Finite Difference Method for Solving Poisson's Equation in Two Dimensions with Alternating Direction Implicit Method

Bedru Yusuf Nega
Metu College of Teachers Education, department of Mathematics, Metu, Ethiopia

Ahmed Buseri Ashine
Madda Walabu University, department of Mathematics, Bale Robe, Ethiopia

Abe Nura Ware
Arsi University, department of Mathematics, Asella, Ethiopia

Abstract

This paper presents some numerical techniques for the solution of two dimensional Poisson's equations. The discrete approximation of Poisson's equations is based on Finite difference method over a regular domain $\mathbb{R}$. In this research five point difference approximations is used for Poisson's equation .To solve the resulting finite difference approximation basic iterative methods; Jacobi, Gauss-Seidal and successive over relaxation (SOR) have been used. Several model problems of Poisson's equations are solved by the iterative methods to identify the efficiency of the iterative methods. And we study the convergence, consistence and stability of the schemes. Alternating direction implicit (ADI) method is a finite difference method (FDM) to solve partial differential equations (PDEs). We employ ADI method to solve Poisson's equations for dirichlet boundary conditions. The discretized equation is modified near the boundaries to incorporate arbitrary domain shapes. The ADI method performs much better than traditional iterative methods because of usage of efficient tri-diagonal solvers.

Keywords: Iteration, Discretized , Finite Difference method (FDM), Alternating direct implicit method (ADI), Poisson's equation, stability and convergence.
DOI: 10.7176/APTA/85-01
Publication date: November 30th 2021

1. Introduction

The finite difference method is one of the several techniques for the numerical solution of partial differential equations. The method is based on discrete approximation of the partial derivatives in partial differential equations obtained by Taylor's expansion near the point of interests. In this chapter, the finite difference method for the solution of the elliptical partial differential equation which is a Poisson's equations in two dimensions is discussed**.** The two dimensional Poisson's equation is used in various engineering applications, for example calculation of heat flow, the charge distribution and magnetic field in a domain. The Poisson equation is popularly called as Heat equation.

Poisson's equation is an elliptic linear in homogeneous partial differential equation of the second order. It is given by $\nabla^2 U = -f$ where ∇ stands for the Laplacian operator, and f is called as load /source function. The solution of the equation U is the unknown scalar potential function.

Scientific computation plays an important role in natural science, engineering and technology. It has become an indispensable tool in many areas. The Poisson equation is an essential partial differential equation in fluid dynamics and electromagnetic. Due to its analytical solution of Poisson's equation is difficult to obtain, an alternative method is to find its numerical solution. A classical method to get the numerical solution of the Poisson's equation is the standard five points or nine point difference scheme. This is simple to use and a second order accurate. In order to enhance the accuracy of the difference scheme it is often required to have finer mesh split or to use more complex difference scheme, which will result in much more computing time and larger amount of data storage. Moreover, as the scale of data increases the dimension of the coefficient matrix of the linear equations will increase and the numerical solution will become less stable. To acquire good experimental data usually need compromise between the calculation efficiency and data storage capacity.

In the numerical solution of the two dimensional Poisson's equation, the function value often fluctuates considerably in areas with a higher absolute value of the derivative (if it exists) the mesh accuracy needs to be higher.

Poisson's equation is a second order partial differential equation (PDE) that appears in many areas of science such as electricity, fluid flow and steady heat conduction. Solution of this equation in a domain requires the specification of certain conditions that the unknown function must satisfy at the boundary of the domain.

When the function itself is specified on a part of the boundary we call that part, the Dirichlet boundary; when the normal derivative of the function is specified on a part of the boundary, we call that part the Neumann boundary. In a problem the entire boundary can be Dirichlet or apart of boundary can be Dirichlet and the rest Neumann.

A problem with Neumann condition specified on the entire boundary does not have a unique solution. In some problems a linear combination of the function and its normal derivatives are specified such situations are called Robin boundary.

The general form for linear second order partial differential equations (PDEs) of a second order in more than two independent variables x ,y.

$$AU_{xx} + BU_{xy} + CU_{yy} + DU_X + EU_y + FU + G = 0 \text{ .... (1)}$$

where the coefficients A, B, C, D, E, F&G are constants or functions of the independent variables x & y.

The first three parts are called principle part of PDEs contains the second derivatives. The nature of the general solution to the equation (1) is determined by principle part. Indeed, the coefficients of them can be used to classify PDEs. The classification is based on the sign of the discriminate $B^2 - 4AC$ as follows.

If $B^2 - 4AC > 0$ the PDEs is called hyperbolic. Example; the wave equation has A=1, B=0, C= -1 therefore, it is Hyperbolic.

If $B^2 - 4AC = 0$ the PDEs is called parabolic. Example; the heat equation has A=1, B=0, C=0 therefore, it is parabolic.

If $B^2 - 4AC < 0$ the PDEs is called elliptic. Example; the Laplace & Poisson's equations have A=1, B=0, C=1 so they are elliptic.

Now the finite difference method for Poisson's equation in 2 dimensions is given by $U = U(x, y)$

$$\nabla^2 U = U_{xx} + U_{yy} = -f(x, y) \text{ .... (2)}$$

In Cartesian coordinate system (R^3), the Poisson's equation is given by

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -f(x, y, z)$$ here x, y, z are the independent space dimensions.

When the Poisson's equation is satisfied by a scalar potential in a given domain Ω, we can find the scalar potential inside the domain by solving the Poisson's equation with the help of boundary conditions prescribed at the boundary $\partial\Omega$.

The two dimensional $(2D)$ Poisson's equation can be written in the form

$$\boldsymbol{U_{xx}(x, y) + U_{yy}(x, y) = -f(x, y), \ \ (x, y) \in \Omega}.$$

Where, Ω is a rectangular domain or union of rectangular domains with suitable boundary conditions defined on Ω. The solution $U(x, y)$and forcing function $f(x, y)$ are assumed to be sufficiently smooth and have the necessary derivatives up to certain orders. A second order accurate solution can be computed by applying standard second order central difference operators $U_{xx}(x, y)$ and $U_{yy}(x, y)$ in the above equation**.**

Higher order (more than two) accurate discretization methods need more complex procedure than the second order accurate discretization method to compute the coefficient matrix but they usually generate linear systems of much smaller size, compared with that from the lower order accurate discretization method.

The alternating direction implicit (ADI) method was introduced by Peaceman and Rachford [1], allows the construction of very efficient implicit schemes. The ADI schemes primates the use of centered difference approximation for the convective terms which gives a truncation error of higher order than the one results from one sided order difference approximation used to stabilize some explicit schemes. The ADI method presents the following advantages; second order accuracy in time and space, good stability properties and easy solution by inversion of tri-diagonal matrices. We shall discuss on the stability, convergence and consistency of the FDM and the convergence of ADI and the basic iterative methods. In most of the literature reviews, works done on FDM solutions of Poisson's equation [2], [3] and [4] lacks more application, discussions of stability, convergence and consistence of the FDM, comparison of the efficiency of iterative methods to improve the solution and the convergence analysis of ADI. W.F. Ames [5] presented the principal idea of using FDM to solve PDE's in irregular domains. S.Li & W.K.Liu [6] proposed the main objective of the generalized finite difference method which is approximate the spatial derivatives for a differentiable function in terms of its value at some randomly distributed nodes. The generalized finite difference method often used in mesh less method which is suitable for any geometry of the domain. J.J .Benito [7] employed Generalized Finite Difference Method to solve elliptic and parabolic equations. G.poplau applied a five point's finite difference method to solve the Poisson's equation for studying charge of particles in an accelerator. J.Izadian [8] has proposed a five points Generalized Finite Difference Method to solve the Poisson's equation on irregular 2 Dimensions Domains.

1.1 Preliminaries

Definitions of Finite Difference Method

The Finite Difference Method is one of the most powerful and widely used approaches to the solution of PDEs, due to its good accuracy and flexibility [9] However, the FDM can be enormously computationally expensive, especially when the number of grid points become large. Much effort has been made to find an efficient solution to FDM solutions for PDEs. Some earlier efforts for discrete elliptic equations were reviewed. These include the solution of Poisson's equation implemented on an Illiac IV in 1972 [10].

In 1960, Finite element methods (FEM) were applied for numerical solution of ordinary differential equation (ODE's) and (PDE's). The FDM and FEM, a mesh based is more suitable when a mesh is regular. P.Jenson [11] employed FDM with fully arbitrary meshes. N.perrone and R. kaos [12] formulated a two dimensional FDM capable of using irregular meshes. Recently the various techniques similar to FDM have been proposed to solve PDE's on irregular domains.

The Finite difference Method is one of the several techniques for obtaining numerical solutions to boundary value problems of ordinary differential equation (ODE) and partial differential equation (PDE).

Suppose the rectangle is described by $R = \{a \le x \le b, c \le y \le d\}$ we will divide R into sub rectangles. If we have m subdivisions in the x direction and n subdivisions in the y direction, then the step size in the x and y directions respectively are

$h = \frac{b-a}{m}$ and $k = \frac{d-c}{n}$.

Using Taylor's series we apply

$$U_x = \frac{U_{i+1,j} - U_{i,j}}{h} \text{ ....forward difference method.}$$

$$U_x = \frac{U_{i,j} - U_{i-1,j}}{h} \text{ ....backward difference method.}$$

$$U_x = \frac{U_{i+1,j} - U_{i-1,j}}{2h} \text{ ....Central difference method.} \qquad \text{where w } h = \Delta x$$

$$U_y = \frac{U_{i,j+1} - U_{i,j}}{k} \text{ ....forward difference method.}$$

$$U_y = \frac{U_{i,j} - U_{i,j-1}}{k} \text{ ....backward difference method.}$$

$$U_y = \frac{U_{i,j+1} - U_{i,j-1}}{2k} \text{ ....central difference method} \qquad \text{where } k = \Delta y$$

And From central difference method for 2 dimensions we have

$$U_{xx} = \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} \quad \text{and}$$

$$U_{yy} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{k^2}$$

Then the finite difference equations for Poisson's equations in 2 dimensions is

$$U_{xx} + U_{yy} = -f(x,y) \;\; for \; (x,y) \; in \; R.$$

by replacing U_{xx} and U_{yy} by their central difference we obtain

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{k^2} = -f(x_i, y_j) = -f_{i,j}$$

For $1 \le i \le m-1 \;\; and \;\; 1 \le j \le n-1.$

The boundary conditions are introduced by

$$U_{o,j} = g(a, y_i)\,, \quad U_{m,j} = g(b, y_j)\,, \quad U_{i,o} = g(x_i, c)\,, \quad and \quad U_{i,n} = g(x_i, d)$$

1.2 A Finite Different Scheme

Discretization of the computational domain

We use a rectangular, equidistant grid of $n \times n$ grid points with mesh size $h = \frac{1}{n}$

$$\Omega h := \{(ih, jh) : i, j = 1, \dots \dots \dots., n-1\}$$

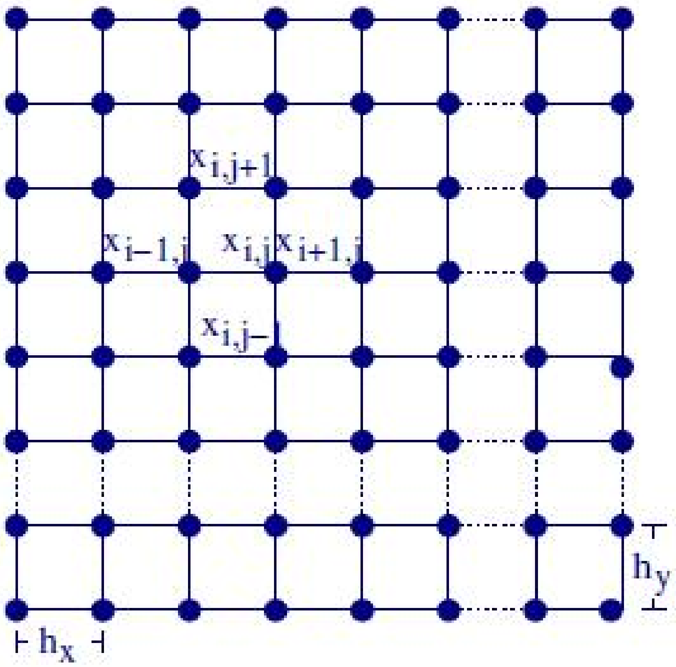


Figure 1 : discretization of domain

We choose $h = h_x = h_y$, where $h_x = \frac{x_{max}-x_{min}}{number\ of\ divisions+1}$ $h_y = \frac{y_{max}-y_{min}}{number\ of\ divisions+1}$

Here x_{max}, x_{min}, y_{max}, y_{min} stand for boundary extremes.

Function U will be computed at grid points $x_{i,j}$. i.e. we have unknowns $u_{i,j} \approx u(x_{i,j})$

2. Governing equations and discretization

2.1 Boundary conditions

For elliptic PDEs there are given boundary conditions where in some property of U is specified $\partial\Omega$ with Poisson's equations. We can define three types of boundary conditions.

a) Dirichlet conditions

When the nodes of grid $U_{i,j}$ is close to the boundary condition, then this node is either close to one boundary node or two boundary nodes (fig. below).

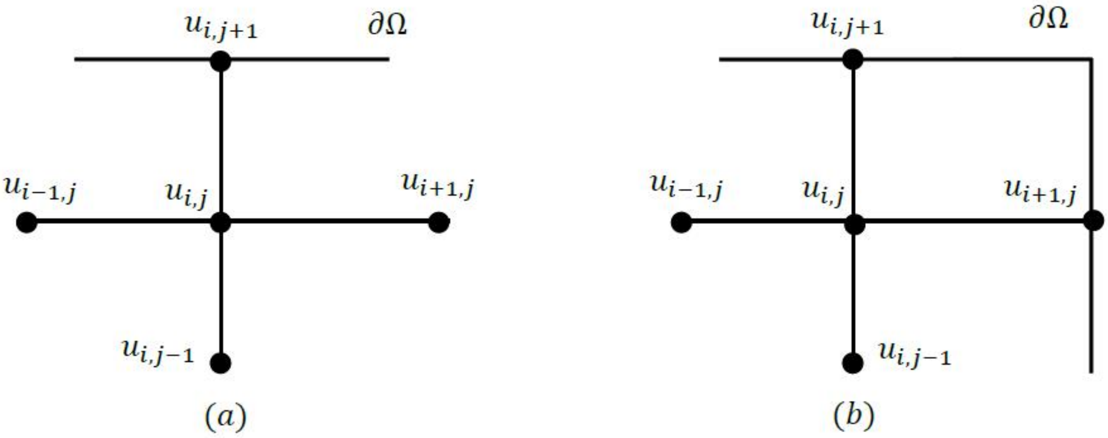


Figure 2 : five Point stencil close to boundary.

In the figure above (a) the boundary node $U_{i,j+1}$ is known value (given), so the five points of finite difference for the Poisson's equation can be written as

$$U_{i+1,j} + U_{i-1,j} + U_{i,j-1} - 4U_{i,j} = h^2 U_{i,j} - U_{i,j+1} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Also in the figure (b) the boundary nodes $U_{i,j+1}$, $U_{i+1,j}$ are the known value (given). So the five points of finite differences for the Poisson's equation can be written as

$$U_{i-1,j} + U_{i,j-1} - 4U_{i,j} = h^2 f_{i,j} - U_{i,j+1} - U_{i+1,j} \text{……………….……… (4)}$$

b) Neumann condition

When $\frac{\partial u}{\partial n}$=$v(x,y)$ is on the boundary $\partial\Omega$ then using five point difference schema, three of the node lie on the boundary line. One inside the boundary at $U_{i-1,j}$ and the fifth at $U_{i+1,j}$ we consider fictitious grid point outside the domain as shown in figure.

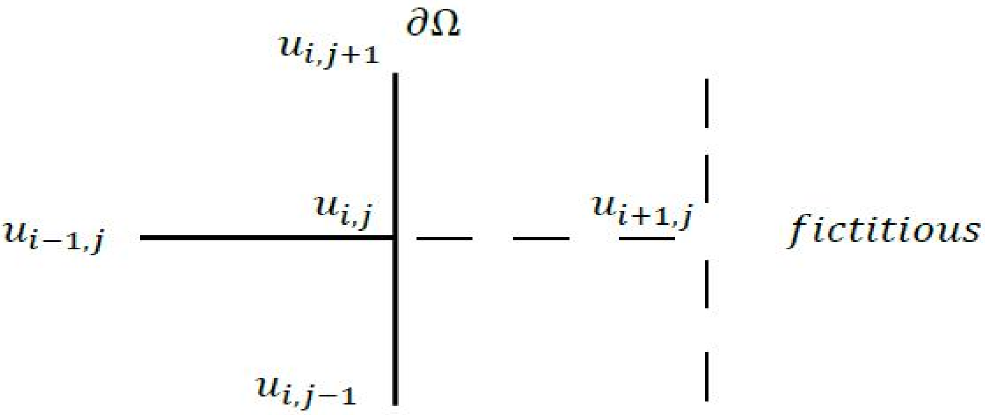


Figure 3 : Fictitious grid point outside the domain ∂Ω.

To find the node $U_{i,j}$ we use the central difference approximation;

$$v(x,y) = \frac{U_{i+1,j}-U_{i-1,j}}{2h} \text{ …………………………………………….. (5)}$$

We rewrite equation (4.3) as

$$U_{i+1,j} = U_{i-1,j} + 2hV_{i,j} \text{ ………………………………………… (6)}$$

So the five point of finite difference for Poisson's equation can be written as

$$-2U_{i-1,j} - U_{i,j+1} - U_{i,j-1} + 4U_{i,j} = 2hV_{i,j} - h^2 f_{i,j}. \text{..................... (7)}$$

c) Mixed condition

The boundary condition of the form the five point finite difference approximation for Poisson's equation and Laplace equation can be incorporated in to the difference equations for the boundary nodal points by an extension of the methods outlined in the drichlet and Neumann problems.

2.2 Discretized Poisson's Equation

Throughout most of our discussion we will usually be concerned with the basic mathematical problem of Poisson's equation with dirichlet boundary conditions on a two Dimension rectangular domain Ω that is

$$U_{xx} + U_{yy} = -f(x,y)\,, (x,y) \in \Omega \subset R^2;$$

$$U(x,y) = g(x,y), (x,y) \in \partial\,\Omega \text{ ………………………….… (8)}$$

Here $\Omega = [a_x, b_x] \times [a_y, b_y]$ and f and g are given functions assumed to be in $C(\Omega)$ or $C(\partial\Omega)$, respectively. We will employ a standard second order centered discretization.

$$\frac{U_{i-1,j}-2U_{i,j}+U_{i+1,j}}{{h_x}^2} + \frac{U_{i,j-1}-2U_{i,j}+U_{i,j+1}}{{h_y}^2} = -f_{i,j},\ \ i,j = 2,3,\ldots, N_x - 1(N_y - 1) \text{…….. (8a)}$$

$$U_{i,j} = g_{i,j}\ , i = 1\ or\ N_x\ with\ i = 2, \ldots\ldots, N_y - 1\ ,$$

$$j = 1\ or\ N_y\ with\ i = 2, \ldots.N_x - 1 \text{………......... (8b)}$$

For simplicity we will assume uniform grid spacing in each of the separate directions. For the special case of the Laplacian considered here, we calculate h_x and h_y from

$$h_x = \frac{b_x - a_x}{N_x - 1}, \qquad h_y = \frac{b_y - a_y}{N_y - 1}$$

$Set\ h_x = h_y = h$ For most analysis and in this case equation (4.6) collapse to

$$U_{i-1,j} + U_{i,j-1} - 4U_{i,j} + U_{i,j+1} + U_{i+1,j} = h^2 f_{i,j}$$

$$i,j = 2,3 \ldots\ldots, N_x - 1(N_y - 1) \text{ ………………………….……………(9)}$$

With boundary condition as before

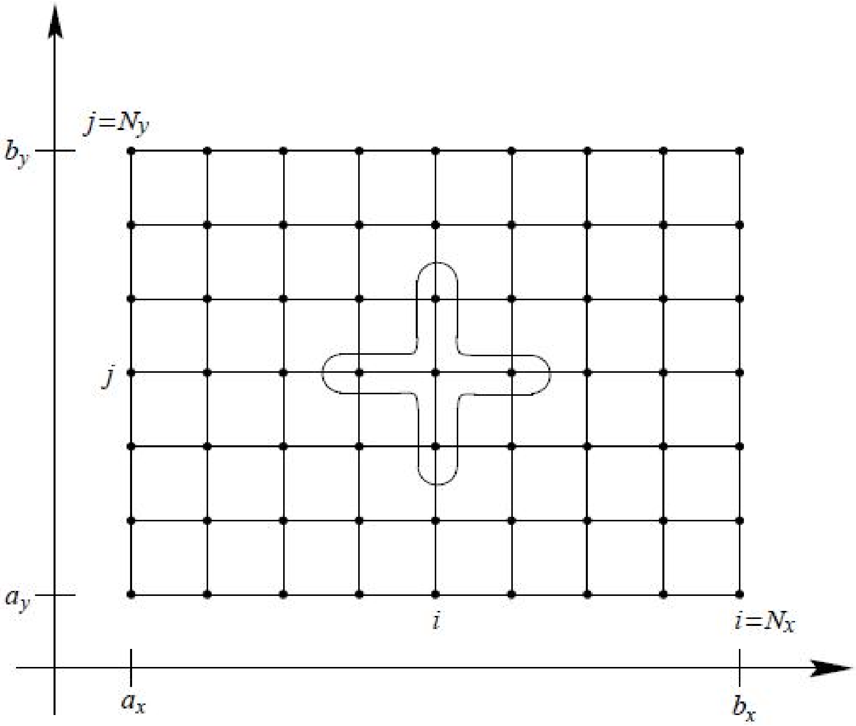


Figure 4: Discretization of the Laplace/Poisson equation on a rectangular grid of $N_x \times N_y$ points.

To obtain the standard five point formula for the Poisson's equation

$U_{xx} + U_{yy} = -f(x, y)\ in\ D \subset \mathbb{R}^2$ With prescribed value of $u(x, y)$ on the boundary ∂D we assume that the domain D is covered by a system of squares with sides of length h parallel to the x&y axes. Using the central difference approximation to the Laplace operator we obtain

$$\frac{1}{h^2}\left(U_{i+1,j} - 2U_{i,j} + U_{i-1,j}\right) + \frac{1}{h^2}\left(U_{i,j+1} - 2U_{i,j} + U_{i,j-1}\right) = -f_{i,j}$$

$$U_{i,j} = \frac{1}{4}\left(U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}\right) + \frac{1}{4}h^2 f_{i,j} \quad \text{……………………………. (10)}$$

Where, $f_{i,j} = f(ih, jh)$. this is known as the five point formula.

2.3 Matrix Formulation

A system of equations is obtained when the difference formula is applied at each interior mesh point. $For\ 1 \leq i\ , j \leq number\ of\ divisions$ We get N_2 number of equations can be put in matrix form as follows $AU = B$

Vector of unknowns= $[U_{11}U_{21,\ldots\ldots}U_{N1}U_{12\ldots\ldots\ldots}U_{N2}, \ldots \ldots U_{NN}]\ T$

Right hand side vector; $B = h^2[f_{11},,,,,,f_{N1},,,,,,,,,,,,,,,,f_{1N},,,,,f_{NN}]T$ the $N^2 \times N^2$ matrix A is given by,

$$A = \left[\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & & & -1 & & & & & & & & & & & \\
-1 & \ddots & \ddots & & & \ddots & & & & & & & & & & \\
 & \ddots & \ddots & -1 & & & \ddots & & & & & & & & & \\
 & & -1 & 4 & & & & -1 & & & & & & & & \\
\hline
-1 & & & & 4 & -1 & & & \ddots & & & & & & & \\
 & \ddots & & & -1 & \ddots & \ddots & & & \ddots & & & & & & \\
 & & \ddots & & & \ddots & \ddots & -1 & & & \ddots & & & & & \\
 & & & -1 & & & -1 & 4 & & & & \ddots & & & & \\
\hline
 & & & & \ddots & & & & \ddots & & & & -1 & & & \\
 & & & & & \ddots & & & & \ddots & & & & \ddots & & \\
 & & & & & & \ddots & & & & \ddots & & & & \ddots & \\
 & & & & & & & \ddots & & & & \ddots & & & & -1 \\
\hline
 & & & & & & & & -1 & & & & 4 & -1 & & \\
 & & & & & & & & & \ddots & & & -1 & \ddots & \ddots & \\
 & & & & & & & & & & \ddots & & & \ddots & \ddots & -1 \\
 & & & & & & & & & & & -1 & & & -1 & 4
\end{array}\right]$$

Figure 5: $\boldsymbol{N^2} \times \boldsymbol{N^2}$ Matrix of A.

2.4 The Finite difference method Approximations, convergence and stability

The Taylor series expansion of a function $u(x, y)$ of two independent variables x and y is

$$u(xi \pm h, yj) = ui \pm 1, j = ui, j \pm h(u_x)i, j + \frac{h^2}{2!}(u_{xx})i, j \pm \frac{h^3}{3!}(u_{xxx})i, j + \cdots \text{……….(11)}$$

$$u(xi, yj \pm k) = ui, j \pm 1, = ui, j \pm k(u_y)i, j + \frac{k^2}{2!}(u_{yy})i, j \pm \frac{k^3}{3!}(u_{yyy})i, j + \cdots \text{……..(12)}$$

Where $u_{ij} = u(x, y), u_{i\pm1,j} = u(x \pm h, y), and\ u_{i,j\pm1} = u(x, y \pm k)$ we choose a set of uniformly spaced rectangles with vertices at $P_{i,j}$ with coordinates (ih, jk) where i, j are positive or negative integers or zero, as shown in fig 6.

We denote $u(ih, jk)$ by $u_{i,j}$.

Using the above Taylor series expansion we write approximate expressions for u_x at the vertex $P_{i,j}$ in terms of $u_{i,j}, u_{i\pm1,j}$.

$$U_x = \frac{1}{h}[u(x + h, y) - u(x, y)] \sim \frac{1}{h}(u_{i+1,j} - u_{i,j}) + o(h) \text{……………..……..………(13)}$$

$$u_x = \frac{1}{h}[u(x, y) - u(x - h, y)] \sim \frac{1}{h}(u_{i,j} - u_{i-1,j}) + o(h) \text{…………………………….. (14)}$$

$$u_x = \frac{1}{2h}[u(x + h, y) - u(x - h, y)] \sim \frac{1}{2h}(u_{i+1,j} - u_{i-1,j}) + o(h^2) \text{………..……..….(15)}$$

These expressions are called the forward first difference, backward first difference and central first difference of u_x respectively. The quantity $o(h)$ or $o(h^2)$ is known as truncation error in this discretization process.

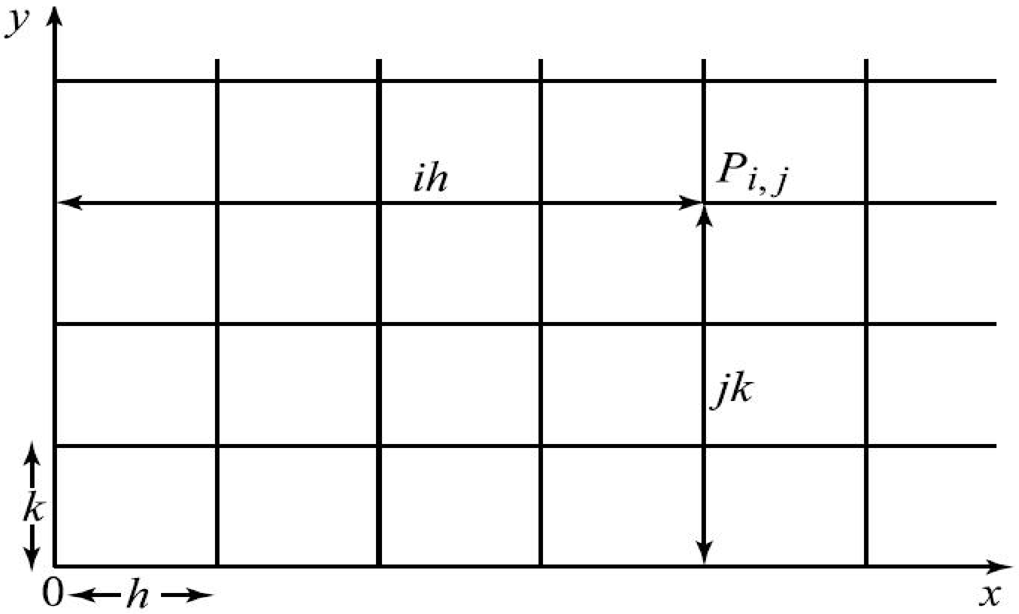


Figure 6: uniformly spaced rectangles

A similar approximate result for u_{xx} at the vertex $p_{i,j}$ is

$$u_{xx} = \frac{1}{h^2}[u(x+h,y) - 2u(x,y) + u(x-h,y)] \sim \frac{1}{h^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right] + o(h^2) \quad \text{...(16)}$$

Similarly the approximate formulas for u_y *and* u_{yy} at $P_{i,j}$ are

$$u_y = \frac{1}{k}[u(x,y+k) - u(x,y)] \sim \frac{1}{k}\left(u_{i,j+1} - u_{i,j}\right) + o(k) \quad \text{.............................. (17)}$$

$$u_y = \frac{1}{k}[u(x,y) - u(x,y-k)] \sim \frac{1}{k}\left(u_{i,j} - u_{i,j-1}\right) + o(k) \text{.............................. (18)}$$

$$u_y = \frac{1}{2k}[u(x,y+k) - u(x,y-k)] \sim \frac{1}{2k}\left(u_{i,j+1} - u_{i,j-1}\right) + o(k^2) \quad \text{................... (19)}$$

$$u_{yy} = \frac{1}{k^2}[u(x,y+k) - 2u(x,y) + u(x,y-k)] \sim \frac{1}{k^2}\left[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] + o(k^2) \quad \text{.. (20)}$$

All these difference formulas are extremely useful in finding numerical solutions of first or second order partial differential equations.

Suppose $U(x,y)$ represents the exact solution of a partial differential equation $L(u) = 0$ with independent variables x and y and $u_{i,j}$ is the exact solution of the corresponding finite difference equation $F\left(u_{i,j}\right) = 0$, then the finite difference scheme is said to be *convergent if* $u_{i,j}$ tends to u as h and k tends to zero.

The difference $d_{i,j} = \left(U_{i,j} - u_{i,j}\right)$ is called the commutative truncation (or discretization) error. This error can generally be minimized by decreasing the grid sizes h and k .however this error depends not only on h and k, but also on the number of terms in the truncated series which is used to approximate each partial derivative.

Another kind of error is introduced when a partial differential equation is approximated by a finite difference equation. If the exact finite difference solution $u_{i,j}$ of the partial differential equation at the grid points $P_{i,j}$ then the value F ($u_{i,j}$) is called the local truncation error at $P_{i,j}$. The finite difference scheme and the partial differential equations are said to be *consistent* if $F\left(u_{i,j}\right)$ tends to zero as h and k tends to zero.

In general, finite difference equations cannot be solved exactly because the numerical computation is carried out only up to a finite number of decimal places. Consequently, another kind of error is introduced in the finite difference solution during the actual process of computation. This kind of error is called the round-off error and it also depends up on the type of computer used.

In practice the actual computational solution is u_{i,j^*} but not $u_{i,j}$ so that the difference $r_{i,j} = \left(U_{i,j} - U_{i,j^*}\right)$ is the round off error at the grid point $P_{i,j}$. In fact this error is introduced to the solution of the finite difference equation by round off errors. In reality the round off error depends mainly on the actual computational process and the finite difference itself.

In contrast to the commutative truncation error, the round-off error cannot be made small by allowing h and k to tend to zero. Thus the total error involved in the finite difference analysis at the point $P_{i,j}$ is given by

$U_{i,j} - u_{i,j^*} = (U_{i,j} - u_{i,j}) + (u_{i,j} - u_{i,j^*}) = d_{i,j} - r_{i,j}$. Usually the discretization error $d_{i,j}$ is bounded when $U_{i,j}$ is bounded because the value of $U_{i,j}$ is fixed for a given partial differential equation with the prescribed boundary and initial data. This fact is used or assumed in order to introduce the concept of stability.

The finite difference algorithm is said to be *stable* if the round off errors are sufficiently small for all $i\ as\ j \to \infty$ that is the growth of $r_{i,j}$ can be controlled. It should be pointed out again that the round off error depends not only on the actual computational process and the type of computer used but also on the finite difference equation itself. Lax (1954) proved a remarkable theorem which establishes the relationship between consistency, stability and convergence of the finite difference algorithm.

Definition 1. A numerical method is called convergent if its global error computed up to a given x satisfies $\lim_{h\to 0}\|\epsilon\| = \lim_{h\to 0}\|y - Y\| \equiv \lim_{h\to 0}\max_{n}|y_n - Y_n| = 0$ note that when taking the limit $h \to 0$,the length $(x - x_0)$ of the computational interval is taken as fixed.

Theorem (**Lax-equivalence theorem**)

The relationship between stability and convergence was hinted at by R. Courant, Frendrichs and Lewy [13] in the 1920's, identified more clearly by Von Neumann [14] in the 1940s and brought into organized form by Lax and Richtmyer [15] in the 1950's the Lax equivalence theorem.

Given a properly posed linear initial value problem and a finite difference approximation to it that satisfies the consistency criterion, stability is a necessary and sufficient condition for convergence.

The idea of the proof

Consistency of the method means that the local truncation error at each step, h τ_n, is sufficiently small so that the accumulated (i.e. global) error, which is on the order of τ_n, tends to zero as h is decreased. Thus

Consistency $\Longrightarrow \|y - Y\|$ is small……………………………………. (21)

Where, as above, Y is the ideal solution of the numerical scheme $F[Y_n, h] = 0$ obtained in the absence of machine round off errors and any errors in initial conditions.

Stability of the method means that if at any given step, the actual solution u_n slightly deviates from the ideal solution Y_n due to the round-off error, and then this deviation remains small and does not grow as n increases. Thus

Stability$\Longrightarrow \|Y - U\|$ is small……………………………………………….. (22)

Equations (4.19) and (4.20) together imply that the maximum difference between the actual computed solution u_n and the exact solution y_n also remains small. Indeed

$\|y - U\| = \|(y - Y) + (Y - U)\| \le \|(y - Y)\| + \|(Y - U)\|$…………………….. (23)

This must be small because each term on the right hand side is small. The fact that the left hand side of the above equation is small means, by definition 1, that the method is convergent.

3. Methods

3.1 Iterative Methods

To solve the above system of equations usual elimination methods cannot be applied. Especially when the number of divisions becomes more, the amount of storage required to store intermediate matrices produced during the computation by elimination methods become very large. Iterative methods are employed to solve system of equations. This is because they take advantage of the sparsity structure of the A matrix. They require storage of just the same as matrix A (fig 5) and can be terminated when we get the solution with sufficient amount of accuracy.

In an iterative, we begin with an initial vector $X^{(0)}$, and generate a sequence of vectors

$X^{(0)} \longrightarrow X^{(1)} \longrightarrow X^{(2)} \longrightarrow \cdots$

Which converge towards the desired solution X .Each iteration requires very less amount of work comparable to that of multiplication of A with a vector which is very less when matrix A is sparse. So we can carry out a large number of iterations.

3.1.1 Jacobi Iterative Method

If we now solve equation (9) for $U_{i,j}$ and introduce iteration counters, we obtain the Jacobi iteration for

$$U_{i,j}{}^{(n+1)} = \frac{1}{4}[U_{i-1,j}{}^{(n)} + U_{i,j-1}{}^{(n)} + U_{i,j+1}{}^{(n)} + U_{i+1,j}{}^{(n)} - h^2 f_{i,j}]\forall_{i,j}$$

$n = 0,1,2,3$………………………………………………………………………... (24)

In the index set of equation (9) this set of equation is clearly in fixed point form and thus can be expressed as $U^{n+1} = BU^{n} + k$…………………………………………………………….…. (25)

$where\ B\ is\ the\ jacobi\ iteration\ matrix.$

This is the Jacobi iteration for the Poisson's problem and it can be shown that for this particular problem it converges from any initial guess $U^{(0)}$ (though very slowly).

Unfortunately, although each iteration of the Jacobi method is quick and easy, it is very slow to converge, especially for large grids. It is therefore impractical but it forms the basis for other more useful methods.

For each interior grid point (i,j), $U_{i,j}$ the next iteration (n+1) is found from (24) once an iteration has been completed for all interior grid points we compute the distance b/n vectors $U^{n+1}\ and\ U^{n}$. if $|U^{n+1} - U^{n}| < tol$ Where tol is apre-defined tolerance, the iterations terminate and the solution is U^{n+1} otherwise the iterations continue.

3.1.2 Gauss- Seidal Method

Gauss- seidal method is very similar to the Jacobi method. It uses new values as soon as they are computed. We obtain the Gauss- seidal iteration formula

$$U_{i,j}{}^{n+1} = \frac{1}{4}\left[U_{i-1,j}{}^{n+1} + U_{i,j-1}{}^{n+1} + U_{i+1,j}{}^{n} + U_{i,j+1}{}^{n} - h^2 f_{i,j}\right] \ \forall\, i,j \qquad \ldots\ldots\ldots\ldots \quad (26)$$

$n = 0,1,2, \ldots\ldots$ in the index set of equation.

The convergence turns out to be somewhat faster (though it is still quite slow).

3.1.3 Successive Over Relaxation Method (SOR)

In this method in order to improve convergence we extrapolate the solution vector obtained from gauss seidal method.

Extrapolation is done in form of weighted average of the previous iteration solution and new solution from Gauss seidal method with the help of relaxation parameter ω.

$X^{(i+1)} = (1-\omega)X_i{}^{(i)} + \omega x \quad where, \omega x$ is obtained using Gauss-seidal method

Or using the method from 4.7

$$U_{i,j}{}^{n+1} = (1-\omega)U_{i,j}{}^{n} + \frac{w}{4}\left(U_{i-1,j}{}^{n+1} + U_{i,j-1}{}^{n+1} + U_{i,j+1}{}^{n} + U_{i+1,j}{}^{n} - h^2 f_{i,j}\right). \ldots\ldots (27)$$

$n = 0,1,2 \ldots\ldots$

The errors in solutions obtained using either the Jacobi or Gauss seidal iterations decrease only slowly, and often in a monotic manner. We can therefore improve on those methods by over correcting our solution at each step using a different formula for $U_{i,j}{}^{n+1}$.

The choice of parameter ω should be between 1&2 so that the convergence is rapid as possible.

The value $\omega = 1$ gives the Gauss seidal method again.

$\omega < 1$ Would produce under relaxation, where we keep a proportion of the old solution.

$\omega > 1$ Produce over relaxation where we actually move further away from the old solution than we would use Gauss seidal.

The best value to use for ω depends on the particular problem being solved, and may also vary as the iterative process proceeds. However, there is no good general method for determining the optimal ω rather than searching by numerical experimentation for an optimal ω. The number of iterations required using SOR is significantly less than for either Jacobi or Gauss seidal and for large grids it is often the most practical of all methods of solution.

3.2 Alternating Direction Implicit method (ADI)

Alternating Direction Implicit (ADI) method is to solve system of equations arising from the Finite Difference discretization of parabolic and elliptic partial differential equations. It was first introduced in mid-1950s Peaceman and Rachford [11].

From Five point approximation for Poisson's equation, we have

$$U_{i+1,j} + U_{i,j+1} + U_{i-1,j} + U_{i,j-1} - 4U_{i,j} = h^2 f_{i,j}$$

$$\left\{\begin{matrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{matrix}\right\} u = h^2 f(x, y)$$

Now we can write

$U_{i-1,j} - 4U_{i,j} + U_{i+1,j} = -U_{i,j-1} - U_{i,j+1} + h^2 f_{i,j}$ Or………………………… (28)

$U_{i,j-1} - 4U_{i,j} + U_{i,j+1} = -U_{i-1,j} - U_{i+1,j} + h^2 f_{i,j}$……………………………… (29)

In the ADI method we proceed by iteration at every mesh point we choose an arbitrary starting value U_{i,j^0} . In each step we compute new values at all mesh points. In one step, we use an iterative formula resulting from (28) and in the next step an iteration formula resulting from (29) and so on in alternating order.

In details, suppose that the approximations $U_{i,j^{(m)}}$ have been computed. Then, to obtain the next approximations $U_{i,j^{(m+1)}}$, we substitute $U_{i,j^{(m)}}$ on the right-hand side of (28) and solve all $U_{i,j^{(m+1)}}$ on the left hand side; that is we use

$U_{i-1,j^{(m+1)}} - 4U_{i,j^{(m+1)}} + U_{i+1,j^{(m+1)}} = -U_{i,j-1^{(m)}} - U_{i,j+1^{(m)}} + h^2 f_{i,j}$ ………… (30)

We use this for a fixed j, that a fixed row j, and for all internal mesh points in this row. This gives a system of N linear algebraic equations in N unknowns; the new approximations of U at these mesh points. Equation (30) involves not only approximations computed in the previous step but also given boundary values. We solve the system (30) tri-diagonal system. Then we go to the next row, obtain another of N equations and solve it and so on until all rows are done. In the next step, we alternate direction, that is we compute the next approximations $U_{i,j^{(m+2)}}$ column by column from the $U_{i,j^{(m+1)}}$ and the given boundary values, using a formula obtained from (29) by substituting the $U_{i,j^{(m+1)}}$ on the right.

$U_{i,j-1^{(m+2)}} - 4U_{i,j^{(m+2)}} + U_{i,j+1^{(m+2)}} = -U_{i-1,j^{(m+1)}} \_ U_{i+1,j^{(m+1)}} + h^2 f_{i,j}$ ……… (31)

For each fixed i, that is for each column. This system of M equations (M= number of internal mesh points per column) in M unknowns which is a tri- diagonal system.

The idea of the ADI method is to execute each iteration in two or three steps (equal number of dimensions). The finite difference formula $(8a)$ is written in such a way that derivatives involving one spatial direction are in left hand side of the equation and remaining terms are carried to the right hand side.

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_x{}^2} = -f(x, y) - \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h_y{}^2} \text{…………………………………} (32)$$

$$\frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h_y{}^2} = -f(x, y) - \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h_x{}^2} \text{………………………………....} (33)$$

Using above equations, each iteration is described in two steps $U^{\left(I+\frac{1}{2}\right)}$ is obtained by using (32) and then $U^{(I+1)}$ is obtained from (33).

$$\frac{U_{i+1,j}{}^{\left(I+\frac{1}{2}\right)} - 2U_{i,j}{}^{\left(I+\frac{1}{2}\right)} + U_{i-1,j}{}^{\left(I+\frac{1}{2}\right)}}{h_x{}^2} = -f(x, y) - \frac{U_{i,j-1}{}^{(I)} - 2U_{i,j}{}^{(I)} + U_{i,j+1}{}^{(I)}}{h_y{}^2} \text{………………} (34)$$

$$\frac{U_{i,j-1}{}^{(I+1)} - 2U_{i,j}{}^{(I+1)} + U_{i,j+1}{}^{(I+1)}}{h_y{}^2} = -f(x, y) - \frac{U_{i-1,j}{}^{\left(I+\frac{1}{2}\right)} - 2U_{i,j}{}^{\left(I+\frac{1}{2}\right)} + U_{i+1,j}{}^{\left(I+\frac{1}{2}\right)}}{h_x{}^2} \text{………....}(35)$$

So in each iteration, the discrete equations are solved first in one spatial direction and then in the other direction also the system of equations should be solved implicitly leading to the terminology alternating direction implicit method (ADI).

3.2.1 Matrix Formation for solving ADI

The difference equation given in (4.33) when applied to each point on the domain gives rise to system of equations which can be given in matrix equations.

$$Hu = B - Vu \text{………………………... ….} (36)$$

$$Vu = B - Hu \text{………………………………} (37)$$

Where

H: Horizontal dimension matrix contains coefficients for unknowns in x direction.

V: vertical dimension matrix contains for coefficients for unknowns in y direction.

B: RHS vector in mathematical terms. $w = \frac{2u_{i,j} - u_{i-1,j} - u_{i+1,j}}{{h_x}^2}$ if w=Hu

$w = \frac{2u_{i,j} - u_{i,j-1} - u_{i,j+1}}{{h_y}^2}$ if w=Vu.

In simple words H and V arise as the central difference approximations of the corresponding terms in the Poisson's equation.

The H matrix for the domain described in the (fig.1) is given by

$$H = \frac{1}{h_x^2} \left[\begin{array}{cccc|cccc|cccc|cccc}
2 & -1 & & & -1 & & & & & & & & & & & \\
-1 & \ddots & \ddots & & & \ddots & & & & & & & & & & \\
 & \ddots & \ddots & -1 & & & \ddots & & & & & & & & & \\
 & & -1 & 2 & & & & -1 & & & & & & & & \\
\hline
-1 & & & & 2 & -1 & & & \ddots & & & & & & & \\
 & \ddots & & & -1 & \ddots & \ddots & & & \ddots & & & & & & \\
 & & \ddots & & & \ddots & \ddots & -1 & & & \ddots & & & & & \\
 & & & -1 & & & -1 & 2 & & & & \ddots & & & & \\
\hline
 & & & & \ddots & & & & \ddots & & & & -1 & & & \\
 & & & & & \ddots & & & & \ddots & & & & \ddots & & \\
 & & & & & & \ddots & & & & \ddots & & & & \ddots & \\
 & & & & & & & \ddots & & & & \ddots & & & & -1 \\
\hline
 & & & & & & & & -1 & & & & 2 & -1 & & \\
 & & & & & & & & & \ddots & & & -1 & \ddots & \ddots & \\
 & & & & & & & & & & \ddots & & & \ddots & \ddots & -1 \\
 & & & & & & & & & & & -1 & & & -1 & 2
\end{array} \right]$$

Figure 7 : H Matrix of Domain Grid

The diagonal entries of H matrix correspond to the current point (xi, yj) for which the equation is being generated. H matrix row contains a sub-diagonal entry for the preceding point ($x = x_{i-1}$) and a super diagonal entry for the next point in x direction. If the adjacent point in a boundary point, then correspondingly sub-diagonal or super diagonal entry may be zero and the right hand side vector is updated with the value at the boundary.

The V matrix for the domain described in figure 1 is given by

$$V = \frac{1}{h_y^2}\left[\begin{array}{c|c|c|c}
\begin{matrix}2 & & \\ & \ddots & \\ & & \ddots \\ & & & 2\end{matrix} & \begin{matrix}-1 & & \\ & \ddots & \\ & & \ddots \\ & & & -1\end{matrix} & & \\ \hline
\begin{matrix}-1 & & \\ & \ddots & \\ & & \ddots \\ & & & -1\end{matrix} & \begin{matrix}2 & & \\ & \ddots & \\ & & \ddots \\ & & & 2\end{matrix} & \begin{matrix}\ddots & & \\ & \ddots & \\ & & \ddots \\ & & & \ddots\end{matrix} & \\ \hline
 & \begin{matrix}\ddots & & \\ & \ddots & \\ & & \ddots \\ & & & \ddots\end{matrix} & \begin{matrix}\ddots & & \\ & \ddots & \\ & & \ddots \\ & & & \ddots\end{matrix} & \begin{matrix}-1 & & \\ & \ddots & \\ & & \ddots \\ & & & -1\end{matrix} \\ \hline
 & & \begin{matrix}-1 & & \\ & \ddots & \\ & & \ddots \\ & & & -1\end{matrix} & \begin{matrix}2 & & \\ & \ddots & \\ & & \ddots \\ & & & 2\end{matrix}
\end{array}\right]$$

Figure 8: V Matrix of Domain Grid

Each row of V matrix corresponds to an equation for a point (xi, yj) in the interior of the domain in the vertical direction. One entry in lower triangular part is non-zero corresponding to the preceding point $y = y_{j-1}$ in the vertical dimension and an entry in upper triangular part is non- zero corresponding to next point $(y = y_{j+1})$ in the vertical dimension. This matrix formulation can also be viewed as the coefficient matrix A described in fig. 5 being split in to parts in the ADI method depending on the number of dimensions of the domain.

For two dimensions,

$$A = H + V$$

The B vector is obtained by evaluation the Poisson load/source function and accumulating the boundary values if the adjacent points to the given point are boundary points.

3.2.2 ADI Method Implementation

Relaxation parameter r is introduced in the equations (37) similar to Gauss seidal method to enhance the convergence .now the steps in an iteration of the ADI method can be summarized as follows:

$$(H + rI)U^{\left(n+\frac{1}{2}\right)} = (rI - V)U^{(n)} + B \text{ ……………………………………….. (38)}$$

$$(V + rI)U^{(n+1)} = (rI - H)U^{\left(n+\frac{1}{2}\right)} + B \text{ …………………………………….. (39)}$$

Each iteration step involves solving a matrix equation which is costly if we employ Gauss-seidal elimination$(o(n^3) where\, n = N^2)$. But we can observe that, H matrix is a tri-diagonal i.e. it contains non zero entries only in diagonal, sub diagonal and super diagonal elements. So to solve system of equations (38) we can employ tri- diagonal solve which involves only o (n) work.

Algorithm tri-diagonal solve can be slightly modified to solve for equations (39) where elimination of non zero lower triangular entry below this diagonal element is done and corresponding row is modified instead of next row. We can observe that this algorithm also involves only $o(n)$ work.

So the implementation of iteration requires only $o(n)$ solving the matrix equations given in the steps of ADI method. But it is still cost lier than classical elimination methods. This may seem to be a disadvantage. But the advantage of ADI method springs from the fact that it converges at much faster rate by clever choice of relaxation parameter r.

3.3 Convergence Analysis for classical iterative methods

Iterative schemes for matrix inversion do not necessary converge so we need to determine conditions for convergence. We would also like to know how fast they converge. We are trying to solve the system of linear equation

A$\underline{\text{U}}$=b……………………………………………………………………... (40)

Where A is an $\boldsymbol{N \times N}$ matrix, $\underline{\text{u}}$ is column vector of N unknowns and $\underline{\text{b}}$ is a column vector of N known constants. It is always possible to scale each equation so that every entry on the main diagonal of A is $\mathbf{1}$. A can then be written as,

$\boldsymbol{A = -L + I - U}$ ……………………………………………………….…. (41)

Where I is the $\boldsymbol{N \times N}$ identity matrix and L and U lower and upper triangular matrices respectively, substituting (41) in to (40) and re arranging gives

$\underline{\text{u}}$= (L+U) $\underline{\text{u}}$+ $\underline{\text{b}}$……………………………………………………………… (42)

3.3.1 Jacobi Iteration

Equation (39) suggests the iterative scheme

$\boldsymbol{u^{m+1} = (L+U)\underline{u}^m + \underline{b}}$ ……………………………………………..….. (43)

This is the Jacobi iteration scheme in matrix form $\boldsymbol{(L+U)}$ is the iteration matrix. In order to analyze the convergence of the Jacobi iteration scheme, we need to look at how the error behaves between iterations. Clearly we want to decrease to zero as iterations continue. The exact solution (40) is $\boldsymbol{\underline{u}}$. Let the error after the $\boldsymbol{m^{th}}$ iteration be $\boldsymbol{\underline{e}^m}$, so

$\boldsymbol{\underline{e}^m = \underline{U}^m} - u$………………………………………………….................. (44)

Note that: $\boldsymbol{\underline{e}^m}$ is a vector in $\boldsymbol{R^N}$ whose components are the errors at each grid points after the $\boldsymbol{m^{th}}$ iteration. Subtracting (42) from (43) gives

$\boldsymbol{\underline{u}^{m+1} - \underline{u} = (L+U)\underline{u}^m + \underline{b} - (L+U)\underline{u} - \underline{b}}$ ……………………………….. (45)

$\boldsymbol{\underline{e}^{m+1} = (L+U)\underline{e}^m}$ ……………………………………………………………. (46)

Writing $\boldsymbol{\underline{e}^0}$ for the initial error in the initial (guessed) values for $\boldsymbol{\underline{u}^0}$ after $\mathbf{1}$ iteration (44) gives

$$\boldsymbol{e^1 = (L+U)\,\underline{e}^0}.$$

A second iteration gives

$$\boldsymbol{e^2 = (L+U)\underline{e}^1}$$

$\boldsymbol{= (L+U)^2\underline{e}^0}$

So after n iteration we have

$$\boldsymbol{e^n = (L+U)^n\underline{e}^0}\text{………………………….. (47)}$$

For convergence of the iterative scheme all components of $\boldsymbol{\underline{e}^n}$ must approach zero in the limit.

$\boldsymbol{i.e}\quad \lim\limits_{n\to\infty} \boldsymbol{e^n} = 0$

$$\lim_{n\to\infty} \boldsymbol{(L+U)^n\underline{e}^0 = 0}$$

Since $\boldsymbol{\underline{e}^0}$ is anon zero constant vectors, we must focus our attention on the behavior of $\boldsymbol{(L+U)^n}$. we assume that $\boldsymbol{(L+U)}$ has n linearly independent Eigen vectors $\boldsymbol{\underline{v1}, \underline{v2}, \underline{v3}, \ldots \ldots \underline{vN}}$ which corresponding Eigen values $\boldsymbol{\lambda_1, \lambda_2, \lambda_3, \ldots . \lambda_{N.}}$ by a standard result from linear algebra $\boldsymbol{(L+U)^n}$ has the same eigen vectors with corresponding Eigen values $\boldsymbol{{\lambda_1}^n, {\lambda_2}^n, \ldots \ldots {\lambda_N}^n}$. the set of Eigen vectors from basis for $\boldsymbol{R^N}$ so for some constants $\boldsymbol{a_i}$ we may write,

$$\boldsymbol{\underline{e}^0 = a_1\underline{v}_1 + a_2\underline{v}_2 + \cdots + a_N\underline{v}_N}$$

$$\boldsymbol{(L+U)^n\,\underline{e}^0 = (L+U)^n\big(a_1\underline{v}_1 + a_2\underline{v}_2 + \cdots + a_N\underline{v}_N\big).}$$

$$\boldsymbol{= (L+U)^n a_1\underline{v}_1 + (L+U)^n a_2\underline{v}_2 + \cdots (L+U)^n a_N\underline{v}_N.}$$

$$\boldsymbol{= a_1(L+U)^n\underline{v}_1 + a_2\,(L+U)^n\underline{v}_2 + \cdots a_N(L+U)^n\underline{v}_N.}$$

$$\boldsymbol{= a_1{\lambda_1}^n\underline{v}_1 + a_2{\lambda_2}^n\underline{v}_2 + \cdots a_N{\lambda_N}^n\underline{v}_N.}$$

This will clearly tend to the zero vectors if and only if,

$|\lambda_i| < 1, for\ i = 1, 2, \dots N.$

Definition 2: the dominant Eigen value of a matrix is the Eigen value with the largest modulus.

Hence we can say that the Jacobi iterative scheme converges if and only if the dominant Eigen value of its iteration matrix has modulus less than 1. We denote the dominant Eigen value of the Jacobi iteration matrix by λ.

3.3.2 Gauss-seidal iteration

Using the previous notation it can be shown that the Gauss-seidal iteration method can be expressed as,

$\underline{u}^{m+1} = U\underline{u}^m + L\underline{u}^{m+1} + \underline{b}$ …………………………………………….. (48)

And a similar analysis to the Jacobi iterative scheme shows that,

$\underline{e}^{m+1} = U\underline{e}^m + L\underline{e}^{m+1}$ …………………………………………... ..(49a)

$\underline{e}^{m+1} = (I - L)^{-1}U\underline{e}^m$ ……………………………….………….… (49b)

$\underline{e}^{m+1} = ((I - L)^{-1}U)^m e^0$ …………………………………………….. (49c)

$(I - L)^{-1}U$ is the Gauss seidal iteration matrix. By an exactly similar analysis to that for Jacobi iteration, the Gauss seidal scheme converges if and only if the dominant Eigen value of its iteration matrix has modulus less than 1.

It can be shown that the dominant Eigen value $= \lambda^2$.

3.3.3 SOR Iterative scheme

A similar analysis to the above shows that the SOR iteration matrix is

$(I - \omega U)^{-1}((1 - \omega)I + \omega L)$ …………………………………………. (50)

It can be shown that its domain Eigen value is

$\frac{2}{1+\sqrt{1-\lambda^2}} - 1$ ……………………………………………………………. (51)

As before the scheme converges if and only if this has modulus less than 1.

A special case for SOR

For a rectangular $p\Delta x\ by\ \ q\Delta y$ computational region the optimal value of the SOR relation parameter can be shown to be

$\omega_o = \frac{2}{1+\sqrt{1-\mu}}$…………………………………………………………..... (52)

Where the dominant Eigen value μ, of the corresponding Gauss seidal scheme is,

$\mu = \frac{\left(\cos\left(\frac{\pi}{p}\right)+\cos\left(\frac{\pi}{q}\right)\right)^2}{4}$…………………………………….……………….. (53)

3.4 Rates of convergence of iterative schemes (ROC)

The rate of convergence of an iterative scheme is a measure of the number of iterations needed to converge to some given tolerance. It turns out that the rate of convergence of an iterative scheme can be defined as

$-loge^{\lambda}$………………….….….…… (54)

Where, λ is the dominant Eigen value of its iteration matrix.

The relative ROC of two schemes with dominant Eigen values λ_1 and λ_2 is $\frac{-log_e\lambda_1}{-log_e\lambda_2}$.. (55)

EXAMPLE

Given that the point Jacobi scheme is convergent with dominant Eigen value λ and the point Gauss-seidal scheme has dominant Eigen value λ^2 and is also convergent by (52).The relative ROC of the Gauss-seidal scheme to the Jacobi scheme is $\frac{-log_e\lambda^2}{-log_e\lambda} = 2.$

i.e. the number of iterations to achieve the same level of accuracy using the Gauss-seidal scheme is approximately half that of the Jacobi scheme.

The relative ROC is not the whole story when comparing iterative schemes. It could be that, an iterative scheme needs many iterations to converge but that each iteration is computationally fast. This could make it a faster than a quick converging but computationally slow scheme.

4. Application

The performance of various iterative methods described above will be tested for the solution of the model problems listed below.

Problem 1

$U_{xx} + U_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y) \;\; \Omega = \{(x,y)/0 < x,y < 1\}$

With $U(x,0) = U(x,1) = U(0,y) = U(1,y) = 0 \in \partial\Omega$

Where $U_{exact} = \sin(\pi x)\sin(\pi y)$

The generated linear systems of the above model problems from the five point discretization are solved using the five point discretization using the following iterative methods.

- Jacobi method
- Gauss seidal method
- Successive over relaxation method (SOR)

The iterative algorithm of the above methods with different values of mesh size

$h = \frac{1}{10}$,$h = \frac{1}{20}$,$h = \frac{1}{40}$

has been computed numerically according to the fixed starting iterative vector and iteration has been terminated according to the following criteria

$$\left\|U^{(k+1)} - U^{(k)}\right\| < \epsilon = 10^{-5}$$

To increase the convergence rates of the SOR methods through the application of the model problems, optimum relaxation parameter has been calculated for different values of mesh size h using the following formula

$\omega_{opt} = \frac{2}{1+\sqrt{1-\beta^2}}$

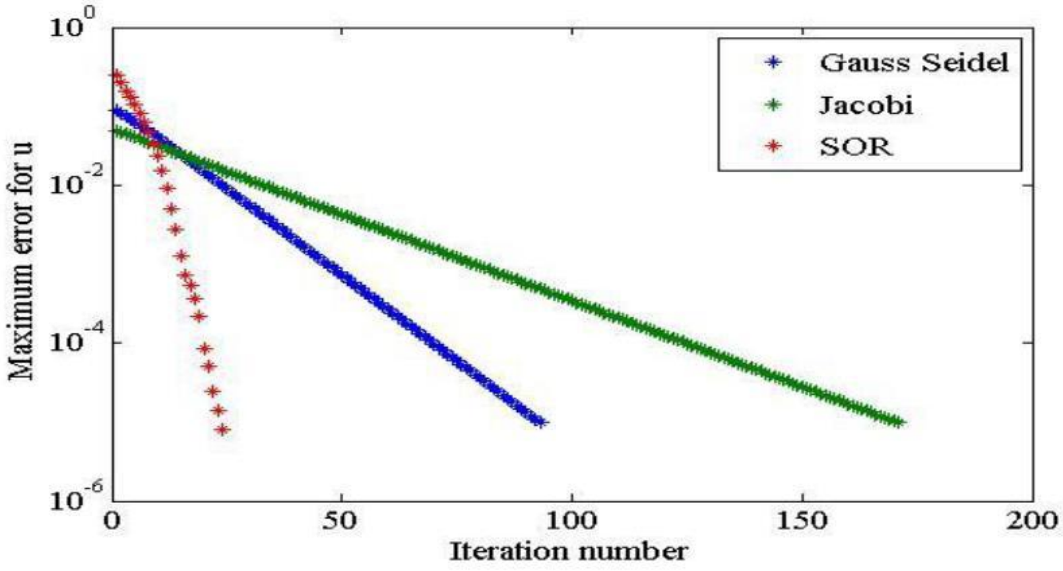
One of the considered comparison factors to evaluate the performance of the methods is the maximum error reduction through the iterative procedure.

Fig (9, 10, and 11) illustrates the maximum error reduction for each iteration methods to solve $problem\ 1$ using the proposed iterative methods with various values of

$h = \frac{1}{10}$,$h = \frac{1}{20}$,$h = \frac{1}{40}$ Respectively. These figures indicate that SOR iterative method requires less iteration than Jacobi and Gauss seidal methods.

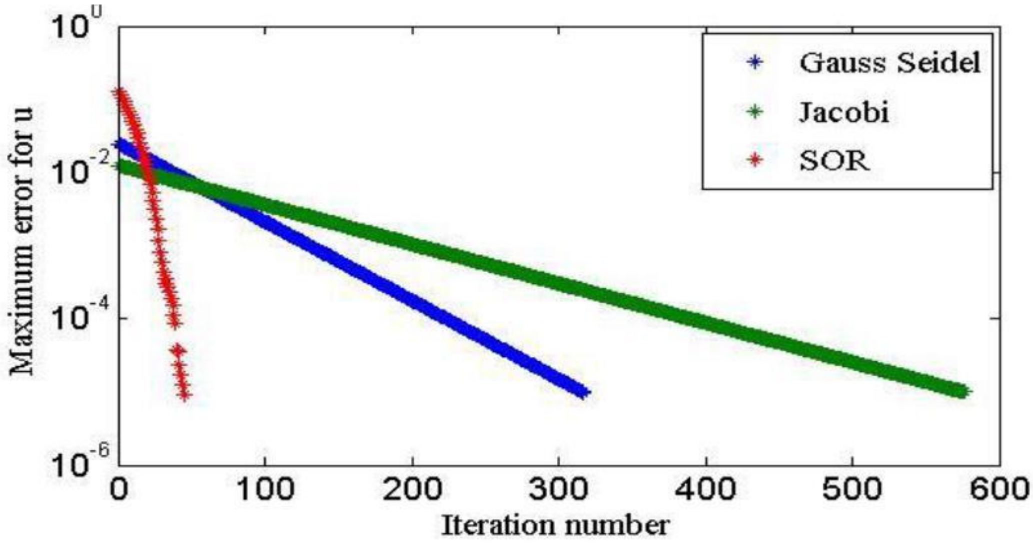
Exact and approximate solutions of the problem with $h = \frac{1}{10}$ are illustrated in Fig (12,13,14). We observed from these figures that proposed iterative methods work well &each iterative method produce a reasonable approximate solution.

Exact and approximate solutions of the iterative methods with $h = \frac{1}{40}$ are illustrated in fig 15.



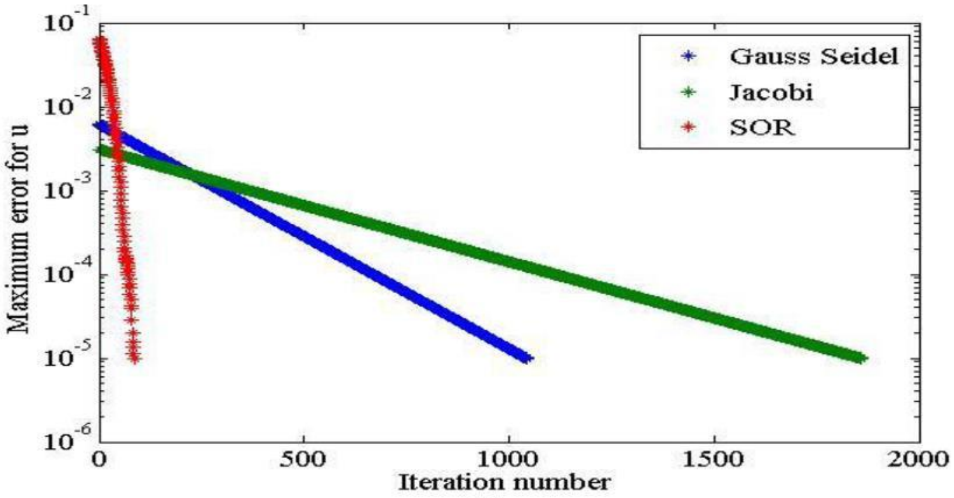
(a)

Figure 9: The Maximum error reduction per each iteration step using Jacobi, Gauss-seidal and SOR methods for $h = \frac{1}{10}$.



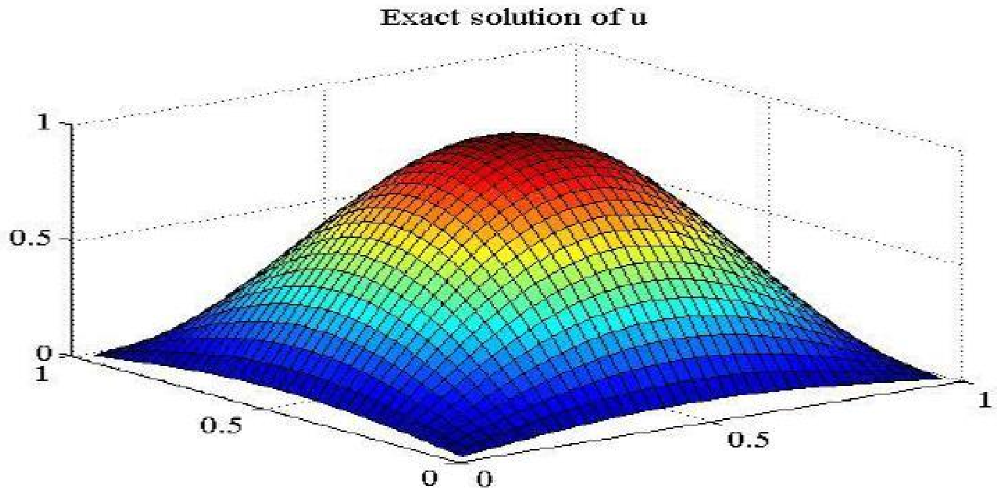
(b)

Figure 10: The Maximum error reduction per each iteration step using Jacobi, Gauss-seidal and SOR methods for $\boldsymbol{h} = \frac{\mathbf{1}}{\mathbf{20}}$.



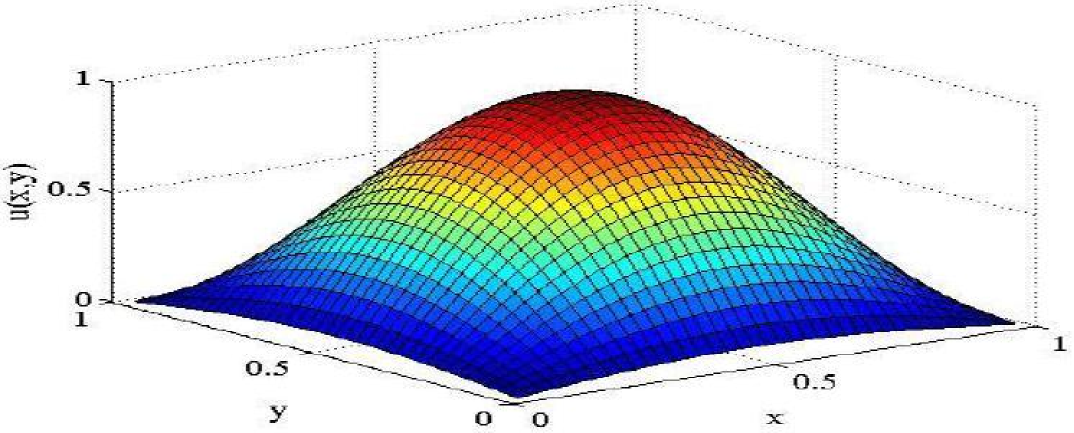
(c)

Figure 11 : The maximum error reduction per each iterations step using Jacobi, Gauss seidal & SOR methods for $\boldsymbol{h} = \frac{\mathbf{1}}{\mathbf{40}}$ for problem 1.



Approximate solution of u using Jacobi Method

Figure 12: Exact and Approximate solution of u using Jacobi Method with $h = \frac{1}{10}$.



Approximate solution of u using Gauss-Seidel Method

Figure 13: Exact and Approximate solution of u using Gauss-seidal method with $h = \frac{1}{10}$

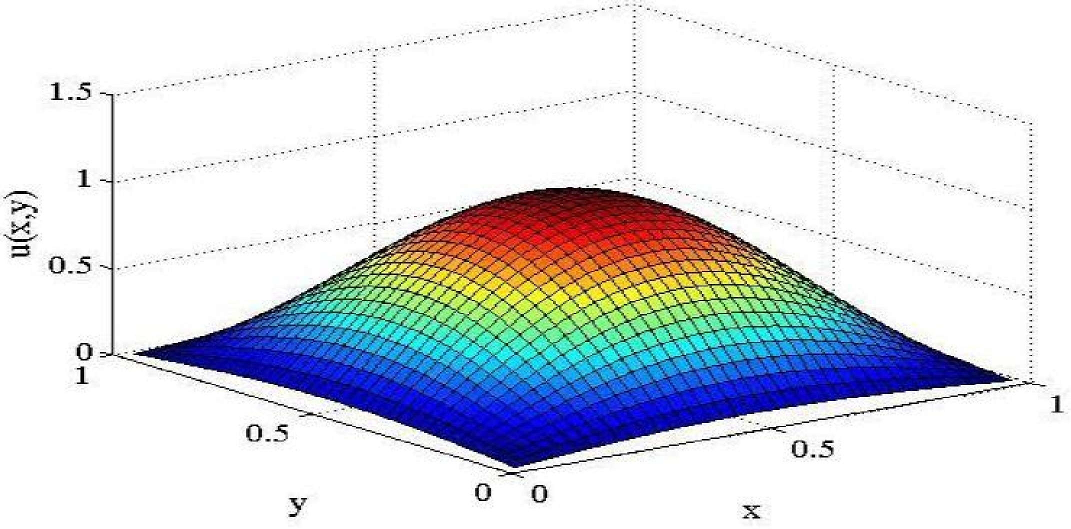


Figure 14: Exact and Approximate solution of u using SOR method with $h = \frac{1}{10}$

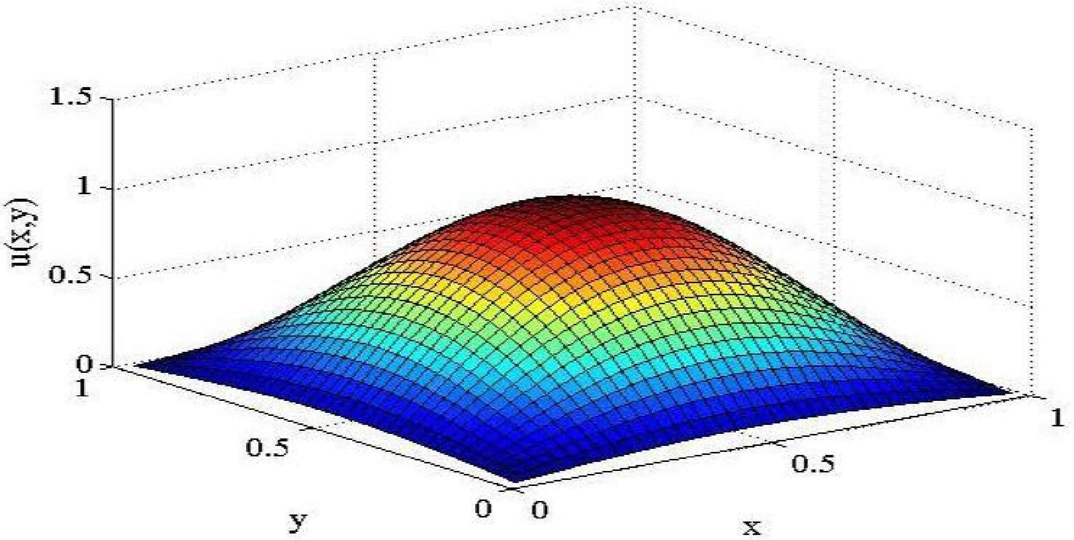


Figure 15 : Exact and approximate solution of U using Jacobi, G. seidal and SOR methods with $h = \frac{1}{40}$ for problem 1.

Problem 2

$U_{xx} + U_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y)\ \ \Omega = \{(x,y)/0 < x,y < 1\}$

With $U(x,0) = U(x,1) = U(0,y) = U(1,y) = 0\ \in \partial\Omega$

Where $U_{exact} = \sin(\pi x)\sin(\pi y)$

Compare the iterations of the iterative methods with different values of step size h.

h	Jacobi	Gauss-seidal	SOR		ADI	U
	Number of iteration	Number of iteration	Number of iteration	ω_{opt}		
$\frac{1}{10}$	171	93	24	1.5279	16	0.6232
$\frac{1}{20}$	576	317	45	1.7295	24	0.5603
$\frac{1}{40}$	1858	1042	85	1.8545	53	0.5219

Table 1: total number of iterations for Jacobi, Gauss –seidal, SOR and ADI methods with different values of $h = \frac{1}{10},\ h = \frac{1}{20}\ , h = \frac{1}{40}$. $Table\ 1$ Illustrates, the total number of iterations for each iterative method with various values of h for the solution of the problem. In comparisons between proposed iterative methods we observed from Table 1, ADI method requires less total number of iteration than SOR and SOR method requires less total number of iteration than Jacobi & G. seidal method and also G. seidal method is twice as faster as Jacobi method.

5. Conclusion

In this work we emphasized the numerical solution of elliptic Poisson's problem using 5 point finite difference discretization. The general linear system is then solved by basic iterative methods namely: Jacobi, G-seidal and SOR. The structures of the matrices generated from the 5 points finite difference discretization are sparse and symmetric positive definite. For that aspect related theorem and definitions are described for the solution of linear system of equations.

Our practical problems were solved for various values of step size h in order to compare the efficiency of the basic iterative methods. The analysis of results shows that the Gauss seidal method converges faster than the Jacobi method because it uses more recent number to make it guess. Due to this the Eigen values of Gauss seidal method will always smaller in magnitude than the Jacobi method. An even faster method is SOR. A numerical result shows that SOR method for a suitable chosen value of optimum relaxation parameter.

We also observed that Jacobi, Gauss seidal and SOR methods are easy to implement .but impractical for problems with large number of grids and also SOR requires the optimum value of relaxation parameter for fast convergence, which needs an extra computation. Thus, the SOR could be considered more efficient than the Jacobi and Gauss-seidal methods for small grids.

And also Poisson solver based on Alternating Direction Implicit (ADI) method is developed. The convergence of ADI method is observed to be much better than convergence of other classical iterative methods. For further research optimal parameter for different geometries needs to be determined and the working of the method should be checked on much more complicated domains

Reference

[1]Peaceman, D.W. and J.R.Rachford, *The numerical solution of parabolic elliptic differential equations,* journal of the society for industrial Applied Mathematics 3(1955), 28-41.

[2] J.Izadian, S.kavoosi, M.Jalili, *A pseudo-spectral method for solving partial differential equations in irregular domain*s, world applied science journal 18(11) (2012)1609-1614.

[3] J.J Benito, U.F.Prieto, L.Gavete, *Influence of several factors in the Generalized Finite difference method*, Applied Mathematical modeling, 25(2001)1039-1053

[4]Yousif Ahmed Qahraman, 2004 *Basic Iterative Method for solving elliptic partial differential equation*: Easter Mediterranean university (July 2014).

[5]W.F, Ames, *Numerical methods for partial differential equations.* Academic press, Newyork (1997).

[6]S.Li, W.K Liu, *mesh free particles methods* Springer (2004)

[7]J.J Benito, U.F.Prieto, L.Gavete, *Influence of several factors in the Generalized Finite difference method*, Applied Mathematical modeling, 25(2001)1039-1053

[8] J.Izadian, S.kavoosi, M.Jalili, *A pseudo-spectral method for solving partial differential equations in irregular domain*s, world applied science journal 18(11) (2012)1609-1614.

[9]J.D.Hoffman, *Numerical methods for engineers and scientists.* Singapore: Mc Graw-Hill.Inc.1992.

[10] J.Ericksen, "*Iterative and direct methods for solving Poisson's equation and their adaptability to ILLIAC IV,*" Univ Illinois, urbana-champaign1972.

[11] P.Jenson, *Finite difference Technique for variable Grids computers and structures* 2(1972)17-29.

[12]N.Perrone, R. kaos*, A Generalized finite difference method for arbitrary meshes, computers and structures*. 5(1975)45-58.

[13]R.courant, D.Hilbert, *methods of mathematical physics*, volume 2, interscience publishers, Newyork, 1962.

[14] Von Neumann J Goldstine HH (1947) ,*Numerical inverting of matrices of high order*. Bull Amer mathematics soc 53:1021-1099.

[15] Lax, P.D.Richtmyer RD, *partial differential equations, lectures on hyperbolic equations,* Stanford University press (1963).