &= \tau(\cos(n+1)\varphi + i\sin(n+1)\varphi - \cos(n+1)\varphi + i\sin(n+1)\varphi) \\ &= 2\tau i\sin(n+1)\varphi = 0. \end{aligned} \tag{2.2.17}$$

From (2.2.17), it follows that $(n+1)\varphi = j\pi$ for some integer j. Therefore, $\varphi = \frac{j\pi}{n+1}$ and on substitution in (2.2.15) yields

$$\lambda_j = 2 - 2\cos\left(\frac{j\pi}{n+1}\right) = 4\sin^2\left(\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n, \tag{2.2.18}$$

which are the eigenvalues of the matrix K for the $1D$ Poisson equation.

The corresponding eigenvectors of the eigenvalues λ_j in (2.2.18) are

$$[u_j]_k = \gamma\sin\left(\frac{kj\pi}{n+1}\right) = \gamma\sin jx_k, \quad j, k = 1, 2, \ldots, n, \tag{2.2.19}$$

where γ is a scaling constant.

The eigenvectors of (2.2.19) are orthogonal and this is guaranteed by the symmetry property of K in (2.2.5). Dividing each column of the matrix of eigenvectors by $\sqrt{\frac{n+1}{2}}$ gives the orthonormal eigenvectors (Strang, 2006)

$$[u_j]_k = \sqrt{\frac{2}{n+1}}\sin\left(\frac{kj\pi}{n+1}\right), \quad j, k = 1, 2, \ldots, n,$$

which is the orthonormal eigenvector matrix S in (2.2.7).

2.2.5 Fast Application of FFT. If K is the matrix in (2.2.5), then we know the eigenvalues and eigenvectors of K explicitly (see 2.2.18).

According to Britanak et al. (2010), matrix S in (2.2.12) is a Type-I discrete sine transform (DST), so we can apply it with the FFT and U can be computed via (2.2.12) in a total of $\mathcal{O}(n^2 \log n)$ operations (Strang, 2006).

The approach to achieving this is illustrated in Algorithm 2 below.

Algorithm 2 The FFT method to solve $KU + UK = F$

Input: $K, F \in \mathbb{R}^{n \times n}$
Ouput: $U \in \mathbb{R}^{n \times n}$

1: **procedure** USING DISCRETE SINE TRANSFORM
2: Compute $F_1 = \text{IDST}(F)$ which is equivalent to $S^T F$
3: Simplify $S^T F S = F_1 S$ and compute $F_2 = (\text{DST}(F_1^T))^T$
4: Compute the Hadamard matrix product of matrix C in (2.2.12) and F_2 i.e. $F_3 = C \circ F_2$
5: Compute $F_4 = \text{DST}(F_3^T)$ which is equivalent to SF_3
6: Simplify $SF_3S^T = F_4S^T$ and compute $U = \text{IDST}(F_4^T)$
7: **return** U

Figure 2.5: *Pseudocode for the FFT method described for solving (2.2.5). The complexity of using the FFT method is* $\mathcal{O}(n^2 \log n)$.

2.3 Numerical Experiments

In this section, we perform a numerical experiment for the $2D$ Poisson problem (2.2.1) where $f(x, y) = 1$.

In order to obtain the solution of (2.2.1) for $f = 1$, we will be using the following approaches:

(a) Kronecker form of (2.2.4);

(b) Sylvester form (2.2.5):

i. using eigenvalue decomposition (eigendecomposition);

ii. using fast fourier transform (FFT);

iii. using the default Sylvester equation solver in Python 3.

For these approaches, we will be plotting absolute error against n as n increases, time against n as n increases, and absolute error against time as n increases. These plots are presented in Figure 2.6.

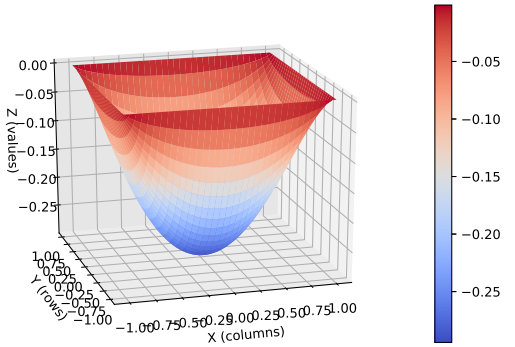
2.4 Summary and Conclusion of Numerical Experiments

We have been able to apply the Kronecker, default Sylvester equation solver on Python 3, Eigendecomposition, and FFT methods to solving the Poisson problem (2.2.1). In the execution time plot presented in Figure 2.6b, we considered $n \in [11, 101]$ for the Kronecker method due to the computational cost involved. For the remaining methods we considered $n \in [11, 2011]$ and we observed that the FFT

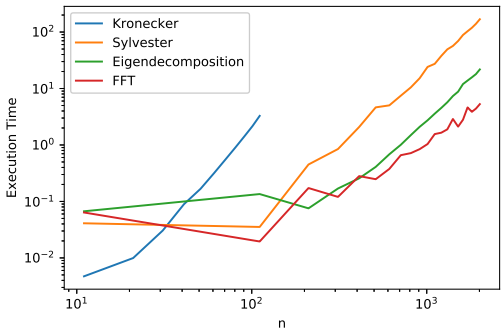
method requires less time of computation compared to the remaining two methods. In Figure 2.6c and 2.6d respectively, we presented the absolute error and absolute error-time plots for the FFT method only. This was due to the fact that the absolute error and absolute error-time plots for the FFT method and other three methods are similar but the FFT accuracy is better.

It is observable that the FFT method which is specific to this problem in comparison with the other three methods mentioned above is quite better in terms of computational cost and accuracy.

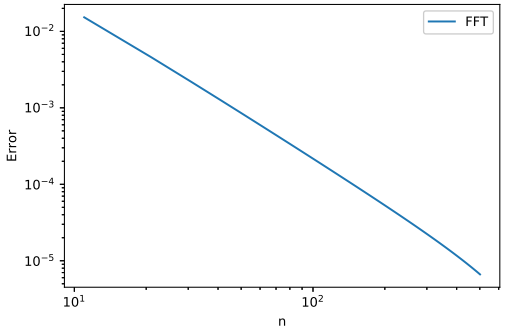
Now that we have a less expensive method to finding the solution to Equation 2.2.1, we proceed to discuss the ADI method in Chapter 3 with a better complexity compared to $\mathcal{O}(n^3)$ of the FFT. This ADI method will be used to derive an optimal complexity spectral method for the Poisson's problem (2.2.1) in Chapter 4.



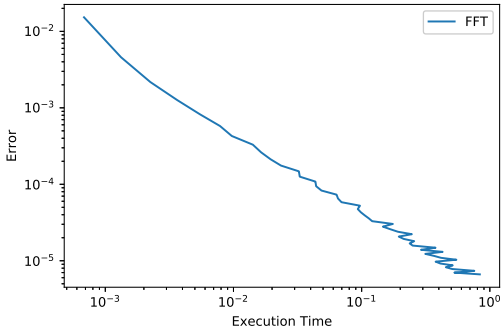
(a) *Plot of the solution* u.



(b) *Execution time against* n *plot as* n *increases.*



(c) *Error against* n *plot as* n *increases.*



(d) *Error against execution time plot as* n *increases.*

Figure 2.6: (a) *Solution* u *to the* $2D$ *Poisson problem (2.2.1) where* $f(x, y) = 1$ *with* $n = 100$; (b) *Time required for the solver of each method (Kronecker, Sylvester Solver, Eigendecomposition, and FFT) to generate the solution* u *for* $11 \leq n \leq 1001$. *For the Kronecker method* $11 \leq n \leq 101$; (c) *Absolute error incurred using the FFT method for* $11 \leq n \leq 1001$ *and we observe that this agrees with the convergence rate derived earlier in Section 2*; (d) *Execution time against absolute error for the FFT method for* $11 \leq n \leq 1001$ *and we will compare this with the spectral solver in Chapter 4. We observe from these numerical experiments that the FFT method is faster.*

3. The Alternating Direction Implicit Method

The alternating direction implicit method (ADI) (Peaceman and Rachford, 1955; Lu and Wachspress, 1991) is an iterative algorithm which solves Sylvester matrix equations of the form

$$AU - UB = F, \tag{3.0.1}$$

where $A, B, F \in \mathbb{C}^{n\times n}$ are known and $U \in \mathbb{C}^{n\times n}$ is the desired solution. The ADI method is often used in solving for the solution vector of PDEs of elliptic type (Wachspress, 1988).

In general, the ADI method is employed in an iterative manner such that the iterates $U_0, U_1, \ldots,$ are computed in the hope that $\|U - U_w\|_2 \to 0$ as $w \to \infty$. At the beginning of the wth iteration of this algorithm, two shifts p_w and q_w are picked, and at the end of each iteration a test (for desirable solution) is performed to decide if the iterative process should be terminated or not.

One way to think of obtaining the solution is illustrated in Algorithm 3.

Algorithm 3 The ADI method to solve AU-UB=F

Input: $A, B, F \in \mathbb{C}^{n\times n}$
Ouput: $U \in \mathbb{C}^{n\times n}$ which is the approximate solution to (3.0.1)

Choose $U_0 := 0$
Set $w := 0$
do
    Select ADI shifts p_w and q_w
    Solve $U_{w+1/2} = (F - (A - p_w I)U_w)(B - p_w I)^{-1}$
    Solve $U_{w+1} = (A - q_w I)^{-1}(F - U_{w+1/2}(B - q_w I))$
    $w := w + 1$
while not converged
return U_W

Figure 3.1: *Pseudocode for the ADI method described as an iterative algorithm for solving (3.0.1). We cannot use this approach because in practise these matrices are not inverted, rather the linear systems will be solved. The approach of solving the linear system will be discussed in Algorithm 4.*

There are several methods that can be employed in selecting the shift parameters and determining when to terminate the iteration process (Sabino, 2006). In selecting these shifts, it is imperative to select good shifts so that the ADI method can converge rapidly (Benner et al., 2009).

In order to make the algorithm practical, we require another property, namely the linear solves which can be computed quickly.

Now, solving (3.0.1) requires that the properties itemized below are satisfied.

P1: Normal Matrices (Fortunato and Townsend, 2017)

The matrices A and B are normal matrices. The normality of these matrices simplifies the error analysis of the ADI method.

If this property holds, then there is a bound on the error $\|U - U_W\|_2$ that only depends on the eigenvalues of matrices A and B and the shifts $p_0, p_1, \ldots, p_{W-1}$ and $q_0, q_1, \ldots, q_{W-1}$ (Benner et al., 2009).

P2: Real and Disjoint Spectra (Fortunato and Townsend, 2017)

There are real disjoint non-empty intervals $[a, b]$ and $[c, d]$ such that the spectra of A, $\sigma(A) \subset [a, b]$ and the spectra of B, $\sigma(B) \subset [c, d]$. The property of (3.0.1) allows us to derive explicit expressions for the ADI shifts using the following theorem.

Theorem 3.1. (Fortunato and Townsend, 2017)

Let W be a fixed integer and let U satisfy (3.0.1), where P1 and P2 hold. We execute the ADI method with the shifts

$$p_w = T\left(-\alpha \mathrm{dn}\left[\frac{2w+1}{2W}K(\nu), \nu\right]\right), \quad q_w = T\left(\alpha \mathrm{dn}\left[\frac{2w+1}{2W}K(\nu), \nu\right]\right), \tag{3.0.2}$$

for $0 \leq w \leq W - 1$, where

$$\nu = \sqrt{1 - \frac{1}{\alpha^2}}K(\nu) \tag{3.0.3}$$

is the complete elliptic integral of the first kind (Abramowitz and Stegun, 1972) and $\mathrm{dn}(z, \nu)$ is the Jacobi elliptic function of the third kind (Hall, 1995). Here, α is the real number given by

$$\alpha = -1 + 2\xi + 2\sqrt{\xi^2 - \xi}, \tag{3.0.4}$$

with

$$\xi = \frac{|c - a||d - b|}{|c - b||d - a|}, \tag{3.0.5}$$

and T is the Möbius transformation that maps $\{-\alpha, -1, 1, \alpha\}$ to $\{a, b, c, d\}$.

For the case where $B = -A^T$, it follows that $c = -b$ and $d = -a$ and then the bound simplifies further as $4\nu(1/\sqrt{\nu}) = 2\mu(a/b)$ and the bound remains valid if we replace $2\mu(a/b)$ by $\log(4b/a)$ (Fortunato and Townsend, 2017) which gives

$$\|U - U_W\|_2 \leq 4\left[\exp\left(\frac{\pi^2}{\log(4b/a)}\right)\right]^{-2W}\|U\|_2. \tag{3.0.6}$$

Theorem 3.1 allows us to use the ADI method just like a direct method to solve $AX + XA^T = F$ when P1 and P2 hold. For a relative accuracy of $0 < \varepsilon < 1$, the simplified bound in (3.0.6) shows $\|U - U_W\|_2 \leq \varepsilon\|U\|_2$ if we take

$$W = \left\lceil \frac{\log(16\nu)\log(4/\varepsilon)}{\pi^2} \right\rceil \tag{3.0.7}$$

and we execute the ADI method with shifts given in (3.0.2).

P3: Fast Shifted Linear Solves (Fortunato and Townsend, 2017)

For any $p, q \in \mathbb{C}$, the linear systems $(A - pI)x = b$ and $(B - qI)x = b$ can be solved in $\mathcal{O}(n)$ operations.

If this property holds, then each ADI iteration costs only $\mathcal{O}(n^2)$ operations and the overall cost of the ADI method with W iterations is $\mathcal{O}(Wn^2)$ operations.

3.1 A Fast ADI Poisson Solver for FD Method

In this section, we will describe the fast ADI Poisson solver with the second order five-point FD stencil.

Recall the Sylvester matrix given by (2.2.5) which is a FD discretization of the Poisson equation (1.0.1) with a five-point stencil on an $(n+1)\times(n+1)$ equispaced grid. We now verify that the three properties holds for (2.2.5):

P1: $A = K$ and $B = -A^T$ are real and symmetric which implies that they are normal matrices.

P2: The eigenvalues of K are given by $-\frac{4}{h^2}\sin^2\left(\frac{k\pi}{2(n+1)}\right)$ for $1 \leq k \leq n$ with $h = \frac{2}{n+1}$. Since $\frac{2x}{\pi} \leq \sin x \leq 1$ for $x \in [0, \frac{\pi}{2}]$ and $h = \frac{2}{n+1}$, the eigenvalues of $A = K$ are contained in the interval $[-n^2, -1]$. The eigenvalues of $B = -A^T$ are contained in $[1, n^2]$.

P3: For any $p, q \in \mathbb{C}$, the linear systems $(A-pI)x = b$ and $(B-qI)x = b$ are tridiagonal and therefore can be solved easily in $\mathcal{O}(n)$ operations (Datta, 2010).

From the simplified bound in (3.0.6), we conclude that $W = \left\lceil \frac{\log(2n)\log(4/\varepsilon)}{\pi^2} \right\rceil$ ADI iterations are required to ensure that $\|U - U_W\|_2 \leq \varepsilon\|U\|_2$ for any $\varepsilon \in (0,1)$, where the shifts are given in Theorem 3.1. The MATLAB code for this shifts described in Fortunato and Townsend (2017) was implemented in Python 3.

Algorithm 4 The Fast ADI method to solve $AU + UA^T = F$ when P1 and P2 hold

Input: $A, F \in \mathbb{R}^{n\times n}, a, b \in \mathbb{R}$ satisfying P1 and P2, and a tolerance $\varepsilon \in (0,1)$
Ouput: $U \in \mathbb{R}^{n\times n}$ such that $\|U - U_W\|_2 \leq \varepsilon\|U\|_2$

1: **procedure** Alternating Direction Implicit Method
2: $\nu := |a+b|^2/(4|a|^2|b|^2)$
3: $W := \lceil \log(16\nu)\log(4/\varepsilon)/\pi^2 \rceil$
4: Set p_w and q_w for $w \in [0, W-1]$
5: $U_0 := 0$
6: **for** $w = 0, \ldots, W-1$ **do**
7: Solve $U_{w+1/2}(B - p_wI) = F - (A - p_wI)U_w$ for $U_{w+1/2}$
8: Solve $(A - q_wI)U_{w+1} = F - U_{w+1/2}(B - q_wI)$ for U_{w+1}
9: **return** U_W

Figure 3.2: *Pseudocode for the ADI method described as an iterative algorithm for solving (3.0.1) when properties P1 and P2 holds.*

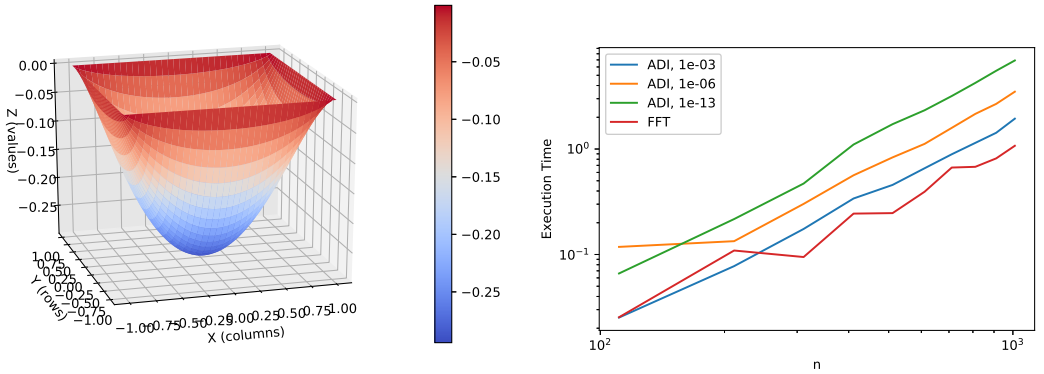
Besides, since P3 holds, each ADI iteration only costs $\mathcal{O}(n^2)$ iterations. We conclude that the fast ADI method illustrated in Algorithm 4 solves the Sylvester equation (2.2.5) in a total of $\mathcal{O}(n^2 \log n \log(1/\varepsilon))$ operations.

One interesting advantage of the fast ADI method is that it allows flexibility through the choice of an error tolerance ε and may be useful for higher order FD methods and non-uniform grids (Fortunato and Townsend, 2017).

3.2 Numerical Experiment Using the Fast ADI Poisson Solver

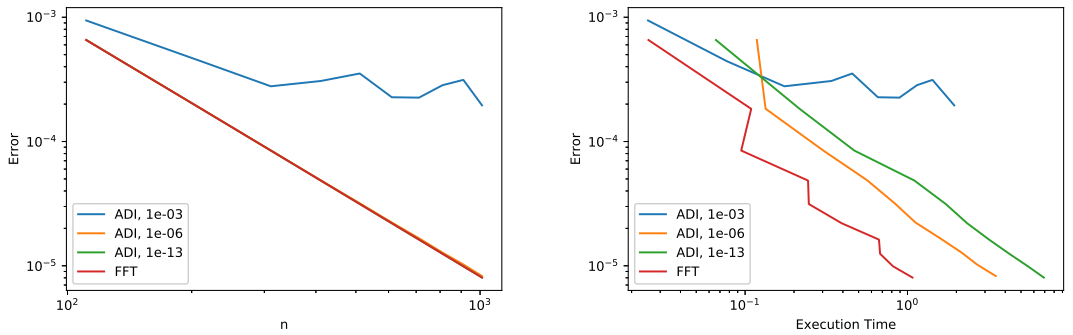
In this section, we perform a numerical experiment for the $2D$ Poisson problem (2.2.1) where $f(x, y) = 1$ using the fast ADI Poisson solver.

We demonstrate the execution time of the fast ADI method in comparison to the FFT method discussed in Chapter 2 for $111 \leq n \leq 1011$. In addition, we will be plotting absolute error against n as n increases and absolute error against time as n increases. These plots are presented in Figure 3.3.



(a) *Plot of the solution u using the fast ADI method.*

(b) *Execution time against n plot as n increases.*



(c) *Error against n plot as n increases.*

(d) *Error against execution time plot as n increases.*

Figure 3.3: (a) *Solution u to the $2D$ Poisson problem (2.2.1) where $f(x, y) = 1$ using ADI method with $n = 100$;* (b) *Execution times for the FFT and fast ADI Poisson solvers (for $\varepsilon = 10^{-3}, 10^{-6}, 10^{-13}$) with $111 \leq n \leq 1011$. We can compare the FFT solver to the fast ADI solver when $\varepsilon = 10^{-3}$;* (c) *Absolute error incurred using the fast ADI method for $111 \leq n \leq 1011$;* (d) *Execution time against absolute error for the fast ADI method for $111 \leq n \leq 1011$. We observe from the numerical experiments that the FFT which is specific to this problem is faster than the ADI method. Accuracy wise, the FFT method behaves the same as ADI and it is better when the tolerance value for the ADI is 10^{-3}. The FFT method wins here.*

3.3 Summary and Conclusion of Numerical Experiments

We have been able to apply the ADI method to solving the Poisson problem (2.2.1). In the execution time plot presented in Figure 3.3b, we considered $n \in [111, 1011]$ and we observed that the FFT method is comparable to the ADI method when $\varepsilon = 10^{-3}$. In Figures 3.3c and 3.3d respectively, we presented

the absolute error and absolute error-time plots for the FFT and ADI methods. It is observable that the FFT method in comparison with the ADI method performs better in terms of computational time. In terms of accuracy, the FFT is the same as the ADI method for small tolerance value but it is better than the ADI method for larger tolerance value.

Having described the ADI method and use it to obtain the solution to Poisson's problem (2.2.1), we proceed to derive an optimal complexity spectral method in Chapter 4.

4. Fast Spectral Methods

Spectral methods were first introduced to discretize problems associated with periodic boundary conditions (Kreiss and Oliger, 1979; Gottlieb and Orszag, 1977). These methods are a class of spatial discretizations for differential equations (Canuto et al., 2006) and are involved in finding the solution to the differential equations in terms of different smooth functions that are known (Gottlieb and Orszag, 1977). In recent times, spectral methods has surfaced as a suitable substitute to the FD and FE methods for the numerical solution of PDEs (Gottlieb and Orszag, 1977; Su, 1998), for periodic and non-periodic boundary conditions.

Spectral methods are based on the fact that the exact solution can often be well approximated by polynomials or trigonometric polynomials. With spectral methods, one can typically achieve higher degree of accuracy which can only be limited by computer accuracy in comparison to the known finite difference and finite elements method (Su, 1998). Classically, the drawback of spectral methods has been that they result in dense matrices. However, there have been recent developments in fast spectral methods (Fortunato and Townsend, 2017).

There are different kinds of spectral methods, some of which include Galerkin, Tau and Chebyshev Collocation spectral methods (Gottlieb and Orszag, 1977; Canuto et al., 2006). But in our study, we will be using the ultraspherical polynomial basis for a Galerkin-type discretization.

4.1 Fast Spectral Poisson Solver

Consider again the Poisson's equation (2.2.1) on the square with zero homogeneous Dirichlet conditions. Since (2.2.1) has homogeneous Dirichlet conditions, then its solution can be written as

$$u(x,y) = (1-x^2)(1-y^2)\kappa(x,y),$$

for some function $\kappa(x,y)$ (Fortunato and Townsend, 2017).

Now, to ascertain that we are deriving a stable spectral method, we expand the function $\kappa(x,y)$ in a orthogonal polynomial basis

$$u(x,y) \approx \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} U_{ij}(1-y^2)(1-x^2)\Phi_i(y)\Phi_j(x), \qquad (x,y)\in[-1,1]^2, \tag{4.1.1}$$

where $\Phi_0, \Phi_1, \ldots$, is a sequence of orthogonal polynomials on $[-1,1]$ and the degree of Φ_j is j for $j>0$. The matrix $U \in \mathbb{R}^{n\times n}$ is the expansion coefficients of the solution and it is our desire to obtain U so that the first $n\times n$ coefficients of $u_{xx}+u_{yy}$ match those of f. The choice of the orthogonal polynomial basis is important to deriving an optimal complexity fast ADI Poisson solver, i.e., we want to construct a Sylvester matrix equation for which the properties P1, P2 and P3 discussed in Chapter 3 hold.

4.2 Ultraspherical Polynomials

The ultraspherical (or Gegenbauer) polynomials of degree j, $C_j^{(\lambda)}$ for $j\geq 0$ are the family of orthogonal polynomials on $[-1,1]$ with respect to the weight function $(1-x^2)\lambda^{1/2}$. These polynomials are generalization of the associated Legendre polynomials (Olver et al., 2010; Szego, 1939; Koekoek and Swarttouw, 1996).

Koekoek and Swarttouw (1996) and Szego (1939) in their work defined the ultraspherical polynomials

$$C_j^{(\lambda)}(x) = \frac{\Gamma(\lambda+\frac{1}{2})}{\Gamma(2\lambda)}\frac{\Gamma(j+2\lambda)}{\Gamma(j+\lambda+\frac{1}{2})}P_j^{(\lambda-\frac{1}{2},\lambda-\frac{1}{2})}(x),$$

as Jacobi polynomials $P_j^{\alpha,\beta}(x)$ with $\alpha=\beta=\lambda-\frac{1}{2}$.

In terms of orthogonality, Koekoek and Swarttouw (1996) defined the polynomials as

$$\int_{-1}^{1}(1-x^2)^{(\lambda-\frac{1}{2})}[C_j^{(\lambda)}]^2dx = \frac{\pi 2^{1-2\lambda}\Gamma(j+2\lambda)}{(j+\lambda)\Gamma(j+1)[\Gamma(\lambda)]^2}, \quad \lambda>-\frac{1}{2} \text{ and } \lambda\neq 0.$$

Considering the normalized utraspherical polynomials of parameter $3/2$, denoted by $\widehat{C}_j^{(3/2)}$, we have

$$\widehat{C}_j^{(3/2)}(x) = \sqrt{\frac{j+(3/2)}{(j+1)(j+2)}}C_j^{(3/2)}(x), \qquad j\geq 0,$$

so that

$$\int_{-1}^{1}(\widehat{C}_j^{(3/2)}(x))^2(1-x^2)dx = 1.$$

Now, to simplify the discretization of u_{xx} in (2.2.1), we pick Φ_j in such a way that

$$\frac{d^2}{dx^2}[(1-x^2)\Phi_j(x)] \tag{4.2.1}$$

has a simpler form in terms of Φ_j.

By the product rule, we have (4.2.1) to be

$$\frac{d^2}{dx^2}[(1-x^2)\Phi_j(x)] = (1-x^2)\Phi_j''(x) - 4x\Phi_j'(x) - 2\Phi_j''(x). \tag{4.2.2}$$

Table 18.8.1 of Olver et al. (2010) gave the normalized ultraspherical polynomial $\widehat{C}_j^{(3/2)}(x)$ of degree j and the parameter $\lambda=3/2$ which satisfies the second order differential equation

$$(1-x^2)[\widehat{C}_j^{(3/2)}]''(x) - 4x[\widehat{C}_j^{(3/2)}]'(x) + j(j+3)\widehat{C}_j^{(3/2)}(x) = 0, \qquad x\in[-1,1]. \tag{4.2.3}$$

It follows that $\widehat{C}_j^{(3/2)}(x)$ is a eigenfunction of the differential operator $u\mapsto \frac{d^2}{dx^2}[(1-x^2)u]$, i.e.,

$$\frac{d^2}{dx^2}[(1-x^2)\widehat{C}_j^{(3/2)}(x)] = (1-x^2)[\widehat{C}_j^{(3/2)}]''(x) - 4x[\widehat{C}_j^{(3/2)}]'(x) - 2\widehat{C}_j^{(3/2)}(x). \tag{4.2.4}$$

Since

$$[\widehat{C}_j^{(3/2)}]''(x) = \frac{1}{(1-x^2)}\left[4x[\widehat{C}_j^{(3/2)}]'(x) - j(j+3)\widehat{C}_j^{(3/2)}(x)\right], \tag{4.2.5}$$

from (4.2.3), it follows that (4.2.4) becomes

$$\begin{aligned}\frac{d^2}{dx^2}[(1-x^2)\widehat{C}_j^{(3/2)}(x)] &= \frac{1}{(1-x^2)}\left[4x[\widehat{C}_j^{(3/2)}]'(x) - j(j+3)\widehat{C}_j^{(3/2)}(x)\right](1-x^2) - 4x[\widehat{C}_j^{(3/2)}]'(x) - 2\widehat{C}_j^{(3/2)}(x).\\ &= -[j(j+3)+2]\widehat{C}_j^{(3/2)}(x), \qquad j\geq 0. \end{aligned} \tag{4.2.6}$$

Based on this simplification, we adopt $\Phi_j = \widehat{C}_j^{(3/2)}(x)$ in (4.1.1).

4.3 Spectral Discretization of Poisson's Equation

In order to discretize (2.2.1), we apply the Laplacian to the expansion in (4.1.1) to obtain the set of equations that the matrix U must satisfy. The action of the Laplacian on each element of our basis is given by

$$\begin{aligned}\nabla^2\left[(1-y^2)(1-x^2)\right]\widehat{C}_i^{(3/2)}(y)\widehat{C}_j^{(3/2)}(x) &= \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\left[(1-y^2)(1-x^2)\right]\widehat{C}_i^{(3/2)}(y)\widehat{C}_j^{(3/2)}(x)\\ &=(1-y^2)\widehat{C}_i^{(3/2)}(y)\frac{\partial^2}{\partial x^2}\left[(1-x^2)\widehat{C}_j^{(3/2)}(x)\right]\\ &\quad+(1-x^2)\widehat{C}_j^{(3/2)}(x)\frac{\partial^2}{\partial y^2}\left[(1-y^2)\widehat{C}_i^{(3/2)}(y)\right]. \qquad (4.3.1)\end{aligned}$$

Since $\frac{d^2}{dx^2}[(1-x^2)\widehat{C}_j^{(3/2)}(x)] = -[j(j+3)+2]\widehat{C}_j^{(3/2)}(x)$ in (4.2.6), then (4.3.1) becomes

$$\begin{aligned}\nabla^2\left[(1-y^2)(1-x^2)\right]\widehat{C}_i^{(3/2)}(y)\widehat{C}_j^{(3/2)}(x) &= -(1-y^2)\widehat{C}_i^{(3/2)}(y)[j(j+3)+2]\widehat{C}_j^{(3/2)}(x)\\ &\quad-(1-x^2)\widehat{C}_j^{(3/2)}(x)[j(j+3)+2]\widehat{C}_j^{(3/2)}(x)\\ &=-[(1-y^2)(j(j+3)+2)\\ &\quad+(1-x^2)(j(j+3)+2)]\widehat{C}_i^{(3/2)}(y)\widehat{C}_j^{(3/2)}(x). \qquad (4.3.2)\end{aligned}$$

Therefore, (2.2.1) can be discretized as a generalized Sylvester matrix equation

$$PUQ^T + QUP^T = R, \qquad (4.3.3)$$

where U is the matrix of $(1-y^2)(1-x^2)\widehat{C}_i^{(3/2)}(y)\widehat{C}_j^{(3/2)}(x)$ expansion of coefficients for the solution $u(x,y)$ in (4.1.1), R is the matrix bivariate $\widehat{C}^{(3/2)}$ expansion coefficients for f in (2.2.1), Q is a diagonal matrix with $Q_{jj} = -(j(j+3)+2)$, and P is the $n\times n$ matrix that represents multiplication by $(1-x^2)$ in the $\widehat{C}^{(3/2)}$ basis, which we derive below.

According to (Olver et al., 2010), the recurrence relation for the unnormalized ultraspherical polynomials, $C^{(3/2)}$, is given by

$$(j+\lambda)C_j^{(\lambda)}(x) = \lambda\left(C_j^{(1+\lambda)}(x) - C_{j-2}^{(1+\lambda)}(x)\right) \qquad (4.3.4)$$

$$4\lambda(j+\lambda+1)(1-x^2)C_j^{(1+\lambda)}(x) = -(j+1)(j+2)C_{j+2}^{(\lambda)}(x) + (j+2\lambda)(j+2\lambda+1)C_j^{(\lambda)}(x) \qquad (4.3.5)$$

Setting $\lambda = \frac{1}{2}$ and $j = j$ in (4.3.4), we have

$$(j+\frac{1}{2})C_j^{(\frac{1}{2})}(x) = \frac{1}{2}\left(C_j^{(\frac{3}{2})}(x) - C_{j-2}^{(\frac{3}{2})}(x)\right),$$

that is,

$$C_j^{(\frac{1}{2})}(x) = \frac{C_j^{(\frac{3}{2})}(x) - C_{j-2}^{(\frac{3}{2})}(x)}{(2j+1)}. \qquad (4.3.6)$$

Setting $\lambda = \frac{1}{2}$ and $j = j+2$ in (4.3.4), we have

$$(j+\frac{5}{2})C_{j+2}^{(\frac{1}{2})}(x) = \frac{1}{2}\left(C_{j+2}^{(\frac{3}{2})}(x) - C_j^{(\frac{3}{2})}(x)\right),$$

thus,

$$C_{j+2}^{(\frac{1}{2})}(x) = \frac{C_{j+2}^{(\frac{3}{2})}(x) - C_j^{(\frac{3}{2})}(x)}{(2j+5)}. \tag{4.3.7}$$

Setting $\lambda = \frac{1}{2}$ in (4.3.5), we have

$$(2j+3)(1-x^2)C_j^{(\frac{3}{2})}(x) = -(j+1)(j+2)C_{j+2}^{(\frac{1}{2})}(x) + (j+1)(j+2)C_j^{(\frac{1}{2})}(x). \tag{4.3.8}$$

Substituting (4.3.6) and (4.3.7) in (4.3.8) yields

$$(1-x^2)C_j^{(\frac{3}{2})}(x) = \frac{-(j+1)(j+2)\left[(2j+1)C_{j+2}^{(3/2)}(x) - 2(2j+3)C_j^{(3/2)}(x) + (2j+5)C_{j-2}^{(3/2)}(x)\right]}{(2j+1)(2j+3)(2j+5)}. \tag{4.3.9}$$

Since (4.3.9) holds, we obtain after algebraic simplification that matrix P is a symmetric pentadiagonal matrix with

$$P_{j,j} = \frac{2(j+1)(j+2)}{(2j+1)(2j+5)}, \quad P_{j+1,j} = P_{j,j+1} = 0, \quad P_{j+2,j} = P_{j,j+2} = \frac{-1}{(2j+3)(2j+5)}\sqrt{\frac{(j+4)!(2j+3)}{j!(2j+7)}}. \tag{4.3.10}$$

Since $Q^T = Q$ in (4.3.3), we multiply on both side of every term in (4.3.3) by Q^{-1} and this yields

$$MU - UN = Q^{-1}RQ^{-1}, \tag{4.3.11}$$

where $M = Q^{-1}P$ and $N = -P^TQ^{-1}$.

To ensure that the fast ADI method for solving (4.3.11) has the optimal complexity, we need the Sylvester matrix equation (4.3.11) to satisfy the properties P1, P2 and P3 discussed in Chapter 3. But, the matrices M and N in (4.3.11) are not normal matrices, which implies that the fast ADI method cannot be directly used. Since M and $N = -M^T$ are pentadiagonal matrices with zeros on the sub-diagonal and super-diagonal, it implies that there exists a diagonal matrix $Q^{1/2}$ for which $\widetilde{M} = Q^{1/2}MQ^{-1/2}$ and $\widetilde{N} = Q^{-1/2}NQ^{1/2} = -\widetilde{M}^T = -\widetilde{M}$ are real symmetric pentadiagonal matrices.

Substituting $M = Q^{-1/2}\widetilde{M}Q^{1/2}$ and $N = Q^{1/2}\widetilde{N}Q^{-1/2}$ in (4.3.11), we have

$$Q^{-1/2}\widetilde{M}Q^{1/2}Q^{-1/2}VQ^{1/2} - Q^{-1/2}VQ^{1/2}Q^{1/2}\widetilde{N}Q^{-1/2} = Q^{-1}RQ^{-1}$$

and therefore,

$$Q^{-1/2}\widetilde{M}VQ^{1/2} - Q^{-1/2}VQ\widetilde{N}Q^{-1/2} = Q^{-1}RQ^{-1}. \tag{4.3.12}$$

Multiplying each term of (4.3.12) on the left and right by $Q^{1/2}$ yields

$$\widetilde{M}VQ - VQ\widetilde{N} = Q^{1/2}\left(Q^{-1}RQ^{-1}\right)Q^{1/2}. \tag{4.3.13}$$

Putting $Z = VQ$ in (4.3.13), we have

$$\widetilde{M}Z - Z\widetilde{N} = Q^{1/2}\left(Q^{-1}RQ^{-1}\right)Q^{1/2}. \tag{4.3.14}$$

Hence, in place of solving (4.3.11), we solve the Sylvester equation

$$\widetilde{M}Z - Z\widetilde{N} = \widetilde{F}, \qquad Z = Q^{1/2}UQ^{1/2} \text{ and } \widetilde{F} = Q^{1/2}\left(Q^{-1}RQ^{-1}\right)Q^{1/2}, \tag{4.3.15}$$

and U is obtainable through $U = Q^{-1/2}ZQ^{-1/2}$. Now, we proceed to verify that the properties P1, P2 and P3 hold for (4.3.15).

P1: $\widetilde{M}$ and $\widetilde{N}$ are real and symmetric which implies that these matrices are normal.

P2: The eigenvalues of $\widetilde{M}$ and $\widetilde{N}$ are contained in the intervals $\left[\frac{-1}{2}, \frac{-1}{2n^4}\right]$ and $\left[\frac{1}{2n^4}, \frac{1}{2}\right]$ respectively (Fortunato and Townsend, 2017).

P3: For any $p, q \in \mathbb{C}$, the linear systems $(\widetilde{M} - pI)x = b$ and $(\widetilde{N} - qI)x = b$ are pentadiagonal matrices with zero sub-diagonal and super-diagonal.

Substituting the bounds in P2 in (3.0.3), (3.0.4) and (3.0.5) and doing Taylor's expansion to substitute into (3.0.7), we have

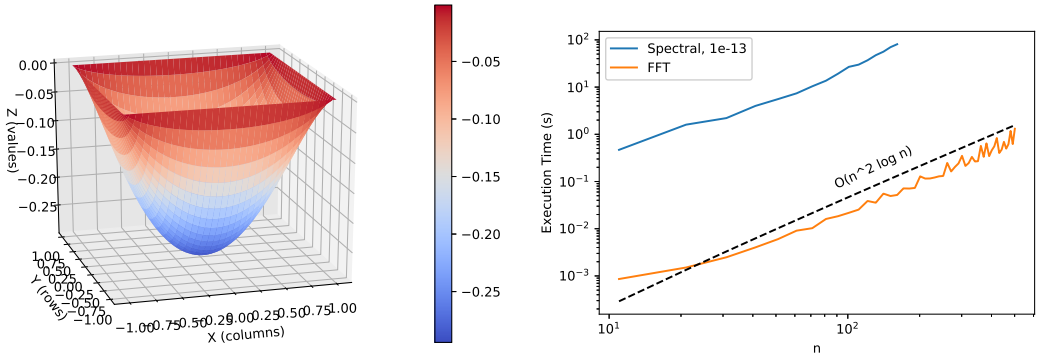
$$W = \left\lceil \frac{\log(120n^4)\log(1/\varepsilon)}{2\pi^2} \right\rceil,$$

which is at most the required ADI iterations to ensure that (4.3.15) is solved within a relative accuracy $0 < \varepsilon < 1$. Since P3 holds, the fast ADI method solves (4.3.15) in $\mathcal{O}(n^2 \log n \log(1/\varepsilon))$ operations, and an additional $\mathcal{O}(n^2)$ operations to obtain U from Z. If we take the coefficient transforms into account, the overall complexity for our optimal complexity Poisson solver is $\mathcal{O}(n^2(\log n)^2 \log(1/\varepsilon))$.

4.4 Numerical Experiment Using the Fast Spectral Method

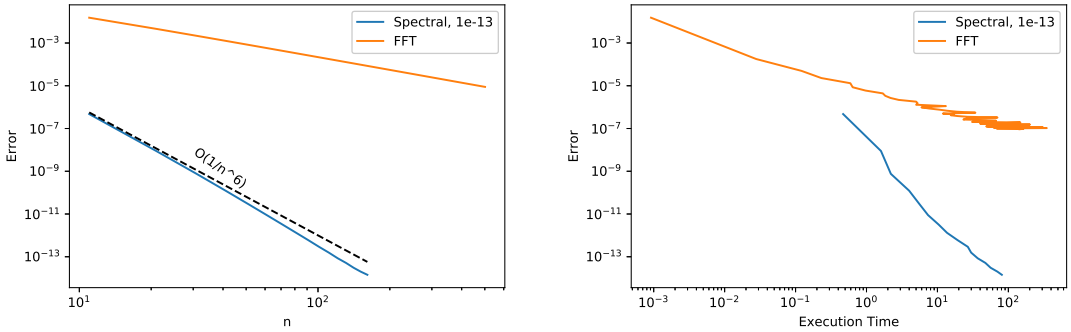
In this section, we perform a numerical experiment for the $2D$ Poisson problem (2.2.1) where $f(x, y) = 1$ using the fast spectral method solver.

We demonstrate the execution time of the optimal complexity spectral method in comparison to the FFT method discussed in Chapter 2. In addition, we plot absolute error against n as n increases and absolute error against time as n increases. These plots are presented in Figure 4.1 and 4.2.



(a) *Plot of the solution u using the spectral method.* (b) *Execution time against n plot as n increases.*

Figure 4.1: (a) *Solution u to the $2D$ Poisson problem (2.2.1) where $f(x, y) = 1$ using optimal complexity spectral method with $n = 100$;* (b) *Execution times for the FFT and optimal complexity spectral method (for $\varepsilon = 1e-13$) with $11 \leq n \leq 161$. We observe from the numerical experiments that for same n, the spectral method takes longer.*



(a) *Error against n plot as n increases.* (b) *Error against execution time plot as n increases.*

Figure 4.2: (a) *Absolute error incurred using the FFT for $11 \leq n \leq 1001$ and optimal complexity spectral methods for $11 \leq n \leq 161$. The convergence of the spectral method for this problem is algebraic instead of geometric. This convergence behaviour is due to the fact that the corners of the square domain introduce singularities in the higher derivatives*; (b) *Execution time against absolute error for FFT for $11 \leq n \leq 4911$ and optimal complexity spectral methods for $11 \leq n \leq 161$. We observe from the numerical experiments that the spectral method gives a better accuracy*.

4.5 Summary and Conclusion of Numerical Experiments

We have been able to apply the ADI algorithm in Chapter 3 to derive an optimal complexity spectral method and we implemented this method to solve the Poisson problem (2.2.1). In the plots presented in Figure 4.1 and Figure 4.2, we considered $n \in [11, 501]$ for the FFT method and $n \in [11, 161]$ for the spectral method.

It is observable from Figure 4.1b, Figure 4.2a, and Figure 4.2b that if we want low accuracy in the fastest time, then we should use the FFT solvers. But, if we prefer high accuracy, then we should use spectral methods.

5. Conclusion and Future Work

5.1 Conclusion

In the course of this study, our focus has been on numerical methods for solving the 2D Poisson equation on a square with zero homogeneous Dirichlet boundary conditions.

We have discussed the linear system of the Kronecker form, eigenvalue decomposition, FFT which are regarded as direct methods and iterative nethods comprising of the ADI and optimal complexity spectral methods. These methods have been used to obtain a numerical solution to the 2D Poisson equation.

The numerical experiments we performed showed that solving the 2D Poisson problem using the finite difference and FFT methods will guarantee low accuracy in the fastest time. But, if our interest is to get high accuracy then we should use the spectral method.

5.2 Future Work

This study has considered the 2D Poisson problem on a square using zero homogeneous Dirichlet boundary conditions. A possible extension to this study is to find a fast method to solve the 2D Poisson problem using non-zero Dirichlet conditions. One way to tackle such a nonhomogeneous problem is to make it homogeneous by moving the boundary conditions to the right hand side, i.e.,

i. we calculate the coefficients W_{BC} of a function V_{BC} which satisfies the Dirichlet conditions;

ii. we calculate the Laplacian of the function V_{BC};

iii. we solve the revised equation $\nabla^2 V_{\mathsf{RHS}} = f - \nabla^2 V_{\mathsf{BC}}$ with zero homogeneous Dirichlet conditions for the coefficients W_{RHS};

iv. we then obtain the solution as $W = W_{\mathsf{RHS}} + W_{\mathsf{BC}}$.

More general boundary conditions, such as Neumann or Robin are still an open problem.

Another interesting extension to this study is to consider Poisson problem using higher order finite difference stencils and methods such as multigrid and fast multipole.

Acknowledgements

It does not, therefore, depend on human desire or effort, but on God's mercy (Romans 9:16). My profound gratitude goes to Almighty God for the grace, strength, wisdom and knowledge He has bestowed on me for the success of this work.

A sincerely appreciation goes to my supervisor Professor Nick Hale for his relentless effort and time expended to guiding me through the period of this project. I am indeed grateful for his wonderful contribution and I consider it a privilege working with him.

Special thanks to the entire staff of AIMS, South Africa for this wonderful opportunity. In particular, I appreciate my tutor Dinna Ranirina for her support throughout this project. I also appreciate my colleagues for the atmosphere of love we have shared together.

To my family, friends and dear friend Oluwabusayo who have supported and encouraged me through out my time at AIMS, I am indeed grateful. God bless you all.

References

Abramowitz, M. and Stegun, I. A. *Hanbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables*, volume 9. Applied Mathematics, National Bureau of Standards, 1972.

Bartels, R. Algorithm 432, Solution of the Matrix Equation AX+XB=C. *Comm, Ass, Computer Machinery*, 15:820–826, 1972. URL https://ci.nii.ac.jp/naid/10015639640/en/.

Beckermann, B. and Townsend, A. On the singular values of matrices with displacement structure. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1227–1248, 2017.

Benner, P., Li, R.-C., and Truhar, N. On the ADI method for Sylvester equations. *Journal of Computational and Applied Mathematics*, 233(4):1035–1045, 2009.

Britanak, V., Yip, P. C., and Rao, K. R. *Discrete cosine and sine transforms: general properties, fast algorithms and integer approximations*. Elsevier, 2010.

Burden, L. L. and Faires, J. D. *Numerical analysis (9th edition)*. Brooks/Cole, Cengage Learning, 20 Channel Center Street Boston, MA02210, USA, 2011.

Canuto, C., Hussaini, M., Quarteroni, A., and Zang, T. Spectral methods: Fundamentals in single domains. 2006. *Sci. Comput. Ser*, 2006.

Chapra, S. C. and Canales, R. P. *Numerical Methods for Engineers (Sixth Edition)*. McGraw-Hill, 1221 Avenue of the Americas, New York, NY 10020, 2010.

Datta, B. N. *Numerical Linear Algebra and Applications*. Society for Industry and Applied Mathematics, Philadelphia, 2010.

Datta, S. *Quantum Transport: Atom to Transistor, 2nd Edition*. Cambridge University Press, 2005.

Dedner, A. Computational PDEs. Available from https://warwick.ac.uk/fac/sci/maths/people/staff/andreas_dedner/lecturematerial/notesma3ho.pdf, 2011.

Feynman, R. P., Leighton, R. B., and Sands, M. *The Feynman Lectures on Physics: Definitive Edition*. Pearson Addison-Wesley, 2005.

Fletcher, C. A. Computational galerkin methods. In *Computational Galerkin Methods*, pages 72–85. Springer, 1984.

Fogolari, F., Brigo, A., and Molinari, H. The Poisson-Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *Journal of Molecular Recognition*, 15(6):377–392, 2002.

Fortunato, D. and Townsend, A. Fast Poisson solvers for spectral methods. *arXiv preprint arXiv:1710.11259*, 2017.

Fuka, V. A free parallel fast Poisson solver. *Applied Mathematics and Computation*, 267:356–364, 2015.

Golub, G. H. and Loan, C. F. V. *Matrix Computation, Third Edition*. The John Hopkins Univeristy Press, 1996.

Gottlieb, D. and Orszag, S. A. *Numerical analysis of spectral methods: theory and applications*, volume 26. SIAM Journal of Numerical Analysis, 1977.

Hall, L. M. Special functions. *Available in: http://web. mst. edu/~ lmhall/SPFNS/spfns. pdf*, 1995.

Iserles, A. *A First Course in the Numerical analysis of Differential Equations*. Press Syndicate of the University of Cambridge, 2009.

Jumat Sulaimon, M. O. and Khatim, M. Red-black half-sweep iterative method using triangle finite element approximation dor 2d poisson equations. *International Conference on Computational Science*, pages 326–333, 2007.

Koekoek, R. and Swarttouw, R. F. The Askey-scheme of hypergeometric orthogonal polynomials and its q-analogue. *arXiv preprint math/9602214*, 1996.

Komla Domelevo, P. O. A Finite Volume Method for the Laplace Equation on Almost Arbitrary Two-DIimensional Grids. *Mathematical Modelling and Numerical Analysis*, 39(6):1203–1249, 2005.

Kreiss, H.-O. and Oliger, J. Stability of the Fourier method. *SIAM Journal on Numerical Analysis*, 16 (3):421–433, 1979.

Lebedev, V. On a Zolotarev problem in the method of alternating directions. *USSR Computational Mathematics and Mathematical Physics*, 17(2):58–76, 1977.

Lu, A. and Wachspress, E. L. Solution of Lyapunov equations by alternating direction implicit iteration. *Computers & Mathematics with Applications*, 21(9):43–58, 1991.

Olver, F. W., Lozier, D. W., Boisvert, R. F., and Clark, C. W. *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press, 2010.

Peaceman, D. W. and Rachford, H. H., Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, 3(1):28–41, 1955.

Randall J, L. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industry and Applied Mathematics, Philadelphia, 2007.

Rønquist, E. M. The poisson problem in $\mathbb{R}^2$:diagonalization methods. Available from https://www.math.ntnu.no/emner/TMA4205/2014h/notes/Poisson2D_diag.p, 2007.

Sabino, J. Solution of large-scale Lyapunov equations via the block modified Smith methods. Technical Report 3971, Rice University, Houston, Texas, 2006.

Strang, G. Finite differences and fast poisson solvers. Available from https://ocw.mit.edu/courses/mathematics/18-086-mathematical-methods-for-engineers-ii-spring-2006/readings/am35.pdf, 2006.

Styan, G. P. Hadamard products and multivariate statistical analysis. *Linear algebra and its applications*, 6:217–240, 1973.

Su, Y. *Collocation Spectral Methods in the solution of Poisson Equation*. Msc., The University of British Columbia, 1998.

Szego, G. *Orthogonal polynomials*, volume 23. American Mathematical Soc., 1939.

Trefethen, L. N. and Bau III, D. *Numerical linear algebra*, volume 50. SIAM, 1997.

Vasileska, D. Numerical Solution of Poisson's Equation. Available from https://nanohub.org/resources/1542/download/numericalanalysis_ppt.pdf, 2010.

Wachspress, E. L. Iterative solution of the Lyapunov matrix equation. *Applied Mathematics Letters*, 1 (1):87–90, 1988.