ISSN 2581-3463

AJMS
Asian Journal of Mathematical Sciences

RESEARCH ARTICLE

On the Numerical Fixed Point Iterative Methods of Solution for the Boundary Value Problems of Elliptic Partial Differential Equation Types

Eziokwu C. Emmanuel, Anokwute Chinelo

Department of Mathematics, Michael Okpara University of Agriculture, Umudike, Abia, Nigeria

Received: 01-08-2018; Revised: 10-09-2018; Accepted: 09-11-2018

ABSTRACT

In this research work, we have studied the finite difference method and used it to solve elliptic partial differential equation (PDE). The effect of the mesh size on typical elliptic PDE has been investigated. The effect of tolerance on the numerical methods used, speed of convergence, and number of iterations was also examined. Three different elliptic PDE's; the Laplace's equation, Poisons equation with the linear inhomogeneous term, and Poisons equations with non-linear inhomogeneous term were used in the study. Computer program was written and implemented in MATLAB to carry out lengthy calculations. It was found that the application of the finite difference methods to an elliptic PDE transforms the PDE to a system of algebraic equations whose coefficient matrix has a block tri-diagonal form. The analysis carried out shows that the accuracy of solutions increases as the mesh is decreased and that the solutions are affected by round off errors. The accuracy of solutions increases as the number of the iterations increases, also the more efficient iterative method to use is the SOR method due to its high degree of accuracy and speed of convergence.

Key words: Partial differential equations, finite difference method, iterative methods, convergence

INTRODUCTION OF BASIC CONCEPTS

The form of the occurrence of many physical problems involving functions of two independent variables motivates one to consider the form, accuracy and the ease in obtaining such solutions, not just the mere fact of the existence of the solution. Many problems of physical science and engineering which involves functions of two independent variables are formulated as partial differential equation (PDE).

A PDE is an equation containing partial derivatives. The dependent variable of any partial differential equation must be a function of at least two independent variables otherwise partial derivative would not arise, for example, the equations

$$\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}=6u \tag{1}$$

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial x \partial y}=0 \tag{2}$$

are typical examples of PDE of the first- and second-order, respectively? X and Y being the independent variable and $U = U(x,y)$ being the dependent variable. To obtain useful information about the physical situation modeled by PDE, it is necessary to solve the equation. It is obvious that many PDEs arising in engineering applications cannot be solved in closed form by known analytical methods (G. Stephenson, 1968). If a solution is needed in such cases, numerical methods must be used. Furthermore, many times it may be more convenient to employ a numerical method to solve P.D.F even though a close form of the solution is available. This is so when the difficulty of computing values from the closed form solution exceeds that using a conventional numerical method for solving equations.

At present, the availability of digital computers makes the application of the numerical method to PDE easier to handle. Therefore, it is of best interest to learn how to obtain accurate and meaningful numerical solutions to PDE s. A wide variety of problem of physics and engineering are modeled using second-order linear PDE. A linear

Address for correspondence:
Eziokwu C. Emmanuel
E-mail: Okereemm@yahoo.com

PDE is said to be linear if it is of the first degree in the dependent variables. For example;

Laplace's equation

$$\nabla^2 u = 0 \tag{3}$$

This Strauss (1992) describes the steady-state temperature distribution of a solid that varies from point to point without time regulation as well as the gravitational potential in a change free region and applies to the flow of an in compressive fluid with no sources.

The heat equation

$$k^2 \nabla^2 u = \frac{\partial u}{\partial t} \tag{4}$$

The function $U(x,t)$ of Equation (4) represents the temperature. U in a region containing no heat source and also applies to the diffusion of chemicals that have a concentration. The constant K is called diffusion. The equation [Oruh (2012)] is clearly second order in the three spatial variables but first in time.

The wave equation

$$C^2 \nabla^2 U = \frac{\partial^2 u}{\partial t^2} \tag{5}$$

Problems in mechanical vibrations often lead to the wave equation. The solution $U(x,t)$ of the equation represents the displacement from equilibrium, $U(x,t)$ of a vibrating string, gas, or liquid. The equation [Stevenson (1968)] also occurs in electromagnetism where U can be a component of the electric or magnetic field in an electromagnetic wave or current-voltage along a transmission line. The quantity C is the speed of propagation of the waves.

Partial differential equation

An equation [Moon and Spencer (1953)] involving partial derivatives of one or more functions of two or more independent variables is called PDE. The order of the highest derivatives is called the order of the PDE's, for example, the wave equation which is given as;

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{K^2} \frac{\partial^2 u}{\partial z^2} \tag{6}$$

In general, a PDE of second order in two independent variables X and Y is of the form $F(x, y, U, U_x, U_y, U_{xx}, U_{y,y})$, where U is the dependent variable in X and Y, U_x, and U_y are the first partial derivatives of $U_{(x,y)}$, respectively, $U_{x,x}$, U_{xy}, and $U_{y,y}$ are the second partial derivatives.

Linear and non-linear PDE's

A PDE [Smith (1995)] is said to be linear if it is of the first degree in the dependent variable (the unknown function). However, it is said to be non-linear if it is not linear. Some essential linear PDE s are;
The wave equation,

$$U_{tt} = C^2 U^{xx} \tag{7}$$

The Laplace equation

$$U_{xx} + U_{yy} = 0 \tag{8}$$

The Poisson's

$$U_{xx} + U_{yy} = f(x,y) \tag{9}$$

Homogenous and Inhomogeneous PDE's

A linear PDE [Smith (1995)] is said to be homogenous if each term of it contains either the dependent variable or one of its derivatives. Example of a homogenous linear PDE is the Laplace's equation that is Equation 7. A PDE is said to be inhomogeneous if there exists a term in it which does not contain the dependent variable or one of its derivatives. An example is the poison's Equation 8.

Solution of a PDE

A solution of a PDE in some region R of the space of the independent variables is a function that has all the partial derivatives appearing in the equation in some domain containing R and satisfies the equation everywhere in R. In general, the totality of solutions of a PDE is very large.

For example, [Peter and Turner (1989)] the functions

$U_{(x,y)} = x^2 - y^2$, $U_{(x,y)} = e^x \sin y$ *and* $U_{(x,y)} = \ln (x^2 + y^2)$

which are entirely different are solutions of the Laplace's Equation 1. The unique solution of the PDE is obtained by additional conditions arising from the problem. For instance, the condition that the solution $U(x,y)$ assume given values of the boundary of a region considered (boundary condition) or when time t is one of the variables that U or U_t or both prescribed at $t = 0$ (initial condition).

Elliptic PDE

The general form of the two-dimensional second-order linear PDE s is given by; [Kalambi (2008)]

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0$$

Where A, B, C, D, E, F, and G are functions of the independent variables X and Y, U is the dependent variable, U_x and U_y are the first partial derivatives of $U_{(x,y)}$, U_{xx} U_{xy} and U_{yy} are the second partial derivatives of $U_{(x,y)}$.

Block tri-diagonal matrix

A matrix is called a tri-diagonal matrix if it has all its non-zero entries on the main diagonal and zero on the sloping parallels immediately above or below the main diagonal [Sheid (1988)]

$$T = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

When a diagonal matrix can be represented in the form in which the diagonal entries and the sloping parallels are matrices instead of single numbers, the matrices are then called block tri-diagonal matrix. The matrix T above is an example of block tri-diagonal matrix because it can be written as [Erwin (1997)]

$$T = \begin{bmatrix} B & C & & \\ A & \ddots & \ddots & \\ & \ddots & \ddots & C \\ & & A & B \end{bmatrix}$$

Where $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$ and $C = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

are all matrices, not single numbers.

METHODS OF SOLUTION

Iterative methods of solving linear system of equation

In this section, we shall study the algorithm and flowcharts of the various iterative methods for solving system of linear equations, namely Jacobi iteration, extrapolated Liebmann (SOR), and unextrapolated Liebmann (Gauss-Seidel) methods. There are other iterative methods which are not discussed here, such as conjugate gradient method (C.G) Generalized Minimal Residual (GMRES) and Chebyshev Iteration.

Jacobi iteration

The Jacobi method of solving a linear system of equations of the form $AX = b$ where A is the coefficient matrix, X the column vectors of the unknowns and b the R.H.S constants. To solve a system of N linear equations, rearrange the rows so that the diagonal elements have magnitude as large as possible relative to the magnitudes of other coefficients in the same row. Define the rearranged system as $AX = b$ then, beginning with an initial approximation to the solution vector $X^{(1)}$, compute each component of $X^{(n+1)}$ for $i = 1,2,\ldots,n$ by Weisstein *et al.* (1993)

$$X^{(n+1)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{a_{i,j}}{a_{ii}} x_j^{(n)}, n = 1, 2 \cdots \quad N$$

A sufficient condition for convergence [Chapra and Carnale (1998) is that

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}} |a_{ij}|, i = 1, 2, \cdots n$$

This process can be repeated until the condition $\max[X^{(n+1)} - X^{(n)}] \leq \varepsilon$ is satisfied, where ε is the

possible tolerance. Jacobi method will converge if the matrix is diagonally dominant. The Jacobi method is also known as the method of simultaneous displacements because each of the equations is simultaneously changed using the most recent set of X values.

Gauss-Seidel method

To solve $AX = b$, given an initial approximation X^0 where A, X, b retain their meaning in algorithm 3.1, respectively, then beginning with an initial approximation to the solution vector $X^{(1)}$, we compute each component of $X^{(n+1)}$ for $n = 1,2,\ldots, N$ by

$$X_i^{(n+1)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{i,j}}{a_{ii}} x_j^{(n+1)} - \sum_{j=i+1}^{N} \frac{a_{i,j}}{a_{ii}} x_j^{(n)}, n = 1,2\cdots N$$

A sufficient condition for convergence [Timber (1989)] is that

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq 1}} |a_{ij}|, i = 1,2,\cdots \ N$$

This process can be repeated until the condition max $[X^{(n+1)} - X^{(n)}] \leq \varepsilon$ is satisfied, where ε is some possible tolerance. Convergence is only guaranteed in case of Gauss-Seidel method if matrix A is diagonally dominant.

Matrix is said to be diagonally dominant if the absolute value of the diagonal element in each row is greater than or equal to the summation of absolute values of rest of elements of that particular row. The iterative process is terminated when a convergence criterion is fulfilled; the Gauss-Seidel method is also known as the method of successive displacement that uses the latest iteration values.

Successive over-relaxation method

In numerical linear algebra, the method of the (SOR) modifies the correction term of the Gauss-Seidel method as follows for $i = 1,2,\ldots, n$

$$x_i^{k+1} = x_i^k + \frac{w}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i}^{n} a_{i,j} x_j^k \right)$$

The sufficient condition for convergence [Strauss (1992)] is also;

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq 1}} |a_{i,j}|, i = 1,2,\cdots N$$

This process can be repeated until the condition max $[X^{(n+1)} - X^{(n)}] \leq \varepsilon$ is satisfied, where ε is some possible tolerance. When this is true, $X^{(n)}$ will converge to the desired solution no matter the initial vector used. This method is a generalization of the Gauss-Seidel method for application in problems of structural engineering.

When $W = 1$ we have the Gauss-Seidel method, $w > 1$ it is called successive over relaxation, while for $w < 1$ it is called successive under-relaxation. The parameter W is called the relaxation parameter and its purpose is to modify the spectral residues for faster convergence.

Finite difference method

In this subsection, we shall discuss in detail the finite difference method and show how to obtain finite difference approximation of a typical elliptic PDE. This is to lay a foundation for its application to elliptic PDE later in this work. A pictorial representation of the mesh points involved in Equation 26.

Finite difference

Let $f_i = f_i (i=1\ldots n)$ be a sequence of values of a function F. The difference of the f_i values is called finite differences and is denoted as follows; the first finite difference,

$$\Delta f_i = fi + 1 - \text{fi}$$

and second finite difference

$$\Delta 2^{f\_i} = \Delta n - 1fi + 1 - \Delta^{nfi} \qquad (10)$$

And Equation 10 is called the nth finite difference of F. The symbol Δ is the forward difference operator and it is defined by $\Delta f_i = f_{i+1} - f_i$ and also the difference listed above may be referred to as forward difference. Other finite difference is backward finite difference operator ∇ defined by

$$\nabla f_i = f_i - f_{i+1}$$

and the central difference operator δ defined by

$$\Delta f_i = fi_+ - f_{i-}$$

The three finite difference operators Δ, ∇, and δ, that is, forward, backward, and central, respectively, are related through the equation $\Delta f_i = \nabla^n f_{i+n} = \delta^n f_{i+\frac{n}{2}}$ (P.Turner, 1989).

Taylor's theorem

In this section, we shall consider [Weisstein (1993)] the Taylor's series expansion of a function $U(x,y_j)$ and we shall establish the formula statement of the theorem. Suppose $f(x) \in C_n[a,b]$ set of continuous functions in the closed intervals $[a,b]$ and $f_{n+1}(x)$ exists in $[a,b]$. Let $x_0 \in [a,b]$ for every $x \in [a,b] \in$ between X_0 and X_0 with $f(x) = P_n(x) + R_n(x)$, where

$$P_n(x) = f(x) + f'(x_0)(x-x_0) + \frac{f'(x_0)(x-x_0)^2}{2!} + \ldots + \frac{f^n(x_0)(x-x_0)^n}{n!} = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)(x-x_0)^k}{k!}$$

and

$$R_n(x) = f^{n+1} \sum \frac{(x_0)(x-x_0)^{n+1}}{(n+1)!}$$

P_n is called the n th degree Taylor polynomial for about x_0 and $R_n(x)$ is called the truncation error associated with $P_n(x)$. The infinite series obtained by taking the limit $P_n(x)$ as $n \to \varepsilon fty$ is called Taylor series for $F(x)$ about X_0 (Burden, Farres, Reynolds, 1988). In analogy with the function $f(x)$ of the variable X, the Taylor series expansion with respect to X for a function $U(x,y)$ of two independent variable is

$$U(x_{i+h}, y_j) = U(x_i, y_j) + hU_x(x_i, y_j) + \frac{h^2}{2!}U_{xx}(x_i, y_j) + \frac{h^3}{3!}U_{xxx}(x_i, y_j) + \cdots \quad (11)$$

Where h the size of the mesh point and (x_i,x_j) is the coordinate of mesh point. If we replace h by $-h$ in the above Equation 11 we obtain

$$U(x_{i-h}, y_j) = U(x_i, y_j) - hU_x(x_i, y_j) + \frac{h^2}{2!}U_{xx}(x_i, y_j) - \frac{h^3}{3!}U_{xxx}(x_i, y_j) + \ldots \quad (12)$$

Similarly, the Taylor series expansion with respect to y of $U(x,y)$ in the neighborhood of x_i,x_j is

$$U(x_i, y_{j+m}) = U(x_i, y_j) + mU_y(x_i, y_j) + \frac{m^2}{2!}U_{yy}(x_i, y_j) + \frac{m^3}{3!}U_{yyy}(x_i, y_j) + \ldots \quad (13)$$

Replacing m in Equation 13, we have,

$$U(x_i, y_{j-m}) = U(x_i, y_j) - mU_y(x_i, y_j) + \frac{m^2}{2!}U_{yy}(x_i, y_j) - \frac{m^3}{3!}U_{yyy}(x_i, y_j) + \ldots \quad (14)$$

We assume that the partial derivatives exist and each series is convergent, in each of the Equation from 11 to 14

Finite difference approximation of partial derivatives

First-order derivatives

Forward difference approximation

By considering equation (11) above, for small h, we may neglect terms of order h^2 in the equation, then we obtain

$$U(x_{i+h}, y_j) = U(x_i, y_j) + hU_x(x_i, y_j)$$

Then, we get an approximation for $U_x(x_i,y_j)$

$$U(x_{i+h}, y_j) - U(x_i, y_j) = hU_x(x_i, y_j)$$

Then,

$$U_x(x_i, y_j) \approx \frac{U(x_{i+h}, y_j) - U(x_i, y_j)}{h} \quad (15)$$

Equation 15 gives the finite forward difference approximation of the U_x at the point (x_i,y_j). Similarly, the forward finite difference approximation of U_y at the point (x_i,y_j) is obtained by considering Equation 13 and neglecting terms of order h^2 to have

$$U(x_i, y_{j+m}) = U(x_i, y_j) + mU_y(x_i, y_j)$$

We also get the approximation for $U_y(x_i, y_j)$, as

$$U_y(x_i, y_j) \approx \frac{U(x_i, y_{j+m}) - U(x_i, y_j)}{m} \quad (16)$$

Backward difference approximation

If we also consider Equation 12 up to first order in h and neglecting terms of order h^2, we obtain

$$U\left(x_{i-h},y_j\right)=U\left(x_i,y_j\right)-hU_x\left(x_i,y_j\right)$$
$$=U\left(x_{i-h},y_j\right)-U\left(x_i,y_j\right)=-hU_x\left(x_i,y_j\right)$$

and so we get,

$$U_x\left(x_i,y_j\right)=\frac{U\left(x_i,y_j\right)-U\left(x_{i-h},y_j\right)}{h} \quad (17)$$

The equation is the backward finite difference approximation to U_x at the point (x_i,y_j). In the same way, the backward difference approximation to at the U_y point (x_i,y_j) is obtained by considering Equation 14 and neglecting terms of order h^2, we obtain

$$U\left(x_i,y_{j-m}\right)=U\left(x_i,y_j\right)-hU_y\left(x_i,y_j\right)$$
$$=U\left(x_i,y_{j-m}\right)-U\left(x_i,y_j\right)$$
$$=-hU_y\left(x_i,y_j\right)$$

Finally, we obtain

$$U_y\left(x_i,y_j\right)=\frac{U\left(x_i,y_j\right)-U\left(x_i,y_{j-m}\right)}{m} \quad (18)$$

The error in the forward and backward difference approximation for U_x and U_y are of the order h and m, respectively.

Central difference approximation

For the central difference approximation, if we [Turner (1989)] neglect the terms of order h^3 in Equation 11 and 12, then if we subtract equation from12 to 11 we obtain

$$U\left(x_{i+h},y_j\right)-U\left(x_{i-h},y_j\right)$$
$$=hU_x\left(x_i,y_j\right)+hU_x\left(x_i,y_j\right)$$
$$\Rightarrow U\left(x_{i+h},y_j\right)-U\left(x_{i-h},y_j\right)=2hU_x\left(x_i,y_j\right) \quad (19)$$
$$\Rightarrow U_x\left(x_i,y_j\right)=\frac{U\left(x_{i+h},y_j\right)-U\left(x_{i-h},y_j\right)}{2h}$$

Equation 19 gives the central difference approximation to the U_x at x_i, y_j. Similarly, using Equation 13 and 14, the central finite approximation to U_y at the point (x_i,y_j) is obtained as

$$U_y\left(x_i,y_j\right)=\frac{U\left(x_i,y_{j+m}\right)-U\left(x_i,y_{j-m}\right)}{2m} \quad (20)$$

In obtaining Equation 20, forms of orderm^3 are neglected. The errors in the central difference approximation to U_x and U_y are orders h^2 and m,^{2} respectively.

Second-order derivatives

The finite difference approximation to U_{xx} [Smith (1995)] is obtained by considering Equation 12 and 13, if we neglect the form of order h^3 in each of the equation and sums them up; we obtain the finite difference approximation to U_{xx} to be

$$U_{xx}\left(x_i,y_j\right)=\frac{U\left(x_{i-h},y_j\right)-2U\left(x_i,y_j\right)+U\left(x_{i+h},y_j\right)}{h^2} \quad (21)$$

The error term in this approximation for U_{xx} is of order h^2. Furthermore, the finite difference to U_{yy} is similarly obtained by adding Equation 13 and 14 after neglecting terms of order m^3, we then obtain the finite difference approximation of U_{yy} at $U(x_i,y_j)$ as

$$U_{yy}\left(x_i,y_j\right)=\frac{U\left(x_{i-h},y_{j-m}\right)-2U\left(x_i,y_j\right)+U\left(x_i,y_{j+m}\right)}{m^2} \quad (22)$$

and its error is of order m^2.

Finite difference approximation of typical elliptic PDE

To illustrate how to apply the finite difference method to a typical elliptic PDE, we have considered a boundary value problem consisting of the elliptic equation; $U_{xx} + U_{yy} = f(x_i,y_j)$ defined in a domain $D = (x_i,y_j)$ for $a \le x \le bc \le y \le d$ subject to boundary condition at the boundary of $D\{U(x_i,y_j)\} = g(x_i,y_j)$ is continuous in the domain D, then the boundary value problem has a unique solution in D. To obtain the finite difference approximation to the above BVP, we subdivide the domain D into a rectangular mesh point (x_i,y_j) where $x_i = a + ih$, $(i = 0,1\ldots n)$ and $y_j = c + jm$, $(j = 0,1\ldots q)$ with $h=\frac{b-a}{n}$ and $m=\frac{d-c}{q}$. Then, at each interior

point of the rectangular mesh point, we replace the partial derivatives U_{xx} and U_{yy} present in the given elliptic equation by their appropriate finite difference approximation, that is, Equation 21 and 22, respectively, and we obtain [Strauss (1992)]

$$\frac{U\left(x_{i-h},y_j\right)-2U\left(x_i,y_j\right)+U\left(x_{i+h},y_j\right)}{h^2}+\frac{U\left(x_i,y_{j-m}\right)-2U\left(x_i,y_j\right)+U\left(x_i,y_{j+m}\right)}{m^2}=f\left(x_i,y_j\right)$$

For ($i = 1,2,\ldots n, j = 1,2,\ldots q$), if we write $U(x_i,y_j) = Z_{i,j}$, $X_{i+h} = X_{i+1}$, $X_{i-h} = X_{i-i}$ and $Y_j + h = Y_j + 1$, $Y_j - h = Y_j - 1$. Then, we have

$$\frac{Z_{i-1,j}-2Z_{i,j}+Z_{i+1,j}}{h^2}+\frac{Z_{i,j-1}-2Z_{i,j}+Z_{i,j+1}}{m^2}=f\left(x_i,y_j\right) \quad (23)$$

If we multiple both sides of the Equation 23 by h^2 we obtain

$$Z_{i-1,j}-2Z_{i,j}+Z_{i+1,j}+\frac{h^2}{m^2}\left(Z_{i,j-1}-2Z_{i,j}+Z_{i,j+1}\right)=h^2f\left(x_i,y_j\right) \quad (24)$$

Suppose that $\frac{h^2}{m^2}=\alpha^2$, then after collecting like terms we obtain

$$Z_{i-1,j}+Z_{i+1,j}-2\left(1+\alpha^2\right)Z_{i,j}+\alpha^2Z_{i,j-1}+\alpha^2Z_{i,j+1}=h^2f\left(x_i,y_j\right) \quad (25)$$

Equation 26 is the finite difference approximation of the Poisson equation x $U_{xx} + U_{yy} = f(x,y)$ in the domain D. A pictorial representation of the mesh points involved in Equation 26 and the coefficient of the unknown solution in the corresponding terms are shown in the Figures 1-7.
The discrete boundary condition is

$$Z_{0,j}=g\left(a,y_j\right)$$
$$Z_{n,j}=g\left(b,y_j\right)$$
$$Z_{i,0}=g\left(x_i,c\right)$$
$$Z_{i,q}=g\left(x_i,d\right)$$
$$i=1,2\ldots n \text{ and } j=1,2\ldots q$$

Equation 26 together with the boundary conditions constitutes the finite difference subject to the above boundary value problem. For $i = 1,2\ldots n$ and $j = 1$, then Equation 26 gives rise to the following system of equation.

$$Z_{0,1}+Z_{2,1}-2\left(1+\alpha^2\right)Z_{1,1}+\alpha^2Z_{1,0}+\alpha^2Z_{1,2}=h^2f_{1,1}$$
$$Z_{1,1}+Z_{3,1}-2\left(1+\alpha^2\right)Z_{2,1}+\alpha^2Z_{2,0}+\alpha^2Z_{2,2}=h^2f_{2,1}$$
$$Z_{2,1}+Z_{4,1}-2\left(1+\alpha^2\right)Z_{3,1}+\alpha^2Z_{3,0}+\alpha^2Z_{3,2}=h^2f_{3,1}$$
$$\vdots$$
$$Z_{n-1,1}+Z_{n+1,1}-2\left(1+\alpha^2\right)Z_{n,1}+\alpha^2Z_{n,0}+\alpha^2Z_{n,2}=h^2f_{n,1}$$

APPLICATIONS

In this section, we shall apply the finite difference method to three boundary value problems involving elliptic PDE. The algebraic systems of equation resulting from the application of the finite difference method will be solved using the considered three iterative methods; they are Jacobi method, Gauss-Seidel method, and SOR method so as to assess their speed of convergence, since our interest is to solve elliptic PDE and also to study the methods of solving systems of linear equation. To facilitate the calculation, we have written a computer program using MATLAB program

Problem 1: (Poisson equation with linear inhomogeneous term)

Use the finite difference approximation to obtain a solution of the boundary value problem

$$U_{xx}+U_{yy}=4,\left(0<x<1,0<y<2\right)$$
$$U\left(x,0\right)=x^2,U\left(x,2\right)=\left(x-2\right),^2\,0\le x\le1$$
$$U\left(0,y\right)=y^2,U\left(1,y\right)=\left(y-1\right),^2\,0\le y\le2$$

i) Solve the algebraic system of equation with the three iterative methods mentioned in section (2.5) and compare their result with the exact solution $U(x,y) = (x - y)^2$ in each case. Use $h = 0.25$ and $k = 0.5$.
ii) Use $h = 0.2$ and $k = 0.2$ and solve the resulting algebraic system of equation with the best iterative method and compare the result with the same exact solution $U(x,y) = (x - y)^2$.

Solution

Case 1: From the problem $\alpha = \frac{h}{k} = \frac{0.25}{0.5} = 0.5$, a $= c = 0$, $b = 1$, $d = 2$.

The mesh points are (x_i, y_j) where $x_i = 0.25i$ and $y_j = 0.5j, i, j = 1,2,3$.

From Equation 26 we have

$$Z_{i-1,j} + Z_{i-1,j} - 2.5Z_{i,j} + 0.25Z_{i,j} + 0.25Z_{i,j+1} = 0.25 \quad (27)$$

Since $\alpha = 0.5$, Equation 27 holds for each interior mesh point. The discrete boundary conditions associated with Equation 27 are

$$Z_{0,j} = (0.5j)^2, j = 0,1,\ldots,4$$
$$Z_{4,j} = (0.5j - 1)^2, j = 0,1,\ldots,4$$
$$Z_{i,0} = (0.25i)^2, i = 0,1,\ldots,4$$
$$Z_{i,4} = (0.25i - 2)^2, i = 0,$$

Hence, we have

$$Z_{0,1} = 0.25,\ Z_{0,2} = 1.0, Z_{0,3} = 2.25, Z_{0,4} = 4.0$$
$$Z_{1,0} = 0.0625, Z_{2,0} = 0.25, Z_{3,0} = 0.5625, Z_{4,0} = 1.0$$

Similarly,

$$Z_{4,1} = 3.0625, Z_{2,4} = 2.25, Z_{3,4} = 1.5625, Z_{4,4} = 1.0$$
$$Z_{4,1} = 0.25, Z_{4,2} = 0, Z_{4,3} = 0.25, Z_{4,4} = 1.0$$

Using Equation 2.1 and applying the boundary conditions, we will have the following systems of equation

The above system can be represented in a matrix form as $AZ = B$ where

$$A = \begin{bmatrix} -2.5 & 1 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2.5 & 1 & 0 & 0.25 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2.5 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & -2.5 & 1 & 0 & 0.25 & 0 & 0 \\ 0 & 0.25 & 0 & 1 & -2.5 & 1 & 0 & 0.25 & 0 \\ 0 & 0 & 0.2 & 0 & 1 & -2.5 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & -2.5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 & 0 & -2.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0.25 & 0 & 1 & -2.5 \end{bmatrix},$$

$$Z = \begin{bmatrix} Z_{11} \\ Z_{2,1} \\ \\ \vdots \\ \\ \\ Z_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} -0.015625, 0.1875, -0.140625, \\ -0.75, 0.25, 0.25, -2.765625, \\ -0.3125, -0.3900625 \end{bmatrix}$$

The numerical solutions are obtained using the program in Appendix 3, 6, and 9

The result for Jacobi, Gauss-Seidel, and SOR iteration is presented in Tables 1-3 below and the exact solution of problem 1 is a function of $U(x,y) = (x - y)^2$ and the values of X are as follows: (0.0625, 0.007, 0.0625, 0.05625, 0.25, 0.0625, 1.5625, 1.00, 0.5625)

Case 2

Solution:

Here, $\alpha = \frac{h}{k} = \frac{0.2}{0.2} = 1, a = c = 0, b = 1, d = 2$, (X_i, Y_j) are the mesh points with

$$X_i = 0.2_i, I = 1,2,3,4$$
$$Y_i = 0.2_j, j = 1,2,3,\ldots,9$$

From Equation 30 we obtain

$$Z_{i-1,j} + Z_{i-1,j} - 4W_{i,j} + Z_{i,j-1} + Z_{i,j+1} = \frac{4}{25} \quad (28)$$

For $i = 1,2,\ldots,4$ and $j = 1,\ldots,9$

The discrete boundary conditions are

$$Z_{i,0} = (0.2_i)^2, i = 1,2,3,4, Z_{i,10} = (0.2i - 2)^2$$
$$Z_{0,j} = (0.2_i)^2, Z_{5,j} = (0.2j - 1)^2, j = 0,1,\ldots,7$$

Using the boundary conditions in Equation 28 and writing out the equation for each interior point, a system of 36 algebraic equations with 36 unknown is obtained.[1-4] Using the program in Appendix 3, the numerical solution to the second case of problem one is obtained. In Tables 5 and 6, we have restricted the use of SOR method, due to its less computational time to converge as compared to the other two iterative methods and it takes care of the data input problem in solving linear systems. The result is presented below;

PROBLEM: LAPLACE EQUATIONS

Use the finite difference approximation to obtain a numerical solution of the boundary value

Table 1: Jacobi Iteration

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1	0.0063	−0.0750	0.0563	0.3000	−0.1000	−0.1000	1.1063	0.1250	0.1563
2	0.0063	−0.0600	0.0163	0.3712	−0.0150	−0.1188	1.1863	0.6200	0.1963
3	0.0194	−0.0675	0.0204	0.4133	0.0570	−0.0848	1.3914	0.6765	0.3924
4	0.0206	−0.0534	0.0208	0.4639	0.0923	-0.0359	1.4182	0.8442	0.4184
5	0.0313	−0.0492	0.0313	0.4808	0.1503	−0.0192	1.4903	0.8689	0.4903
6	0.0346	−0.0349	0.0346	0.5123	0.1666	0.0123	1.5019	0.9323	0.5019
7	0.0435	−0.0306	0.0435	0.5203	0.1995	0.0203	1.5304	0.9432	0.5304
8	0.0460	−0.0202	0.0460	0.5372	0.2075	0.0372	1.5355	0.9693	0.5355
9	0.0519	−0.0174	0.0519	0.5412	0.2247	0.0412	1.5477	0.9742	0.5477
10	0.0534	−0.0110	0.0534	0.5498	0.2286	0.0498	1.5500	0.9856	0.5500
11	0.0568	−0.0094	0.0568	0.5518	0.2373	0.0518	1.5555	0.9879	0.5555
12	0.0577	−0.0058	0.0577	0.5562	0.2393	0.0562	1.5566	0.9931	0.5566
13	0.0595	−0.0049	0.0595	0.5571	0.2437	0.0571	1.5591	0.9942	0.5591
14	0.0600	−0.0030	0.0600	0.5593	0.2446	0.0593	1.5596	0.9967	0.5596
15	0.0610	−0.0025	0.0610	0.5598	0.2468	0.0598	1.5608	0.9972	0.5608
16	0.0612	−0.0015	0.0612	0.5609	0.2473	0.0609	1.5611	0.9984	0.5611
17	0.0617	−0.0013	0.0617	0.5612	0.2484	0.0612	1.5617	0.9986	0.5617
18	0.0618	−0.0008	0.0618	0.5617	0.2487	0.0617	1.5618	0.9992	0.5618
19	0.0621	−0.0007	0.0621	0.5618	0.2492	0.0618	1.5621	0.9993	0.5621
20	0.0622	−0.0004	0.0622	0.5621	0.2493	0.0621	1.5622	0.9996	0.5622
21	0.0623	−0.0003	0.0623	0.5622	0.2496	0.0622	1.5623	0.9997	0.5623
22	0.0623	−0.0002	0.0623	0.5623	0.2497	0.0623	1.5623	0.9998	0.5623
23	0.0624	−0.0002	0.0624	0.5623	0.2498	0.0623	1.5624	0.9998	0.5624
24	0.0624	−0.0001	0.0624	0.5624	0.2498	0.0624	1.5624	0.9999	0.5624
25	0.0625	−0.0001	0.0625	0.5624	0.2499	0.0624	1.5625	0.9999	0.5625
26	0.0625	−0.0000	0.0625	0.5625	0.2499	0.0625	1.5625	1.0000	0.5625
27	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625
28	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625
29	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625
30	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625

The method converged after 28th iteration

Table 2: Gauss-Seidel iteration

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1	0.00625	−0.07250	0.02725	0.02725	0.30063	0.01300	−0.09207	1.13631	0.37937
2	0.00731	−0.05987	0.02309	0.41956	0.08309	0.02652	1.38054	0.83727	0.48851
3	0.02426	0.04775	0.03450	0.47372	0.15783	1.48853	1.38054	0.93160	0.53043
4	0.03452	−0.03161	0.04515	0.51544	0.20235	0.03850	1.38054	0.96958	0.54793
5	0.04515	−0.01865	0.05264	0.53850	0.22589	0.05041	1.54793	0.98593	0.55567
6	0.05264	−0.01030	0.05717	0.55041	0.23789	0.05644	1.55567	0.99332	0.5922
7	0.05717	−0.00547	0.05971	0.55644	0.24394	0.05947	1.55922	0.99677	0.56091
8	0.05971	−0.00547	0.05971	0.55644	0.24394	0.05947	1.55922	0.99677	0.56091
9	0.05971	−0.00284	0.06106	0.55947	0.24697	0.06098	1.56091	0.99842	0.56172
10	0.06177	−0.00074	0.06213	0.56174	0.24924	0.06212	1.56211	0.99961	0.56231
11	0.06213	−0.00037	0.06231	0.56212	0.24962	0.06231	1.56231	0.99981	0.56240
12	0.06231	−0.00019	0.06241	0.56231	0.24981	0.06241	1.56240	0.99990	0.56245
13	0.06241	−0.00009	0.06245	0.56241	0.24991	0.06245	1.56245	0.99995	0.56248
14	0.06245	−0.00005	0.06248	0.56245	0.24995	0.06248	1.56248	0.99998	0.56249
15	0.06248	−0.00002	0.06249	0.56248	0.24998	0.06249	1.56249	0.99999	0.56249
16	0.06249	−0.00001	0.06249	0.56249	0.24999	0.06249	1.56249	0.99999	0.56250
17	0.06249	−0.00001	0.06250	0.56249	0.24999	0.06250	1.56250	1.00000	0.56250
18	0.06250	−0.00000	0.06250	0.56250	0.25000	0.06250	1.56250	1.00000	0.56250
19	0.06250	−0.00000	0.06250	0.56250	0.25000	0.06250	1.56250	1.00000	0.56250
20	0.06250	−0.00000	0.06250	0.56250	0.25000	0.06250	1.56250	1.00000	0.5625

We see that the method converges after 18 iteration

Table 3: SOR iteration

1	0.0070	−0.0809	0.0268	0.3368	0.0298	−0.0956	1.2767	0.7153	0.4847
2	0.0077	−0.0555	0.0242	0.4528	0.1183	0.0095	1.4570	0.9373	0.5378
3	0.0319	−0.0389	0.0437	0.5014	0.2033	0.0431	1.5402	0.9813	0.5549
4	0.0419	−0.0182	0.0544	0.5441	0.2345	0.0561	1.5547	0.9936	0.5598
5	0.0548	−0.0066	0.0598	0.5560	0.2447	0.0603	1.5599	0.9978	0.5616
6	0.0597	−0.0023	0.0616	0.5603	0.2481	0.0617	1.5616	0.9992	0.5622
7	0.0616	−0.0008	0.0622	0.5617	0.2494	0.0622	1.5622	0.9997	0.5624
8	0.0622	−0.0003	0.0624	0.5622	0.2498	0.0624	1.5624	0.9999	0.5625
9	0.0624	−0.0001	0.0625	0.5624	0.2499	0.0625	1.5625	1.0000	0.5625
10	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625
11	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625
12	0.0625	−0.0000	0.0625	0.5625	0.2500	0.0625	1.5625	1.0000	0.5625

Observe that after 10 iterations the method converges

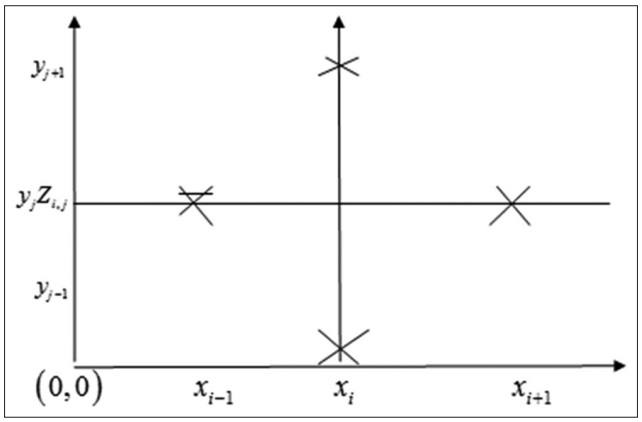


Figure 1: A pictorial representation of the mesh points

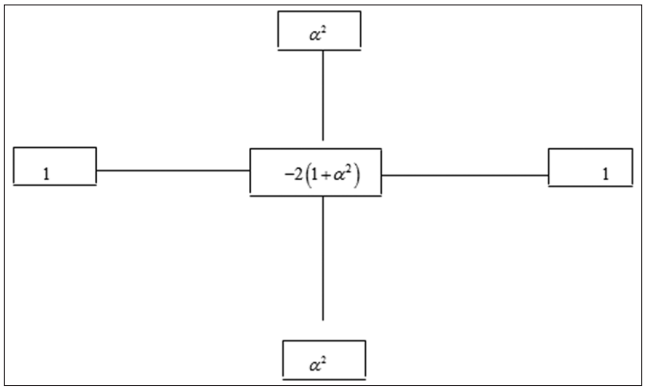


Figure 2: The coefficient of the unknown solutions

problem[5-7]

$$U_{xx} + U_{yy} = 0, \left(0 < x < \frac{1}{2}, 0 < y < \frac{1}{2}\right)$$

$$U(0, y) = 0, U\left(\tfrac{1}{2}, y\right) = 200y$$

$$U(x, 0) = 0, U\left(x, \tfrac{1}{2}\right) = 200x$$

Compare the result with the exact solution U $(x,y) = 400xy$ when

i. TOL=10^{-2}

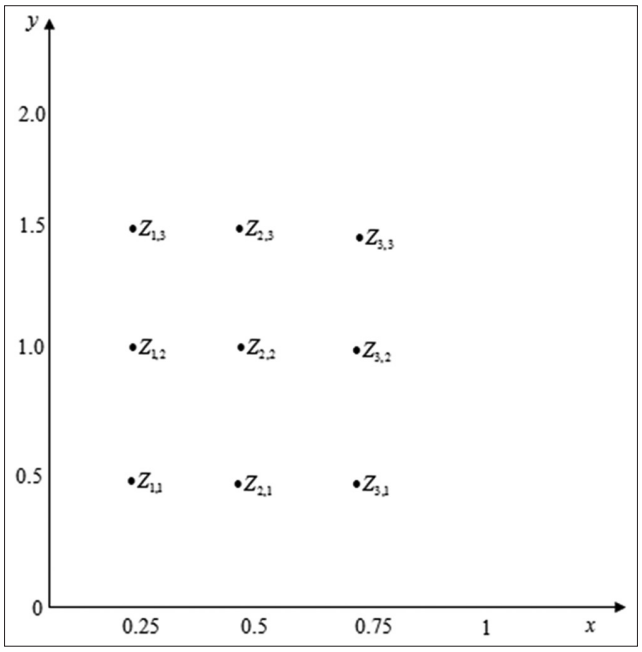


Figure 3: The mesh point of problem 1

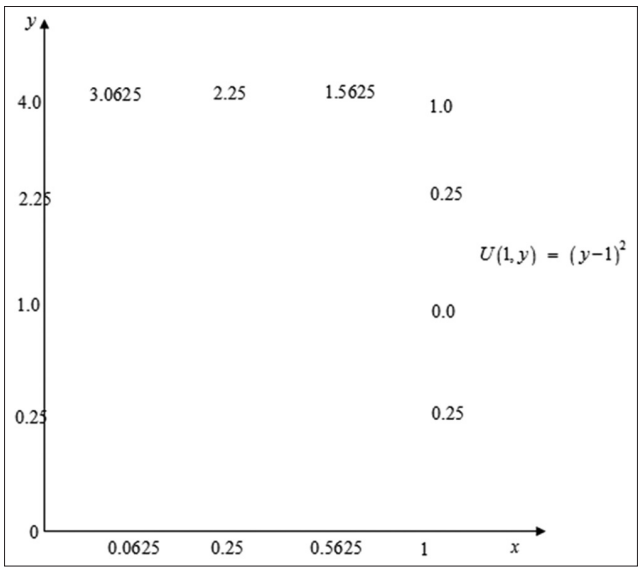


Figure 4: The discrete boundary conditions of problem 1

ii. TOL=10^{-4}

iii. TOL=10^{-6}

Solution

From the above problem; $\alpha = \frac{h}{k} = \frac{1}{8} / \frac{1}{8} = 1$, for $a = 0, b = \frac{1}{2}, c = 0, d = \frac{1}{2}$. The mesh points are $(X_i Y_j)$ where $X_i = \frac{1}{8} i$ and $Y_j = \frac{1}{8} j \;\; i, j = 1,2$

From Equation 30, we have

$$Z_{i\text{-}1,j} + Z_{i+1,j} - 4Z_{i,j} + Z_{i,j\text{-}1} + Z_{i,j+1} = 0 \qquad (29)$$

Since α = 1

Equation 29 holds for each interior mesh point

$$(x_i y_i), i = 1,2,3, j = 1,2,3$$

The discrete boundary conditions associated with Equation 3.3 are

$$Z_{0,j} = 0, j = 0,1,\ldots,4$$

$$Z_{4,j} = \frac{200^*}{8} j, j = 0,1,\ldots,4$$

$$Z_{i,0} = 0$$

$$Z_{i,4} = \frac{200^*}{8} i, i = 0,1,\ldots,4$$

Hence, we have

$$Z_{0,1} = Z_{0,2} = Z_{0,3} = Z_{0,4} = 0$$

$$Z_{1,0} = Z_{2,0} = Z_{3,0} = Z_{4,0} = 0$$

Similarly,

$$Z_{1,4} = 25, Z_{2,4} = 50, Z_{3,4} = 75, Z_{4,4} = 100$$

$$Z_{4,1} = 25, Z_{4,2} = 50, Z_{4,3} = 75, Z_{4,4} = 100$$

Using Equation 4.3 and applying the boundary conditions, we will have the following systems of equations;

$$-4Z_{1,1} + Z_{2,1} + Z_{1,2} = 0$$
$$Z_{1,1} - 4Z_{2,1} + Z_{2,2} + Z_{3,1} = 0$$
$$Z_{2,1} - 4Z_{3,1} + Z_{3,2} = -25$$
$$Z_{1,1} - 4Z_{1,2} + Z_{2,2} + Z_{1,3} = 0$$
$$Z_{2,1} + Z_{1,2} - 4Z_{2,2} + Z_{3,2} + Z_{2,3} = 0$$
$$Z_{3,1} + Z_{2,2} - 4Z_{3,2} + Z_{3,3} = -50$$
$$Z_{1,2} - 4Z_{1,3} + Z_{2,3} = -25$$
$$Z_{2,2} + Z_{1,3} - 4Z_{2,3} + Z_{3,3} = -50$$
$$Z_{3,2} + Z_{2,3} - 4Z_{3,3} = -150$$

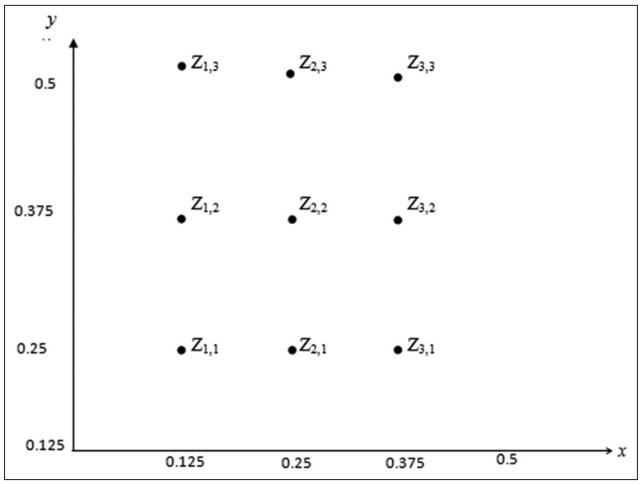


Figure 5: The mesh point of problem 2

The above 9×9 system can be represented in a matrix form as $AZ = B$ where

$$A = \begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \quad Z = \begin{bmatrix} z_{1,1} \\ z_{2,1} \\ \\ \\ \vdots \\ \\ \\ \\ z_{3,3} \end{bmatrix}$$

$$b = [0; 0; -25; 0; 0; -50; -25; -50; -150]$$

The numerical solutions are obtained by running the program in Appendix 3
The result is presented in Table 7.
TOL=10^{-2}

i, j	X_i	Y_i	$U_{(x,y)}$	$Z_{(x,y)}$	Error
1,1	0.1250	0.1250	6.250000	6.247723	2.766×10^{-3}
2,1	0.2500	0.1250	12.500000	12.494469	5.531×10^{-3}
3,1	0.3750	0.1250	18.750000	18.744469	5.531×10^{-3}
1,2	0.1250	0.2500	12.500000	12.497234	2.766×10^{-3}
2,2	0.2500	0.2500	25.000000	24.994469	5.531×10^{-3}
3,2	0.3700	0.2500	37.500000	37.494469	5.531×10^{-3}
1,3	0.1250	0.3750	18.750000	18.748617	1.383×10^{-3}
2,3	0.2500	0.3750	37.500000	37.497234	2.766×10^{-3}
3,3	0.3750	0.3750	56.250000	56.247234	2.766×10^{-3}

The tolerance truncated the iteration at the 10^{th} iteration.

TOL = 10^{-4}

i, j	X_i	Y_i	$U_{(x,y)}$	$Z_{(x,y)}$	Error
1,1	0.1250	0.1250	6.250000	6.249941	$5.9\times10_{-5}$
2,1	0.2500	0.1250	12.500000	12.499951	4.9×10^{-5}
3,1	0.3750	0.1250	18.750000	18.749979	2.1×10^{-5}
1,2	0.1250	0.2500	12.500000	12.499951	4.9×10^{-5}
2,2	0.2500	0.2500	25.000000	24.999959	4.1×10^{-5}

Table 4: Solution of problem 1 case 1 for SOR iteration

i, j	X_i	Y_i	$U_{(x,y)}$	$Z_{(x,y)}$	Error
1,1	0.2500	0.5000	0.0625	0.0622	3×10^{-4}
1,2	0.5000	1.5000	0.0000	0.0003	-3×10^{-4}
1,3	0.7500	1.5000	0.0625	0.0624	1×10^{-4}
2,1	0.2500	1.0000	0.5625	0.5622	4×10^{-4}
2,2	0.5000	1.0000	0.2500	0.2498	2×10^{-4}
2,3	0.7500	1.0000	0.0625	0.0624	1×10^{-4}
3,1	0.2500	0.5000	1.5625	1.5624	1×10^{-4}
3,2	0.5000	1.5000	1.0000	0.9999	1×10^{-4}
3,3	0.7500	1.5000	0.5625	0.5625	0.0000
3,2	0.3700	0.2500	37.500000	37.499983	1.7×10^{-5}
1,3	0.1250	0.3750	18.750000	18.749979	2.1×10^{-5}
2,3	0.2500	0.3750	37.500000	37.499983	1.7×10^{-5}
3,3	0.3750	0.3750	56.250000	56.249993	7.0×10^{-5}

The tolerance truncated the iteration at the 14th iteration.

TOL = 10^{-6}

i, j	X_i	Y_i	$U_{(x,y)}$	$Z_{(x,y)}$	Error
1,1	0.1250	0.1250	6.250000	6.250000	0.00000
2,1	0.2500	0.1250	12.500000	12.500000	0.00000
3,1	0.3750	0.1250	18.750000	18.500000	0.00000
1,2	0.1250	0.2500	12.500000	12.500000	0.00000
2,2	0.2500	0.2500	25.000000	25.000000	0.00000
3,2	0.3700	0.2500	37.500000	37.000000	0.00000
1,3	0.1250	0.3750	18.750000	18.750000	0.00000
2,3	0.2500	0.3750	37.500000	37.500000	0.00000
3,3	0.3750	0.3750	56.200000	56.200000	0.00000

The tolerance stopped the iteration at the 19th iteration.

Problem: Poisson Equation with Nonlinear Inhomogeneous Term

Use the finite difference approximation to obtain a numerical solution to the boundary value problem

$$U_{xx}+U_{yy}=xe^{y},0<x<2,0<y<1,$$

with the boundary conditions

$$U(0,y)=0,U(2,y)=2e^{y},0\le y\le 1$$
$$U(x,0)=x,U(x,1)=ex,=0\le x\le 2$$

Compare the result with the exact solution at
i. 40th iteration
ii. 80th iteration

Use $h=\frac{1}{3}$ and $k=\frac{1}{5}$

Solution

From the above problem $\alpha=\frac{h}{k}=\frac{5}{3}$,

For $a = 0$, $b = 2$, $c = 0$, $d = 1$. The mesh points are x_i, y where, $x_i=\frac{i}{3}$, $y_j=\frac{j}{5}$, $i=1,\ldots,5$ and $j = 1,\ldots,4$.

From the Equation 30, we have

$$9Z_{i-1,j}+25Z_{i,j-1}-68Z_{i,j}+9Z_{i+1,j}+25Z_{i,j+1}=\frac{i}{3}e^{\frac{j}{5}} \quad (31)$$

Equation 32 holds to each interior mesh point (x_i,y_j), $i = 1,2,\ldots5, j = 1,2,\ldots4$

The discrete boundary conditions associated with Equation 33 are

$$Z_{i,0}=\frac{i}{3},\quad Z_{i,1}=\frac{i}{3}e\ ,i=1,\ldots5$$
$$Z_{0,j}=0\ ,Z_{2,j}=2e^{j/5}$$

Applying the boundary conditions and re-arranging the equations, we have

$$-68Z_{1,1}+9Z_{2,1}+25Z_{1,2}=-7.926199081$$
$$9Z_{1,1}-68Z_{2,1}+9Z_{3,1}+25Z_{2,2}=-15.85239816$$
$$9Z_{2,1}-68Z_{3,1}+9Z_{4,1}+25Z_{3,2}=-23.7785924$$
$$9Z_{3,1}-68Z_{4,1}+9Z_{5,1}+25Z_{4,2}=-31.70479631$$
$$9Z_{4,1}-68Z_{5,1}+25Z_{5,2}=-61.61624505$$
$$25Z_{2,1}-68Z_{1,2}+9Z_{2,2}+25Z_{1,3}=0.4972748992$$
$$9Z_{1,2}+25Z_{2,1}-68Z_{2,2}+9Z_{3,2}+25Z_{2,3}=0.9945497084$$
$$9Z_{2,2}+25Z_{3,1}-68Z_{3,2}+9Z_{4,2}+25Z_{3,3}=1.491824698$$
$$9Z_{3,2}+25Z_{4,1}-68Z_{4,1}+9Z_{5,2}+25Z_{4,3}=1.989099507$$
$$9Z_{4,2}+25Z_{5,1}-68Z_{5,2}+25Z_{5,3}=-51.121931462$$
$$25Z_{1,2}-68Z_{1,3}+9Z_{2,3}=22.04497557$$
$$9Z_{1,3}+25Z_{2,2}-68Z_{2,3}+9Z_{3,3}=-44.08995128$$
$$9Z_{2,3}+25Z_{2,3}-68Z_{3,3}+9Z_{4,3}=-66.13492690$$
$$9Z_{3,3}+25Z_{4,2}-68Z_{4,3}+9Z_{5,3}=-88.17990254$$
$$9Z_{4,3}+25Z_{5,2}-68Z_{5,2}=-15.56784400$$
$$25Z_{2,3}+25Z_{2,5}=92.09308829$$
$$25Z_{3,3}+25Z_{3,5}=1381396323$$
$$25Z_{4,3}+25Z_{4,5}=192.34110219$$
$$25Z_{5,3}+25Z_{5,4}=239.1020568$$

Using the program in Appendix 6
The approximation solution was obtained in each case. These are as shown in Tables 8 and 9, respectively.[8]

DISCUSSION OF RESULTS

Effect of iterative methods

We have examined three iterative methods of solving

Table 5: Solution of problem one case 2 for SOR iteration

i, j	X_i	Y_i	$U_{(x, y)}$	$Z_{(x, y)}$	Error
1,1	0.2000	0.2000	0.000000	0.004158	-4.2×10^{-3}
2,1	0.4000	0.2000	0.040000	0.045367	-5.4×10^{-3}
3,1	0.6000	0.2000	0.160000	0.172616	−0.012616
4,1	0.8000	0.2000	0.360000	0.373180	−0.013180
1,2	0.2000	0.4000	0.040000	0.043237	-3.2×10^{-3}
2,2	0.4000	0.4000	0.000000	0.018022	−0.018022
3,2	0.6000	0.4000	0.040000	0.079887	−0.039887
4,2	0.8000	0.4000	0.160000	0.203799	−0.040438
1,3	0.2000	0.6000	0.160000	0.168698	-8.8×10^{-3}
2,3	0.4000	0.6000	0.040000	0.080522	−0.040522
3,3	0.6000	0.6000	0.000000	0.098215	−0.098215
4,3	0.8000	0.6000	0.040000	0.191596	−0.151596
1,4	0.2000	0.8000	0.360000	0.372260	−0.012260
2,4	0.4000	0.8000	0.160000	0.220535	−0.060535
3,4	0.6000	0.8000	0.040000	0.217418	−0.177418
4,4	0.8000	0.8000	0.000000	0.469614	−0.469614
1,5	0.2000	1.0000	0.640000	0.645682	-5.7×10^{-3}
2,5	0.4000	1.0000	0.360000	0.398618	−0.038618
3,5	0.6000	1.0000	0.160000	0.259080	−0.099080
4,5	0.8000	1.0000	0.040000	0.194428	−0.159443
1,6	0.2000	1.2000	1.000000	0.998192	1.8×10^{-3}
2,6	0.4000	1.2000	0.640000	0.654943	−0.014943
3,6	0.6000	1.2000	0.360000	0.402175	−0.042175
4,6	0.8000	1.2000	0.160000	0.213012	−0.053012
1,7	0.2000	1.4000	1.440000	1.434826	5.2×10^{-3}
2,7	0.4000	1.4000	1.000000	1.001978	-1.9×10^{-3}
3,7	0.6000	1.4000	0.640000	0.654473	−0.014473
4,7	0.8000	1.4000	0.360000	0.378063	−0.018063
1,8	0.2000	1.6000	1.960000	1.955450	4.6×10^{-3}
2,8	0.4000	1.6000	1.440000	1.438198	1.8×10^{-3}
3,8	0.6000	1.6000	1.000000	1.003983	-3.9×10^{-3}
4,8	0.8000	1.6000	0.640000	0.645985	-5.9×10^{-3}
1,9	0.2000	1.8000	2.560000	2.557913	2.1×10^{-3}
2,9	0.4000	1.8000	1.960000	1.958924	1.1×10^{-3}
3,9	0.6000	1.8000	1.440000	1.441153	-1.2×10^{-3}
4,9	0.8000	1.8000	1.000000	1.001964	-1.9×10^{-3}

linear systems of equation, after a thorough and comprehensive study on this research work, it was observed that , Equation 30, from the result in Table 2 that Jacobi method takes longer time to converge Equation 32, even at the 25th iteration it was still struggling to converge and become stable for the 9×9 linear system obtained by the application of finite difference method to the elliptic PDE. Gauss-Seidel converged at 18th iteration in Table 3 while SOR has 10th iteration in Table 4. Thus, it is advisable to use the SOR method for solving a system of linear equations because it requires less computation time to converge as compared to the other two iterative methods used Equation 33.[9]

Table 6: Solution of problem three at 40th iteration

i, j	X_i	Y_i	$Z_{(x, y)}$	$U_{(x, y)}$	Error
1,1	0.3333	0.2000	0.40666	0.40713	4.7
2,1	0.6667	0.2000	0.81382	0.81427	4.5×10^{-4}
3,1	1.0000	0.2000	1.22090	1.22140	3.0×10^{-4}
4,1	1.3333	0.2000	1.62630	1.62850	2.0×10^{-4}
5,1	1.6667	0.2000	2.03510	2.03570	6.0×10^{-4}
1,2	0.3333	0.4000	0.49658	0.49727	6.9×10^{-4}
2,2	0.6667	0.4000	0.99396	0.99455	5.9×10^{-4}
3,2	1.0000	0.4000	1.49130	1.49180	5.0×10^{-4}
4,2	1.3333	0.4000	1.98880	1.98910	3.0×10^{-4}
5,2	1.6667	0.4000	2.48620	2.48640	2.0×10^{-4}
1,3	0.3333	0.6000	0.60670	0.60737	6.7×10^{-4}
2,3	0.6667	0.6000	1.21460	1.21470	1.0×10^{-4}
3,3	1.0000	0.6000	1.82180	1.82210	3.0×10^{-4}
4,3	1.3333	0.6000	2.42820	2.42950	1.3×10^{-4}
5,3	1.6667	0.6000	3.03660	3.03691	3.1×10^{-4}
1,4	0.3333	0.8000	0.74129	0.74185	5.6×10^{-4}
2,4	0.6667	0.8000	1.48340	1.48370	3.0×10^{-4}
3,4	1.0000	0.8000	2.22530	2.22550	2.0×10^{-4}
4,4	1.3333	0.8000	2.96690	2.96740	5.0×10^{-4}
5,4	1.6667	0.8000	3.7088	3.70920	4.0×10^{-4}

Table 7: Solution of problem three at 80th iteration.

i, j	X_i	Y_j	$Z_{(x, y)}$	$U_{(x, y)}$	Error
1,1	0.3333	0.2000	0.40726	0.40713	-1.3×10^{-4}
2,1	0.6667	0.2000	0.81452	0.81427	-2.5×10^{-4}
3,1	1.0000	0.2000	1.22180	1.22140	-4.0×10^{-4}
4,1	1.3333	0.2000	1.62900	1.62850	-5.0×10^{-4}
5,1	1.6667	0.2000	2.03600	2.03570	-3.0×10^{-4}
1,2	0.3333	0.4000	0.49748	0.49727	-2.1×10^{-4}
2,2	0.6667	0.4000	0.99496	0.99455	-4.1×10^{-4}
3,2	1.0000	0.4000	1.49240	1.49180	-6.0×10^{-4}
4,2	1.3333	0.4000	1.98980	1.98910	-7.0×10^{-4}
5,2	1.6667	0.4000	2.48700	2.48640	-6.0×10^{-4}
1,3	0.3333	0.6000	0.60760	0.60737	-2.3×10^{-4}
2,3	0.6667	0.6000	1.21520	1.21470	-5.0×10^{-4}
3,3	1.0000	0.6000	1.82270	1.82211	-5.9×10^{-4}
4,3	1.3333	0.6000	2.43020	2.42950	-7.0×10^{-4}
5,3	1.6667	0.6000	3.03750	3.03690	-6.0×10^{-4}
1,4	0.3333	0.8000	0.74210	0.74185	-2.5×10^{-4}
2,4	0.6667	0.8000	1.48400	1.48370	-3.0×10^{-4}
3,4	1.0000	0.8000	2.22600	2.22550	-5.0×10^{-4}
4,4	1.3333	0.8000	2.96790	2.96740	-5.0×10^{-4}
5,4	1.6667	0.8000	3.70970	3.70920	-5.0×10^{-4}

Effect of mesh size

From the result in Tables 5 and 6, more accuracy was attained by increasing the number of mesh points from 9 to 36 which resulted by reducing the mesh size. For the nine mesh points, the approximations were accurate up to one decimal place when compared with the actual solution.

However, in Table 6, where there are 36 mesh points, the solutions were accurate up to two decimal places and some three decimal places by comparison with the actual solution. However, increasing the mesh size requires more iteration.

Effect of tolerance value

It was observed from the result in Table 7 that a better approximation is obtained by a reduction in the tolerance value. When the tolerance value as defined in Table 6 was reduced from 10^{-2} to 10^{-4}, more accurate approximation was obtained. At a tolerance value of 10^{-6} actual solutions were obtained. This is because the round off error accumulates more error when tolerance 10^{-2} and 10^{-4} were used than when the tolerance 10^{-6} is used.

Effects of iteration values

From Tables 8 and 9, it could be deduced that more accurate approximations are obtained by increasing the number of iterations. At 40th iteration, the solutions were accurate up to two decimal places, but at 80th iteration, the solutions became accurate up to three decimal places. A general observation here shows that at some points, the solution values at 80th iterations are larger than the actual solutions while at 40th iteration they are smaller.

The summary of the result

1. Use of the SOR method converges faster
2. Many no of mesh points more accurate but high number of iterations
3. The lower value of tolerance better accuracy

CONCLUSION

In the course of this research work, we were trying to solve the PDE of elliptic type with the finite difference method. In this method, we make the area of integration of the elliptic equation to be overlaid by a system of rectangular meshes by two sets of equally spaced lines on sets of parallel to X – axis and the other parallel to Y.

An approximated solution to PDE is found at the point of intersection of the lines which are called mesh points. The solution is obtained by approximating the PDE over the area by n-algebraic equations involving the values of the

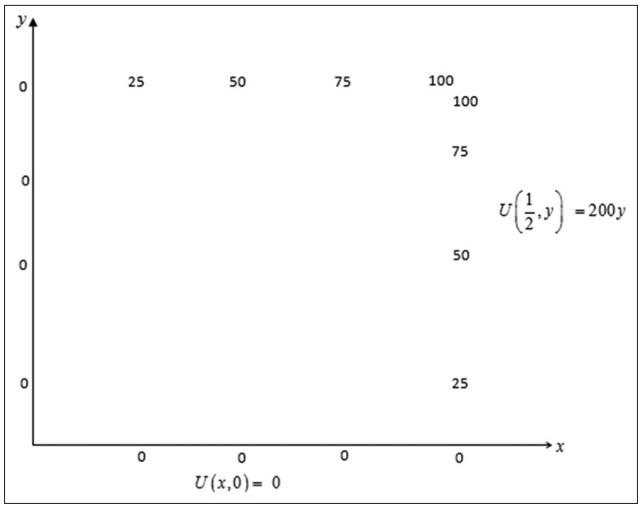


Figure 6: T The discrete boundary conditions of problem 2

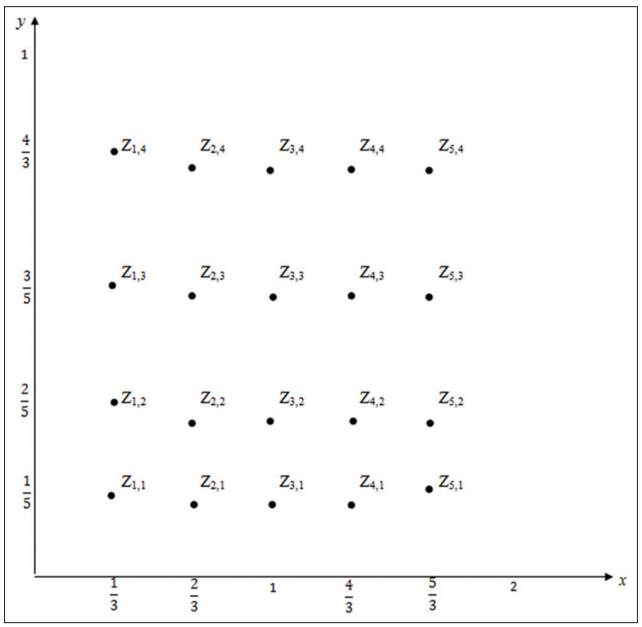


Figure 7: The mesh point of problem 3

solution at the n-mesh point interval of the region of integration. Then, for each of the n-interval mesh points, the algebraic equations approximating the PDE are written down.[10-12]

The system of algebraic equations obtained by applying the finite difference approximation to the PDE was solved with the three iterative methods in problem one case 1 while in problem one case 2, problem 2, and problem 3, the systems were solved with the use of SOR method due to its less computational time to converge. The better approximation was obtained using many mesh points, calculating each approximation for several times and by reducing the tolerance value. A computer program (MATLAB programming) was used in making the computation within some seconds provided that the program is well posed. The main advantage of numerical over

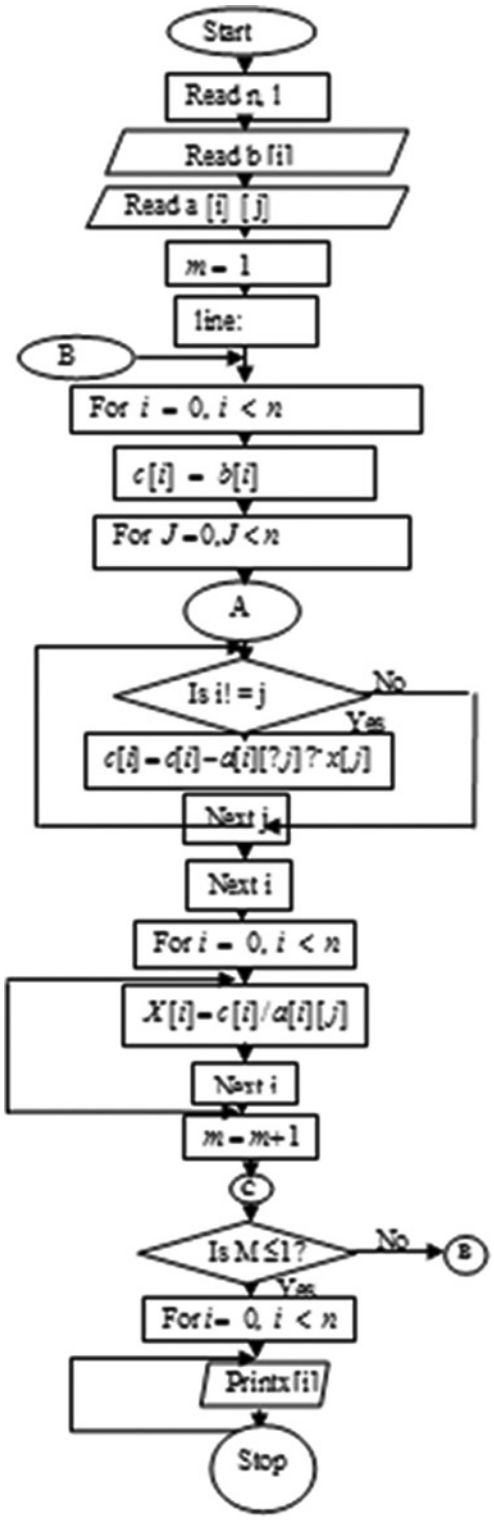
other methods such as analytical and graphical methods is that it allows us to solve such equation like elliptic PDE s with more ease to obtain the approximate solutions which are useful in scientific research.

REFERENCES

1. Tveito A, Winther R. Introduction to Partial Differential Equation, a Computational Approach. Vol. 29. Berlin: Springer; 1998.
2. Oruh BI. MTH 423 Lecturer Note on Numerical Analysis II. Nigeria: MOUAU; 2012.
3. Weisstein EW. Elliptic Partial Differential Equation. New York: John Wiley; 1993.
4. Kreyszig E. Advanced Engineering Mathematics. 7th ed. New York: John Wiley and Sons Inc.; 1997.
5. Stephenson G. Partial Differential Equation for Scientists and Engineers. 3rd ed. Hong Kong: Longman; 1968.
6. Smith GD. Numerical Solution of Partial Differential Equations. Oxford: Oxford University Press; 1995.
7. Kalambi I. Comparison of three iterative methods for the solution of linear equations. J Appl Sci Environ 2008;12:53-5.
8. Moon P, Spencer DE. Resent investigation of separation of Laplace's equation. Proc Am Maths Soc 1953;302:4.
9. Turner PR. Guide to Numerical Analysis. Hong Kong: Macmillan Education Ltd.; 1989.
10. Scheid F. Numerical Analysis. Schaum's Series. New York: McGraw Hill; 1988.
11. Chapra SC, Canale RP. Numerical Methods for Engineer. U.S.A: WCB/McGraw-Hill; 1998.
12. Strauss WA. Partial Differential Equation, an Introduction. New York: John Wiley and Sons; 1992.

APPENDIX

Appendix 1: JACOBI FLOWCHART



Appendix 2: JACOBI ITERATIVE METHOD: Algorithm2.1;

Step 1; Read n l (n is the number of iteration)
Step 2; Read bi, i = 0, 1, 2., (n−1) (RHS constant)
Step 3; For i = 0(1)(n−1), do till (8)
Step 4; X 1 ← 0
Step 5; For j = 0(1)(n−1), do till (7)
Step 6; Read aij
Step 7; Next j
Step 8; Next i
Step 9; m1
Step 10; For i = 0(1) (n−1), do till (6)
Step 11; C i ← b
Step 12; For j = 0(1) (n−1), do till (15)
Step 13; if i 6= j
Step 14; C i ← −a ij×j
Step 15; Else, next j
Step 16; Next i
Step 17; For i = 0(1) (n−1), do till (19)
Step18; X 1 ←C i/a ij
Step 19; Next i
Step 20;M ← m+1
Step 21; If M ≤ L
Step 22; Go to (10)
Step 23; Else,
Step 24; For i = 0(1) (n−1), do till (26)
Step 25; Write X 1
Step 26; Next i
Step 27; End

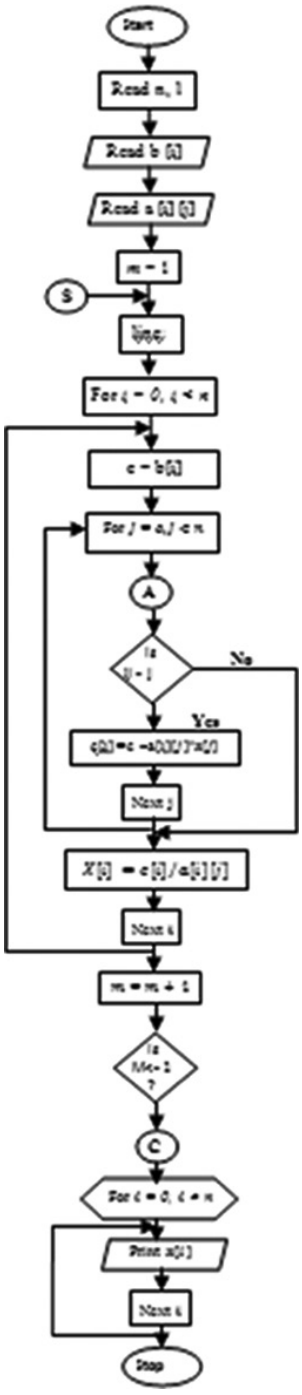
Appendix 3: MATLAB M-FILE FOR JACOBI ITERATIVE METHOD WITHOUT TOLERANCE

```
function[x,J,c]= jacobi n o t ol(A,b,n,z)
%
% x = jacobi(A,b,n,z)
%
% Jacobi iteration on system A*x = b with printing
% n – number of iterations
% z – initial vector (default 0)
%
% x – final iterate
% J – Jacobi matrix
% c – Jacobi vector
%
ifnargin<=3, z=0*b; end
D = diag(diag(A));
J = D\(D−A);
c = D;
x=z;
for k = 1:n
x = J*x + c;
fprintf(1,'%3d ',k)
fprintf(1,'%5.6f ',x')
fprintf(1, ′ \ n')
End
```

Appendix 4: MATLAB M-FILES FOR THE JACOBI ITERATIVE METHOD WITH TOLERANCE MATLAB SCRIPT

```
functionxnew = jacobi(A,xold,b,maxits,tol)
M = diag(diag(A)); N = M-A;
for k = 1:maxits
xnew = M\(N*xold+b);
if norm(xnew-xold)<tol + eps(norm(xnew))
return
end;
xold = xnew; fprintf(1,'%3d ',k)
fprintf(1,'%5.6f ',xold')
fprintf(1,'\ n')
end;
```

Appendix 5: GAUSS-SEIDEL FLOWCHART



Appendix 6: GAUSS-SEIDEL ITERATIVE METHOD; Algorithm2.2

Step 1; Read n,l (l is the number of iteration)
Step 2; Read bi, i=0,1,2.,(n-1)(RHS constant)
Step 3; For i=0 (1)(n-1) do till (8)
Step 4; Xi← 0
Step 5; For j=0 (1)(n-1), do till (7)
Step 6; Read aij
Step 7; Next j
Step 8; Next i
Step 9; M← 1
Step 10; For i=0(1)(n-1), do till (16)
Step 11; C← bi
Step 12; For j=0(1)(n-1), do till (15)
Step 13; If i6= j c ← c−a ij×j
Step 14; Else, next j
Step 15; Xi← c/a ij
Step 16; Next i
Step 17; M← m+i
Step 18; If m≤ i
Step 19; Go to 10
Step 20; Else
Step 21; For i=0(1)(n-1) do till 24
Step 22; Write xi
Step 23; Next i
Step 24; End

Appendix 7: MATLAB M-FILEF OR GAUSS-SEIDEL ITERATIVE METHOD WITHOUT TOLERANCE

```
function [x,G,c] = gsmp(A,b,n,z)
%
% x = gsmp(A,b,n,z)
%
% Gauss-Seidel iteration on system A*x = b with
   printing
% using matrix multiplication (not optimal)
% n – number of iterations
% z – initial vector (default 0)
%
% x – final iterate
% G – Gauss-Seidel matrix
% c – Gauss-Seidel vector
%
ifnargin<=3, z=0*b; end
LD = tril(A);
G = -LD\triu(A,1);
c = LD;
‾

x=z;
for i = 1:n
x = G*x + c;
fprintf(1,'fprintf(1,'fprintf(1,'\n')
end
```

Appendix 8: GAUSS-SEIDEL ITERATIVE METHOD WITH TOLERANCE MATLAB SCRIPT

```
function x = Gauss-Seidel(A,xold,b,maxits,tol)
D = diag(diag(A)); L = tril(A)- D;
U = A-D-L;
M = D + L; N = -U;
bp = b;
x = xold;
for k = 1:maxits
x = M\(N*x+bp);
if norm(x-xold)<tol + eps(norm(x))
return
end;
xold = x;
fprintf(1,'%3d ',k)
fprintf(1,'%5.6f ',xold')
fprintf(1,'\n')
end;
```

Appendix 9: SOR ITERATIVE METHOD; Algorithm 2.3

Step 1; Input: A, b, w
Step 2; Output: ∅
Step 3; Choose an initial guess ∅ to the solution
Step 4; Repeat until convergence
Step 5; for i from 1 until n do
σ ← 0
Step 6; for j from 1 until n do
Step 7; if j 6= i then
Step 8; σ ← σ + a i,j ∅j
Step 9; end if (j← loop)
Step 10; ∅ i ← (1-w) ∅ i +
w/a
ii
(bi- σ)
Step 11; end if (i←loop)
Step 12; Check convergence if reached
Step 13; end (report)

Appendix 10: MATLAB M-FILE FOR SUCCESSIVE OVER RELAXATION ITERATIVE METHOD WITHOUT TOLERANCE

```
function x = sor1(A,b,n,w,z)
%
% x = sor(A,b,n,w,z)
%
% SOR iteration on system A*x = b with printing
% n – number of iterations
% w – SOR parameter
% z – initial vector (default 0)
%
% x – final iterate
%
ifnargin<=4, z=0*b; end
m,l= size(A);
D = diag(diag(A));
J = D\(D - A);
c = D\b;
x=z;
for k = 1:n
for i=1:m
x(i,1) = (1-w)*x(i,1) + w*(J(i,:) * x + c(i,1));
end
fprintf(1,'%3d ',k)
fprintf(1,'%5.6f ',x')
fprintf(1,'\n')
end
25
```

Appendix 11: SORITERATIVE METHOD WITH TOLERANCE MATLAB SCRIPT

```
function x = SOR(A,xold,b,w,maxits,tol)
D = diag(diag(A)); L = tril(A)- D;
U = A-D-L;
M = D + w*L; N = (1-w)*D-w*U;
bp = w*b;
x = xold;
for k = 1:maxits
x = M\(N*x+bp);
if norm(x-xold)<tol + eps(norm(x))
return
end;
xold = x;
fprintf(1,'%3d ',k)
fprintf(1,'\n')
end;
```