

An-Najah National University

Faculty of Graduate Studies

**Finite Difference and Finite Element
Methods for Solving Elliptic Partial
Differential Equations**

By

Malik Fehmi Ahmed Abu Al-Rob

Supervisor

Prof. Naji Qatanani

**This Thesis is Submitted in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Mathematics, Faculty of Graduate
Studies, An-Najah National University, Nablus, Palestine.**

2016

**Finite Difference and Finite Element Methods for
Solving Elliptic Partial Differential Equations**

By

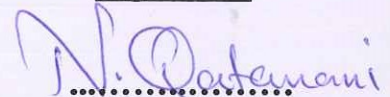
Malik Fehmi Ahmed Abu Al-Rob

This thesis was defended successfully on 20 /1/2016 and approved by :

Defense committee members

Signature

- Prof. Naji Qatanani / Supervisor


.....

- Dr. Sae'd Mallak / External Examiner


.....

- Dr. Anwar Saleh / Internal Examiner


.....

III

بسم الله الرحمن الرحيم

" يرفع الله الذين اوتوا العلم منكم درجات والله بما تعملون خبير "

صدق الله العظيم

Dedication

I dedicate my work to all my family members, to my parents, my sister, my brothers who encourage me to learn, grow and develop and who have been a source of encouragement and inspiration to me.

I also dedicate this dissertation to my homeland Palestine.

Acknowledgement

In the beginning, I am grateful to the God to complete this thesis. I wish to express my sincere thanks to Prof. Naji Qatanani for providing me with all the necessary facilities for the research and I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

My thanks also to my internal examiner Dr. Anwar Saleh and my external examiner Dr. Saed Mallak from Palestine Technical University – Kadoorie for their positive comments.

I am extremely thankful to Mr. Hayel Hussein from the Arab American University who directly help me to complete this thesis.

I take this opportunity to express gratitude to all my family members for their help and support. I also thank my parents for the unceasing encouragement, support and attention.

My sense of gratitude to everyone, who directly or indirectly, have lent their hand in this venture.

الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل عنوان :

Finite Difference and Finite Element Methods for Solving Elliptic Partial Differential Equations

أقر بأن ما اشتملت عليه هذه الرسالة إنما هي نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل أي درجة علمية أو بحث علمي أو بحثي لدى أية مؤسسة تعليمية أو بحثية أخرى.

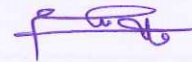
Declaration

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

Student's name:

اسم الطالب: مالك فهد أحمد أبو الرب

Signature:

التوقيع: 

Date:

التاريخ: 20/11/2016

Contents

Dedication	IV
Acknowledgement.....	V
Declaration	VI
Contents.....	VII
Abstract	IX
Introduction	1
Chapter One.....	5
Finite Difference and Finite Element Methodsfor Solving Elliptic Partial Differential Equations.....	5
1.1 Discretization of Elliptic PDE by Finite Difference Method.....	6
1.2 The Principle of Finite Difference Method:.....	7
1.3 Strategy of Discretization.....	8
1.4 Elliptic PDE subject to Boundary Conditions:	13
1.4.1 Laplace equation with Dirichlet Boundary Conditions:	13
1.4.2 Poisson Equation with Dirichlet Boundary Conditions:	18
1.4.3 Laplace Equation with Neumann Boundary Conditions:	18
1.4.4 Poisson equation with Neumann Boundary Conditions:	20
1.5 Finite Element Method:.....	21
1.6 The Principle of Finite Element Method:	21
1.6.1 Finite Element Method for Dirichlet boundary value problems: 22	
1.6.2 Finite Element Method with Neumann Boundary condition:	29
Chapter Two.....	33
Iterative Methods for Solving Linear systems	33
2.1 Jacobi Method.....	34
2.2 Gauss-Seidel Method.....	37
2.3 Successive over Relaxation Method (SOR Method).....	40
2.4 Conjugate Gradient Method.....	42
2.5 Convergence of Iterative Methods.....	45

Chapter Three.....	52
Numerical Results.....	52
Example 3.1	52
Example 3.2	64
Example 3.3	71
Example 3.4	84
Example 3.5	86
3.1 Comparison between results for finite difference method and finite element method:.....	89
3.2 Conclusions.....	90
References	91
Appendix A	94
Appendix B	96
Appendix C	98
Appendix D	101
Appendix E.....	104
Appendix F.....	106
Appendix G	108
Appendix H	112
الملخص.....	ب

IX
**Finite Difference and Finite Element Methods for Solving Elliptic
Partial Differential Equations**

By
Malik Fehmi Ahmed Abu Al-Rob
Supervisor
Prof. Dr. Naji Qatanani

Abstract

Elliptic partial differential equations appear frequently in various fields of science and engineering. These involve equilibrium problems and steady state phenomena. The most common examples of such equations are the Poisson's and Laplace equations. These equations are classified as second order linear partial differential equations.

Most of these physical problems are very hard to solve analytically, instead, they can be solved numerically using computational methods.

In this thesis, boundary value problems involving Poisson's and Laplace equations with different types of boundary conditions will be solved numerically using the finite difference method (FDM) and the finite element method (FEM).

The discretizing procedure transforms the boundary value problem into a linear system of n algebraic equations. Some iterative techniques, namely: the Jacobi, the Gauss-Seidel, Successive over Relaxation (SOR), and the Conjugate Gradient method will be used to solve such linear system. Numerical results show that the finite difference method is more efficient than the finite element method for regular domains, whereas the finite element method is more accurate for complex and irregular domains. Moreover, we observe that the SOR iterative technique gives the most efficient results among the other iterative schemes.

Introduction

Many physical phenomena and engineering problems involving temperature, electrical potential, astronomy and membrane displacement can be described by elliptic partial differential equations.

The boundary value problems modeling these physical phenomena are too hard to be solved analytically. Alternatively, we can use some computational methods to solve such problems.

The use of Finite Difference Method (FDM) depends upon Taylor expansion to approximate the solution of partial differential equation (PDE) that uses a regular shape of network of lines to construct the discretization of the PDE. This is a potential bottleneck of the method when handling complex geometries in multiple dimensions. This issue motivated the use of an integral form of the PDEs and subsequently the development of the Finite Element Method (FEM) [22].

On the other hand, the FEM is the most general method for the numerical solution of the three types of partial differential equations, namely: elliptic, parabolic, and hyperbolic equations.

This method was introduced by engineers in the late 50's and early 60's for the numerical solution of partial differential equations in structural engineering (elasticity equations, plate equations, and so on) [9].

At that point of time, this method was thought of as a generalization of earlier methods in structural engineering for beams, frames, and plates where the structure was subdivided into small parts, so called finite elements.

When the mathematical study of the finite element method started in the mid 60's, it soon became clear that the method is a general technique for the

numerical solution of partial differential equations with roots in the variational methods in mathematics introduced in the beginning of the century.

The FEM dates back to 1909 when Ritz developed an effective method for the approximate solution of problems in the mechanics of deformable solids. It includes an approximation of energy functional by the known functions with unknown coefficients. Minimization of functional in relation to each unknown leads to a system of equations from which the unknown coefficients may be determined. One of the main restrictions in the Ritz method is that functions used should satisfy the boundary conditions of the problem [7].

In 1943 Courant considerably increased possibilities of the Ritz method by introducing special linear functions defined over triangular regions and applied the method for the solution of torsion problems. As unknowns, the values of functions in the node points of triangular regions were chosen. Thus, the main restriction of the Ritz functions – a satisfaction to the boundary conditions was eliminated [7]. The Ritz method together with the Courant modification is similar with FEM proposed independently by Clough many years later introducing for the first time in 1960 the term “finite element” in the paper “The finite element method in plane stress analysis” [7]. The main reason of wide spreading of FEM in 1960 is the possibility to use computers for the big volume of computations required by FEM. However, Courant did not have such possibility in 1943 [7].

An important contribution was brought into FEM development by the papers of Argyris, Turner, Martin, Hrennikov and many others [23]. The first book on FEM was published in 1967 by Zienkiewicz and Cheung and

called “The finite element method in structural and continuum mechanics”[23]. This book presents the broad interpretation of the method and its applicability to any general field problems. Although the method has been extensively used previously in the field of structural mechanics, it has been successfully applied now for the solution of several other types of engineering problems like heat conduction, fluid dynamics, electric and magnetic fields, and others [23].

On the other hand, the finite difference method was invented by a Chinese scientist named Feng Kang in the late 1950’s. He proposed the finite difference method as a systematic numerical method for solving partial differential equations that are applied to the computations of dam constructions. It is speculated that the same method was also independently invented in the west, named in the west the FEM. It is now considered that the invention of the finite difference method is a milestone of computational mathematics.

Error bounds for difference approximations of elliptic problems were first derived by Gerschgorin (1930) whose work was based on a discrete analogue of the maximum principle for Laplace’s equation. This approach was actively pursued through the 1960s by Collatz, Motzkin, Wasow, Bramble, and Hubbard, and various approximations of elliptic equations and associated boundary conditions were analyzed [19].

For time-dependent problems, considerable progress in finite difference methods was made during the period of, and immediately following, the second world war, when large-scale practical applications became possible with the aid of computers [19]. A major role was played by the work of von Neumann, partly reported in O’Brien, Hyman and Kaplan (1951). For

parabolic equations a highlight of the early theory was the important paper by John (1952) [19]. For mixed initial–boundary value problems the use of implicit methods was also established in this period by, e.g., Crank and Nicolson (1947). The finite difference theory for general initial value problems and parabolic problems then had an intense period of development during the 1950s and 1960s, when the concept of stability was explored in the Lax equivalence theorem and the Kreiss matrix lemmas, with further major contributions given by Douglas, Lees, Samarskii, Widlund and others [19]. For hyperbolic equations, and particularly for nonlinear conservation laws, the finite difference method has continued to play a dominating role up until the present time, starting with work by Friedrichs, Lax, Wendroff, and others.

Some standard references on finite difference methods are the textbooks of Collatz, Forsythe and Wasow and Richtmyer and Morton [19].

This thesis is organized as follows:

Chapter one introduces both the finite difference method and the finite element method used to solve elliptic partial differential equations. The discretization procedure for partial differential equations and boundary conditions are represented explicitly. In chapter two, some iterative techniques namely, the Jacobi, the Guass-Seidel, Successive over Relaxation (SOR), and the Conjugate Gradient method are presented together with their convergence properties. Chapter three contains some numerical examples and concluding results.

Chapter One

Finite Difference and Finite Element Methods for Solving Elliptic Partial Differential Equations

Physical and engineering problems such as equilibrium problems and steady state phenomena (independent of time) can be described as elliptic partial differential equations (elliptic PDEs). These equations express the behavior of such problems. Second order linear partial differential equations are mainly considered as

$$A \frac{\partial^2 u}{\partial x^2} + B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^2 u}{\partial y^2} + D \frac{\partial u}{\partial x} + E \frac{\partial u}{\partial y} + F u = G(x, y)$$

or simply

$$A u_{xx} + B u_{xy} + C u_{yy} + D u_x + E u_y + F u = G(x, y) \quad (1.1)$$

where A, B, C, D, E, F , and the free term G are the coefficients of Eq. (1.1) which can be constants or functions of two independent variables x and y and u is the unknown function of two independent variables x and y .

Eq. (1.1) is classified into three types depending on the discriminant $(B^2 - 4AC)$ as follows:

1. Hyperbolic if the discriminant is positive ($B^2 - 4AC > 0$).
2. Parabolic if the discriminant is zero ($B^2 - 4AC = 0$).
3. Elliptic if the discriminant is negative ($B^2 - 4AC < 0$).

We will deal with elliptic PDEs (or in general, with steady state problems) with respect to two types of boundary conditions. These conditions are:

1. Dirichlet Boundary Condition:

The condition where the value of the unknown function is prescribed on the boundary of the domain.

2. Neumann Boundary Condition:

The condition where the value of the normal derivative $\frac{\partial u}{\partial n}$ is given on the boundary of the domain.

In this thesis, we use Finite Difference and Finite Element methods for solving elliptic partial differential equations in two dimensions such as Laplace equation and Poisson equation.

When these techniques are used for solving elliptic PDEs, a system of linear equations will be generated and should be solved using several iterative schemes such as Jacobi, Gauss-Seidel, Successive over Relaxation (SOR), and Conjugate Gradient methods.

1.1 Discretization of Elliptic PDE by Finite Difference Method

The Finite Difference Method (FDM) is a well-known method that is used to approximate the solution of partial differential equations. It was already known by L. Euler (1707-1783) in one dimension of space and was probably extended to dimension two by C. Runge (1856-1927). This method is effective when the domain of the problem has boundaries with regular shapes. In this thesis, we will deal with the finite difference method with rectangular domain of regular boundaries shapes.

1.2 The Principle of Finite Difference Method:

The idea of FDM is to replace the partial derivatives of dependent variable (unknown function) with partial differential equation using finite difference approximations with $O(h^n)$ errors. This procedure converts the region (where the independent variables in PDE are defined on) to a mesh grid of points where the dependent variables are approximated. The replacement of partial derivatives with difference approximation formulas depends on Taylor's Theorem. So, Taylor's Theorem is introduced.

Taylor's Theorem 1.2.1

Let $u(x)$ has $n \in \mathbb{N}$ continuous derivatives over the interval (a,b) . Then, for $a < x_0, x_0 + h < b$, we can write the value of $u(x)$ and its derivatives nearby the point $x_0 + h$ as follows:

$$u(x_0 + h) = u(x_0) + h \frac{u_x(x_0)}{1!} + h^2 \frac{u_{xx}(x_0)}{2!} + h^3 \frac{u_{xxx}(x_0)}{3!} + \dots + h^{n-1} \frac{u_{n-1}(x_0)}{(n-1)!} + O(h^n) \quad (1.2)$$

where

1. $u_x(x_0)$ is the first derivative of u with respect to x at the point x_0 .
2. $u_{n-1}(x_0)$ is the $n-1^{\text{th}}$ derivative of u with respect to x at the point x_0 .
3. $O(h^n)$ [pronounced as order h to the n] is an unknown error term that satisfies the property: for $f(h) = O(h^n)$

$$\lim_{h \rightarrow 0} \frac{f(h)}{h^n} = c$$

for any nonzero constant c .

When we eliminate the error term, $O(h^n)$, from the right-hand side of Eq. (1.2), we get an approximation to $u(x_0+h)$.

1.3 Strategy of Discretization

Using finite difference method to discretize elliptic PDE with its boundary conditions, we can consider the following Poisson equation:

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = G(x,y)$$

or we can simply write this equation in another form as:

$$u_{xx} + u_{yy} = G(x,y), \text{ for } (x,y) \in R \quad (1.3)$$

The rectangular domain $R = \{(x,y) \mid a < x < b, c < y < d\}$ and

$u(x,y) = g(x,y)$ for any $(x,y) \in S$, where:

S denotes the boundary of a region R , $G(x,y)$ is a continuous function on R and $g(x,y)$ is continuous on S . The continuity of both G and g guarantees a unique solution of Eq. (1.3).

Now, we will use the finite difference algorithm for solving elliptic PDE, like Eq. (1.3).

The Finite Difference Algorithm

Step 1: Choose positive integers n and m .

Step 2: Define $h = \frac{b-a}{n}$ and $\Delta = \frac{d-c}{m}$.

This step partitions the interval $[a,b]$ into n equal parts of width h and partitions the interval $[c,d]$ into m equal parts of width k as step 3 illustrates.

Step 3: Define

$$x_i = a + ih, \quad i = 0, 1, 2, \dots, n$$

$$y_j = c + jk, \quad j = 0, 1, 2, \dots, m$$

Step 2 and step 3 are illustrated in figure 1.1.

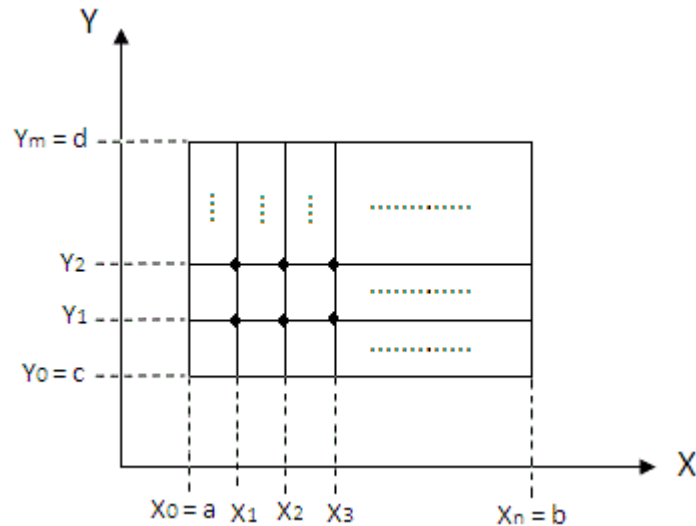


Figure 1.1

It is clear from figure 1.1 that we have horizontal and vertical lines inside the rectangle R . These lines are called "grid lines" and their intersections are called "mesh points" of the grid. For each mesh point inside the grid, (x_i, y_j) , $i = 1, 2, \dots, n-1$ and $j = 1, 2, \dots, m-1$ [2],[21]. We use Taylor series in the variable x about x_i to generate the central-difference formula:

$$u_{xx}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j))}{h^2} - \frac{h^2}{12} \cdot \frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) \quad (1.4)$$

where $\xi_i \in (x_{i-1}, x_{i+1})$.

Also, we use Taylor series in the variable y about y_j to generate the central-difference formula:

$$u_{yy}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1}))}{k^2} - \frac{k^2}{12} \cdot \frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) \quad (1.5)$$

where $\eta_j \in (y_{j-1}, y_{j+1})$ [15].

By inserting Eq. (1.4) and Eq. (1.5) into Eq. (1.3), we get:

$$\begin{aligned} & \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j))}{h^2} - \frac{h^2}{12} \cdot \frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \\ & \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1}))}{k^2} - \frac{k^2}{12} \cdot \frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) = G(x_i, y_j) \end{aligned} \quad (1.6)$$

for each $i = 1, 2, 3, \dots, n-1$ and $j = 1, 2, 3, \dots, m-1$.

The boundary conditions are:

1. $u(x_0, y_j) = g(x_0, y_j), \quad \forall j = 0, 1, 2, \dots, m.$
 2. $u(x_n, y_j) = g(x_n, y_j), \quad \forall j = 0, 1, 2, \dots, m.$
 3. $u(x_i, y_0) = g(x_i, y_0), \quad \forall i = 1, 2, \dots, n-1.$
 4. $u(x_i, y_m) = g(x_i, y_m), \quad \forall i = 1, 2, \dots, n-1.$
- (1.7)

Now, by rearranging Eq. (1.6), we get:

$$\begin{aligned} & \frac{-2u(x_i, y_j)}{h^2} + \frac{-2u(x_i, y_j)}{k^2} + \frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) + u(x_i, y_{j-1}))}{k^2} \\ & = \frac{h^2}{12} \cdot \frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12} \cdot \frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) + G(x_i, y_j) \end{aligned}$$

or it can simply be written as

$$2\left[\frac{-1}{k^2} - \frac{1}{h^2}\right]u(x_i, y_j) + \frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) + u(x_i, y_{j-1}))}{k^2}$$

$$= \frac{h^2}{12} \cdot \frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12} \cdot \frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) + G(x_i, y_j)$$

Multiplying both sides by $-h^2$, we get:

$$\begin{aligned} & 2 \left[\left(\frac{h}{k} \right)^2 + 1 \right] u(x_i, y_j) - [u(x_{i+1}, y_j) + u(x_{i-1}, y_j)] - \left(\frac{h}{k} \right)^2 [u(x_i, y_{j+1}) + \\ & u(x_i, y_{j-1})] \\ & = -h^2 \left[\frac{h^2}{12} \cdot \frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12} \cdot \frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) \right] - h^2 G(x_i, y_j) \end{aligned}$$

In difference-equation form, this results in the central-difference method with local truncation error $O(h^2 + k^2)$.

Simplifying the last equation and letting $u_{i,j}$ approximate $u(x_i, y_j)$, we get:

$$\begin{aligned} & 2 \left[\left(\frac{h}{k} \right)^2 + 1 \right] u_{i,j} - [u_{i+1,j} + u_{i-1,j}] - \left(\frac{h}{k} \right)^2 [u_{i,j+1} + u_{i,j-1}] \\ & = -h^2 G(x_i, y_j) \quad (1.8) \end{aligned}$$

for each $i = 1, 2, \dots, n-1$ and $j = 1, 2, \dots, m-1$.

with boundary conditions:

$$\left. \begin{aligned} & i) u_{0,j} = g(x_0, y_j), \quad \forall j = 0, 1, 2, \dots, m. \\ & ii) u_{n,j} = g(x_n, y_j), \quad \forall j = 0, 1, 2, \dots, m. \\ & iii) u_{i,0} = g(x_i, y_0), \quad \forall i = 1, 2, \dots, n-1. \\ & iv) u_{i,m} = g(x_i, y_m), \quad \forall i = 1, 2, \dots, n-1. \end{aligned} \right\} \quad (1.9)$$

where $u_{i,j}$ approximates $u(x_i, y_j)$.

For more details, see references [2], [3] and [8].

Eq. (1.8) involves approximations to the unknown function $u(x,y)$ at the points

$$(x_i, y_j), (x_{i+1}, y_j), (x_{i-1}, y_j), (x_i, y_{j+1}), \text{ and } (x_i, y_{j-1})$$

These points form a star–shape region in the grid (as figure 1.2 shows) which shows that any equation involves approximations about (x_i, y_j) .

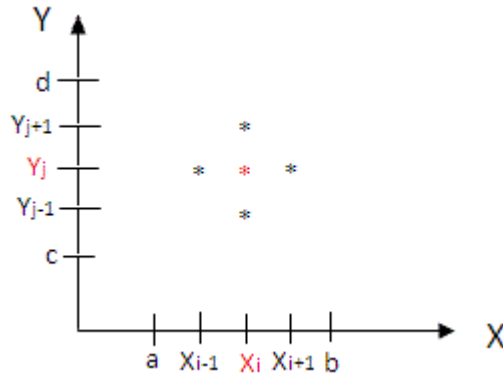


Figure 1.2

When we use formula (1.8) with boundary conditions (1.9), then at all points (x_i, y_j) that are adjacent to a boundary mesh point, we have an $(n - 1) \times (m - 1)$ by $(n - 1) \times (m - 1)$ linear system with the unknowns being the approximations $w_{i,j}$ to $u(x_i, y_j)$ at the interior mesh points.

The generated linear system should be solved by Jacobi, Gauss-Seidel, Successive over Relaxation (SOR), or Conjugate Gradient methods. This system (that involves the unknowns) produces satisfactory results if a relabeling of the interior mesh points is introduced. A favorable labeling of these points is [3] , [8] and [20]:

$$L_r = (x_i, y_j) \text{ and } u_r = u_{i,j}$$

$$\text{where } r = i + (m - 1 - j)(n - 1),$$

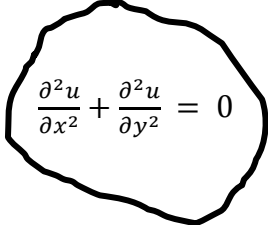
$$\forall i = 1, 2, \dots, n - 1 \text{ and } \forall j = 1, 2, \dots, m - 1.$$

1.4 Elliptic PDE subject to Boundary Conditions:

Solution of Laplace equation and Poisson equation on the boundary of a domain R needs certain conditions where the unknown function (dependent variable) must satisfy these conditions on the boundary S . We will deal with Laplace equation and Poisson equation with respect to two types of boundary conditions. These are Dirichlet and Neumann boundary conditions.

1.4.1 Laplace equation with Dirichlet Boundary Conditions:

When the function is defined on any part of a domain R , then we call this part Dirichlet boundary S_D , i.e. the unknown function u is prescribed on the boundary, that is, $u(x, y) = g(x, y), (x, y) \in S$ where the function g is a known function.

$$S_D : u = g \quad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$


To derive the formula of finite difference approximation with Dirichlet boundary condition for Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad (1.10)$$

We consider three points $i+1$, i , and $i-1$ which are located on X-axis with equal distance h between them (as figure 1.3 shows).

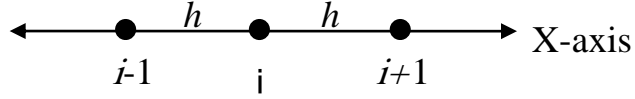


Figure 1.3

Let the value of the function $u(x,y)$ at the points $(i-1,j)$, (i,j) , and $(i+1,j)$ be $u_{i-1,j}$, $u_{i,j}$, and $u_{i+1,j}$, respectively.

Now, use Taylor series to express $u_{i+1,j}$ and $u_{i-1,j}$ in the form of Taylor expansions about the point i as follows:

$$u_{i+1,j} = u_{i,j} + \frac{h}{1!} \cdot \frac{\partial u}{\partial x} \Big|_i + \frac{h^2}{2!} \cdot \frac{\partial^2 u}{\partial x^2} \Big|_i + \frac{h^3}{3!} \cdot \frac{\partial^3 u}{\partial x^3} \Big|_i + \frac{h^4}{4!} \cdot \frac{\partial^4 u}{\partial x^4} \Big|_i + O(h^5) \quad (1.11)$$

$$u_{i-1,j} = u_{i,j} - \frac{h}{1!} \cdot \frac{\partial u}{\partial x} \Big|_i + \frac{h^2}{2!} \cdot \frac{\partial^2 u}{\partial x^2} \Big|_i - \frac{h^3}{3!} \cdot \frac{\partial^3 u}{\partial x^3} \Big|_i + \frac{h^4}{4!} \cdot \frac{\partial^4 u}{\partial x^4} \Big|_i + O(h^5) \quad (1.12)$$

By adding Eq. (1.11) and Eq. (1.12), we get:

$$u_{i+1,j} + u_{i-1,j} = 2u_{i,j} + h^2 \frac{\partial^2 u}{\partial x^2} \Big|_i + \frac{h^4}{12} \cdot \frac{\partial^4 u}{\partial x^4} \Big|_i + O(h^5)$$

By rearranging the above equation, we get:

$$\frac{\partial^2 u}{\partial x^2} \Big|_i = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2) \quad (1.13)$$

Eq.(1.13) is a finite difference approximation formula with error term $O(h^2)$ of second order for $\frac{\partial^2 u}{\partial x^2} \Big|_i$.

Now, subtracting Eq. (1.12) from Eq. (1.11), we get:

$$u_{i+1,j} - u_{i-1,j} = \frac{\partial u}{\partial x} \Big|_i + \frac{h^3}{3} \cdot \frac{\partial^3 u}{\partial x^3} \Big|_i + O(h^5)$$

By rearranging the above equation, we get:

$$\frac{\partial u}{\partial x} \Big|_i = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + O(h^2) \quad (1.14)$$

Eq.(1.14) is a finite difference approximation formula with error term $O(h^2)$ of second order for $\frac{\partial u}{\partial x}|_i$.

Similarly, consider three points $j+1$, j , and $j-1$ which are located on the Y-axis with equal distance h between them (as figure 1.4 shows).

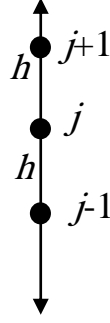


Figure 1.4

Let the value of the function $u(x,y)$ at the points $(i,j+1)$, (i,j) , and $(i,j-1)$ be $u_{i,j+1}$, $u_{i,j}$, and $u_{i,j-1}$, respectively. Using Taylor series to express $u_{i,j+1}$ and $u_{i,j-1}$ in the form of Taylor expansions at the point j , the finite difference approximation formulas with error term $O(h^2)$ of second order for $\frac{\partial^2 u}{\partial y^2}|_j$ and $\frac{\partial u}{\partial y}|_j$ are, respectively:

$$\frac{\partial^2 u}{\partial y^2}|_j = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2) \quad (1.15)$$

and

$$\frac{\partial u}{\partial y}|_j = \frac{u_{i,j+1} - u_{i,j-1}}{2h} + O(h^2) \quad (1.16)$$

Now, by combining figure 1.3 and figure 1.4 together, we get the star-shape (or 5-points stencil) region about the point (i,j) as shown in figure 1.5 [4].

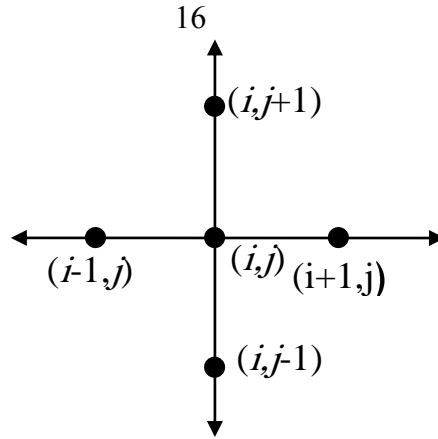


Figure 1.5

Inserting Eq. (1.13) and Eq. (1.15) into Eq. (1.10) yields:

$$\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) |_{(i,j)} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0$$

By rearranging the above equation, we get [12]:

$$(u_{i+1,j} + u_{i,j+1}) - 4u_{i,j} + (u_{i-1,j} + u_{i,j-1}) = 0$$

So,

$$u_{i,j} = \frac{1}{4} [u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1}] \quad (1.17)$$

In general, if u satisfies Laplace equation, then u , at any point in the domain R , is the average of the values of u at the four surrounding points in the 5-point stencil as shown in figure 1.2, page 12.

Now, suppose we have Dirichlet boundary conditions defined on the rectangular domain such that $1 \leq i \leq m$ and $1 \leq j \leq n$ as shown in figure 1.6 [4].

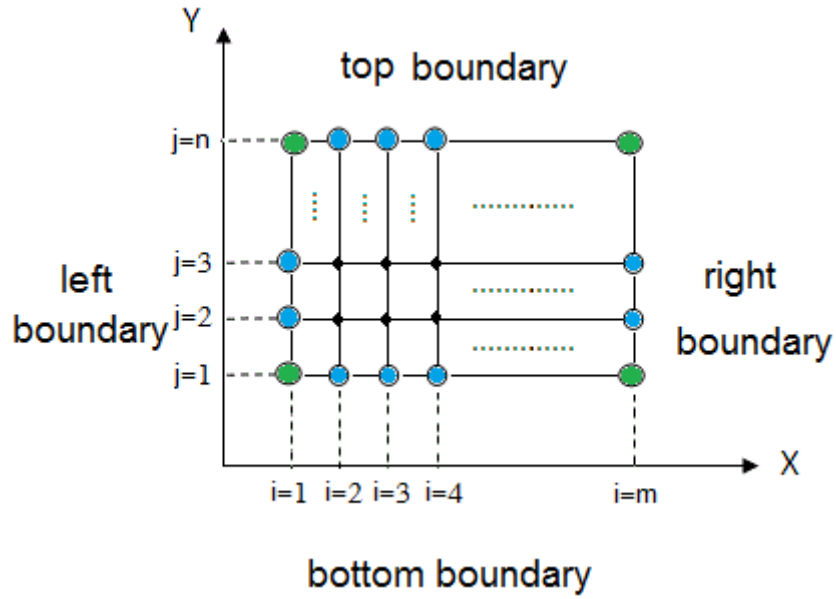


Figure 1.6

Let $u(x,y) = g(x,y)$ be given on all boundaries of the domain, that is $u = g$ is defined on the left, top, right, and bottom boundary walls so that the boundary grid points (blue points) and the corner grid points (green points) are known [3].

In other words, the values of the points $(x_i, y_j), \forall i = 2, 3, \dots, m - 1$ and $\forall j = 2, 3, \dots, n - 1$ under the function g are known. For the corner grid points, we use the following equations:

$$u(1,1) = \frac{1}{2} [u(2,1) + u(1,2)]$$

$$u(m,1) = \frac{1}{2} [u(m-1,1) + u(m,2)] \quad (1.18)$$

$$u(1,n) = \frac{1}{2} [u(1,n-1) + u(2,n)]$$

$$u(m,n) = \frac{1}{2} [u(m,n-1) + u(m-1,n)]$$

[4] and [17].

1.4.2 Poisson Equation with Dirichlet Boundary Conditions:

To derive the formula of finite difference approximation with Dirichlet boundary condition for Poisson equation:

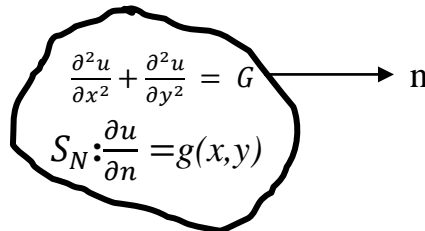
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x, y) \quad (1.19)$$

Following similar approach for Laplace equation with some amendments in Eq. (1.17), that is [12]:

$$u_{i,j} = \frac{1}{4} [u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1}] - \frac{h^2}{4} G_{i,j} \quad (1.20)$$

1.4.3 Laplace Equation with Neumann Boundary Conditions:

When the normal derivative of the unknown function u is prescribed on the boundary of a domain R , then we call this part Neumann boundary S_N , i.e. the value of the normal derivative $\frac{\partial u}{\partial n} = g(x, y)$ is given on the boundary of the domain, where $g(x, y)$ is a given function.



$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= G \\ S_N: \frac{\partial u}{\partial n} &= g(x, y) \end{aligned}$$

To derive the formula of finite difference approximation with Neumann boundary condition for Laplace equation

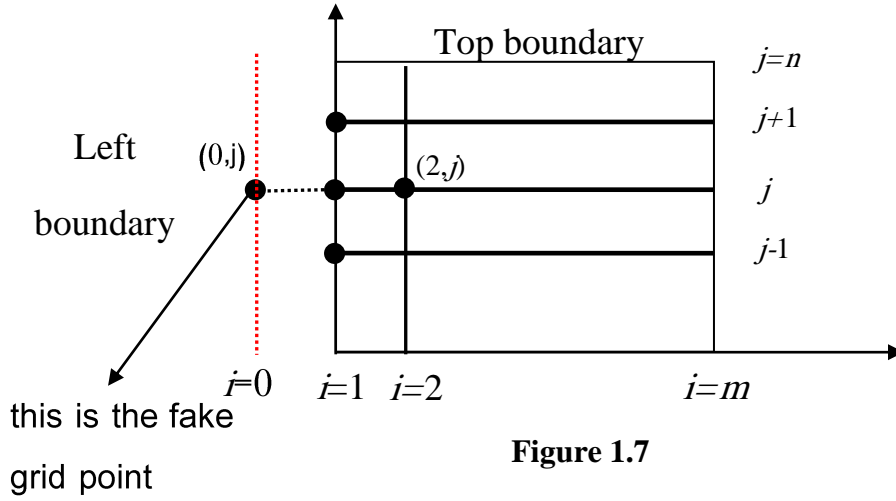
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Consider that we have a rectangle domain as shown in figure 1.7.

Suppose that Dirichlet condition is specified on top, right, and bottom walls and Neumann condition is defined on the remaining wall which is the left wall as follows:

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} = -g(y) \quad (1.21)$$

Now, we want to approximate Eq. (1.21) using the second order approximation using Eq. (1.14). This procedure puts the grid points $(1,j)$ outside the domain towards the left that is located on imaginary boundary (red line) that their fake coordinates will be $(0,j)$ [4], [8].



So, Eq. (1.21) is approximated using Eq. (1.14) at the line $i = 1$

$$\left. \frac{\partial u}{\partial x} \right|_{(1,j)} = \frac{u_{1+1,j} - u_{1-1,j}}{2h} = \frac{u(2,j) - u(0,j)}{2h} = -g(1,j)$$

Thus,

$$u(0,j) = u(2,j) + 2h g(1,j) \quad (1.22)$$

Now, we write Eq. (1.17) at the point $(1,j)$ as

$$\begin{aligned} u_{1,j} &= \frac{1}{4} [u_{1+1,j} + u_{1,j+1} + u_{1-1,j} + u_{1,j-1}] \\ u(1,j) &= \frac{1}{4} [u_{2,j} + u_{1,j+1} + u_{0,j} + u_{1,j-1}] \\ &= \frac{1}{4} [u(2,j) + u(1,j+1) + u(0,j) + u(1,j-1)] \end{aligned}$$

By substituting Eq. (1.22) in the previous formula, we get:

$$u(1,j) = \frac{1}{4} [u(2,j) + u(1,j+1) + u(2,j) + 2h g(1,j) + u(1,j-1)]$$

$$u(1,j) = \frac{1}{4} [2u(2,j) + 2h g(1,j) + u(1,j+1) + u(1,j-1)] \quad (1.23)$$

For any two positive integers m and n , we use Eq. (1.23) for $2 \leq j \leq n-1$, where $g(1,j)$ is a specified function. As Dirichlet condition is specified on north, east, and south walls, the values

$\{u(i,n), 2 \leq i \leq m-1\}$, $\{u(m,j), 2 \leq j \leq n-1\}$, and $\{u(i,1), 2 \leq i \leq m-1\}$ are known.

To find the values of corner grid points, we use Eq. (1.18).

1.4.4 Poisson equation with Neumann Boundary Conditions:

Consider the Poisson equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x, y)$$

with Neumann boundary condition:

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} = -g(y)$$

defined on the rectangular domain.

Similar to Laplace equation, the difference approximation formula of Neumann condition at the fake (ghost) grid point $(0,j)$ is Eq. (1.22), that is:

$$u(0,j) = u(2,j) + 2h g(1,j)$$

Now, using Eq. (1.20) to find the value of the point $(1,j)$, we get:

$$u_{1,j} = \frac{1}{4} [u_{2,j} + u_{1,j+1} + u_{0,j} + u_{1,j-1}] - \frac{h^2}{4} G_{i,j} \quad (1.25)$$

By substitute Eq. (1.22) into Eq. (1.25) with $u(0,j) = u_{0,j}$, we get:

$$u_{1,j} = \frac{1}{4} [u_{2,j} + u_{1,j+1} + (u(2,j) + 2h g(1,j)) + u_{1,j-1}] - \frac{h^2}{4} G_{1,j}$$

So

$$u_{1,j} = \frac{1}{4} [u_{2,j} + u_{1,j+1} + u_{2,j} + 2h g_{1,j} + u_{1,j-1}] - \frac{h^2}{4} G_{1,j}$$

$$u_{1,j} = \frac{1}{4} [2 u_{2,j} + u_{1,j+1} + 2h g_{1,j} + u_{1,j-1}] - \frac{h^2}{4} G_{1,j} \quad (1.26)$$

If $i \neq 1$, we use Eq. (1.20).

Using the same method, we can deal with other boundary points except the corner points. For corner points, we use Eq. (1.18) to find their values.

[4] and [8].

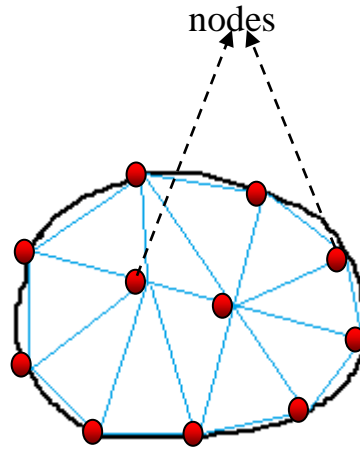
1.5 Finite Element Method:

The Finite Element Method (FEM) is the most known numerical method used for approximating the solution of partial differential equations on domains with irregular shapes.

1.6 The Principle of Finite Element Method:

The idea of the FEM is to partition the region (domain) to finite number of elements (parts) of regular shapes that are either triangles or rectangles (as figure 1.8 shows). These elements describe the behavior of the domain. A node is a vertex where two or more elements are intersected (Red points as in figure 1.8). The FEM can be applied on many scientific and engineering

problems such as fluid flow, heat transfer, electromagnetic fields, aerospace, civil engineering, and so on.



Two dimensional irregular region divided
into triangular elements

Figure 1.8

1.6.1 Finite Element Method for Dirichlet boundary value problems:

This section discusses the finite element method that is used to solve two dimensional elliptic partial differential equations with Dirichlet boundary conditions in a rectangular domain and focuses on finite element solution using spreadsheets with triangular grid.

Now, we want to approximate the solution of Laplace equation

$$u_{xx} + u_{yy} = 0$$

defined on a rectangular domain with Dirichlet boundary conditions defined on the top, left, right and bottom boundaries (edges) as shown in figure 1.9.

Divide the interval $0 < x < a$ into m equal subinterval on bottom boundary. Also, divide the interval $0 < y < b$ into n equal subinterval on left boundary. The interior nodes (points) are unknown whereas the boundary nodes are.

For example, divide the region into 40 equal triangular elements.

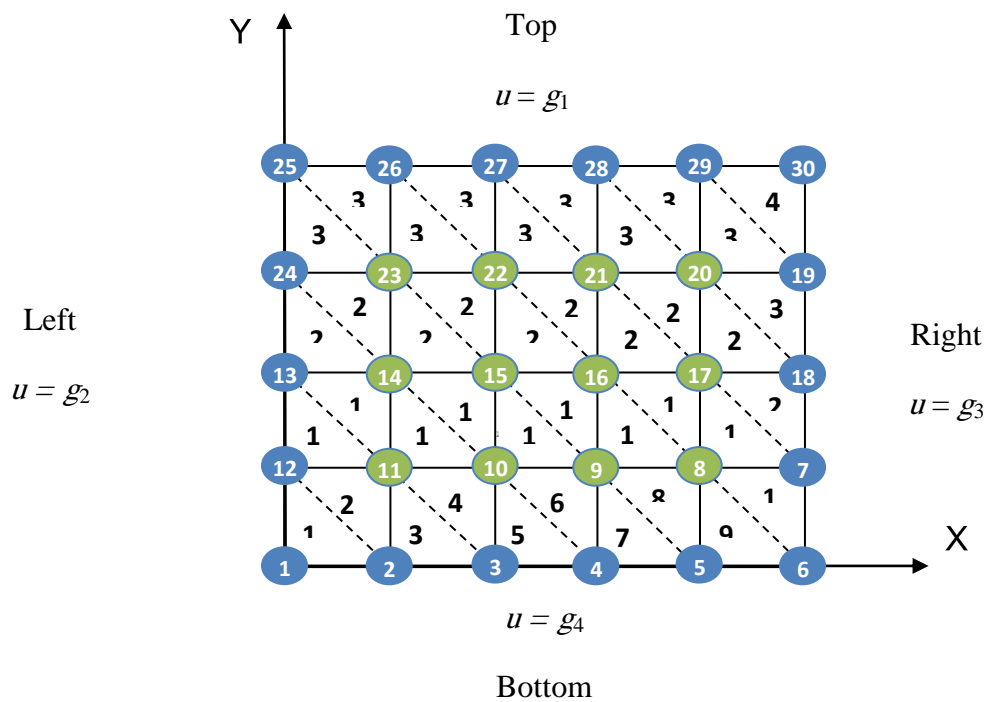


Figure 1.9

In this discretization, there are 30 global nodes. The blue nodes are known since they are located on the boundaries that the function u is defined on them but the green nodes (interior nodes) are not.

In this case, $m = 5$ portions (from node 1 to node 2, from node 2 to node 3, from node 3 to node 4, from node 4 to node 5, and from node 5 to node 6).

The length of each subinterval is equal to $\frac{a}{m} = \frac{a}{5}$.

$n = 4$ (from node 1 to node 12, from node 12 to node 13, from node 13 to node 24, and from node 24 to node 25). The length of each subinterval is equal to $\frac{b}{n} = \frac{b}{4}$.

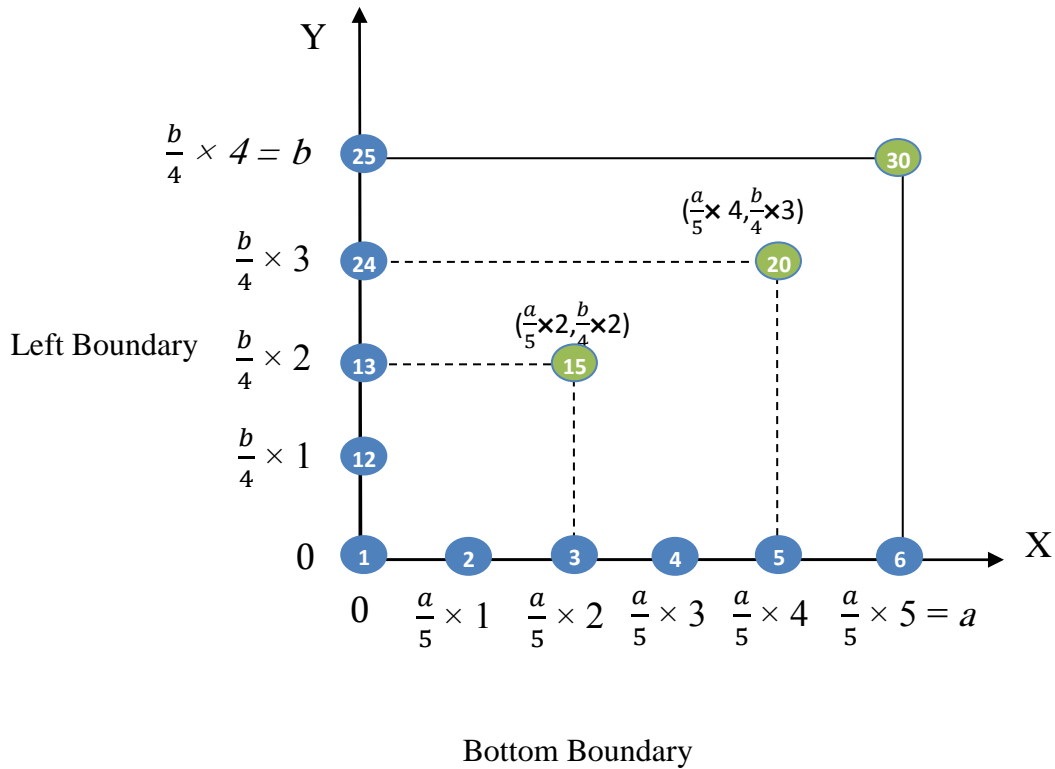


Figure 1.10

Now, we can easily find the coordinates for each node as shown in figure 1.10 as follows [11]:

Node 1: $(0,0)$

Node 15: $(\frac{a}{5} \times 2, \frac{b}{4} \times 2)$

Node 20: $(\frac{a}{5} \times 4, \frac{b}{4} \times 3)$

Node 30: (a,b)

In the same manner for the remaining nodes.

Now, for each element (triangle) e , we determine the local node numbers 1, 2, and 3 that must be assigned so that global nodes associated with an element are traversed in a counterclockwise sense.

For element 1:

At node 1: the local node number is 1, so $(x_1, y_1) = (0, 0)$

At node 2: the local node number is 2, so $(x_2, y_2) = (\frac{a}{5}, 0)$

At node 12: the local node number is 3, so $(x_3, y_3) = (0, \frac{b}{4})$

These are shown in figure 1.11.

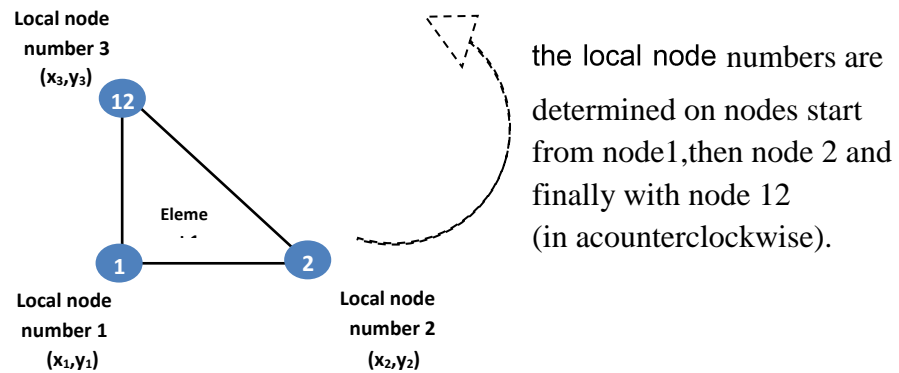


Figure 1.11

Similarly, we determine the local node numbers 1, 2 and 3 for each element e in the same way as in element 1.

The following must be computed for each element e :

$$P_1 = y_2 - y_3 \qquad Q_1 = x_3 - x_2$$

$$P_2 = y_3 - y_1 \qquad Q_2 = x_1 - x_3$$

$$P_3 = y_1 - y_2 \qquad Q_3 = x_2 - x_1$$

For element 1:

$$(x_1, y_1) = (0, 0), (x_2, y_2) = \left(\frac{a}{5}, 0\right), (x_3, y_3) = \left(0, \frac{b}{4}\right)$$

$$P_1 = y_2 - y_3 = 0 - \frac{b}{4} = -\frac{b}{4}$$

$$Q_1 = x_3 - x_2 = 0 - \frac{a}{5} = -\frac{a}{5}$$

The same thing for other elements.

Now, for each element e , we want to find the 3×3 element coefficient matrix for which the entries are given by the equation [10]:

$$C_{ij}^{(e)} = \frac{1}{4A} [P_i P_j + Q_i Q_j], \quad \text{for } i, j = 1, 2, 3. \quad (1.27)$$

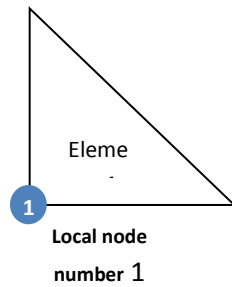
where:

$$A = \frac{1}{2} [P_2 Q_3 - P_3 Q_2]$$

When we find the element coefficient matrices, then the global coefficient matrix C is assembled from the element coefficient matrices. If the number of nodes is N , then the global coefficient matrix C will be an $N \times N$ matrix (in our case, $N = 30$).

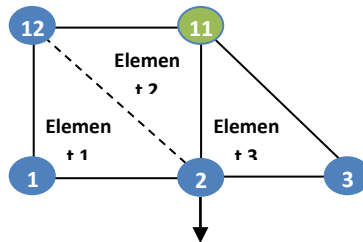
We can compute the entries of main diagonal as follows:

$C_{1,1}$: is the entry that is located on row 1 and column 1 in the global coefficient matrix C which corresponds to node 1 that belongs to element 1 only. Node 1 is assigned local node number 1 in element 1 as shown in the following figure.



$C_{1,1} = C_{11}^{(1)}$, where $C_{11}^{(1)}$ is the entry that is located on row 1 and column 1 in the element coefficient matrix for element 1.

$C_{2,2}$: is the entry that is located on row 2 and column 2 in the global coefficient matrix C which corresponds to node 2 that belongs to elements 1, 2, and 3. Node 2 is assigned local node number 2 in element 1 and local node number 1 in elements 2 and 3 as shown in the following figure.



Node 2 has Local node number 2 in element 1 and Local node number 1 in elements 2 and 3.

Figure 1.12

$C_{2,2} = C_{22}^{(1)} + C_{11}^{(2)} + C_{11}^{(3)}$, where $C_{22}^{(1)}$ is the entry that is located on row 2 and column 2 in the element coefficient matrix for element 1 and $C_{11}^{(2)}$, $C_{11}^{(3)}$ are the entries that are located on row 1 and column 1 in the element coefficient matrix for element 2 and element 3, respectively.

Using the same method, we can find the remaining diagonal entries C_{ii} , for $i = 1, \dots, N$. For other entries in the global coefficient matrix C , we do that using a different method.

For more details, see reference [11].

Take, for example, the entry $C_{2,11}$ in the global coefficient matrix C . It corresponds to node 2 and node 11. So, the link between node 2 and node 11 is called global link which corresponds to local link 1–2 of element 2 and local link 1–3 of element 3 as shown in figure 1.12. Hence,

$$C_{2,11} = C_{12}^{(2)} + C_{13}^{(3)}.$$

The other off-diagonal entries are treated similarly.

Now, defining vector u_v to be a vector of unknowns (interior nodes, green nodes) and vector u_n to be a vector of prescribed boundary values. In other words, u_n is a vector of the value of nodes that are located on the boundaries (blue nodes) as shown in figure 1.9.

Define matrix C_{vv} to be a matrix of unknown nodes obtained from the global coefficient matrix C and matrix C_{vn} to be a matrix of unknown nodes and prescribed boundary values that is also obtained from the global coefficient matrix C .

In our case, C_{vv} is a 12×12 matrix since we have 12 interior nodes (green nodes) and C_{vn} is a 12×18 matrix since we have 12 interior nodes (green nodes) and 18 boundary nodes (blue nodes) as shown in figure 1.9.

The vector u_v of unknown nodes can be computed by using:

$$u_v = -C_{vv}^{-1}C_{vn}u_n \quad (1.28)$$

The vector u_v contains the approximations to the unknown nodes (interior nodes) [11].

1.6.2 Finite Element Method with Neumann Boundary condition:

We consider a stationary problem in two dimensions:

$$- \Delta u = f \text{ in } \Omega, \quad (1.29)$$

$$u = 0 \text{ on } \Gamma,$$

Where Ω is a bounded domain in the plane with boundary Γ , f is a given real-valued piecewise continuous bounded function in Ω .

Define the following subspace of a Sobolev space:

$C = \{\alpha(x,y) \mid \alpha \text{ is a continuous function on } \Omega, \alpha_x \text{ and } \alpha_y \text{ are piecewise continuous and bounded on } \Omega, \text{ and } \alpha = 0 \text{ on } \Gamma\}$.

Now, let $B = (b_1, b_2)$ be a vector-valued function. By applying Green's theorem and the divergence theorem, we get:

$$\int_{\Gamma} B \cdot n \, d\zeta = \int_{\Omega} \nabla \cdot B \, dx \quad (1.30)$$

Where:

$n = \langle n_1, n_2 \rangle$ is the outward unit normal to Γ .

$B \cdot n$ is the dot product between B and n .

$\nabla \cdot B$ is the divergence operator of B .

Take $\alpha, \beta \in C$ with B defined as:

$$B = \left(\frac{\partial \alpha}{\partial x} \beta, 0 \right) \text{ and } B = \left(0, \frac{\partial \alpha}{\partial y} \beta \right) \quad (1.31)$$

Inserting Eq. (1.31) into Eq. (1.30), respectively, then we get:

$$\int_{\Gamma} \frac{\partial \alpha}{\partial x} \beta n_1 d\zeta = \int_{\Omega} \frac{\partial^2 \alpha}{\partial x^2} \beta dx + \int_{\Omega} \frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x} dx \quad (1.32)$$

and

$$\int_{\Gamma} \frac{\partial \alpha}{\partial y} \beta n_2 d\zeta = \int_{\Omega} \frac{\partial^2 \alpha}{\partial y^2} \beta dx + \int_{\Omega} \frac{\partial \alpha}{\partial y} \frac{\partial \beta}{\partial y} dx \quad (1.33)$$

Combining Eq. (1.32) and Eq. (1.33), we obtain [23]:

$$\begin{aligned} \int_{\Gamma} \frac{\partial \alpha}{\partial x} \beta n_1 d\zeta + \int_{\Gamma} \frac{\partial \alpha}{\partial y} \beta n_2 d\zeta &= \int_{\Omega} \frac{\partial^2 \alpha}{\partial x^2} \beta dx + \int_{\Omega} \frac{\partial^2 \alpha}{\partial y^2} \beta dx \\ &+ \int_{\Omega} \frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x} dx + \int_{\Omega} \frac{\partial \alpha}{\partial y} \frac{\partial \beta}{\partial y} dx \end{aligned} \quad (1.34)$$

$$\int_{\Gamma} \left(\frac{\partial \alpha}{\partial x} n_1 + \frac{\partial \alpha}{\partial y} n_2 \right) \beta d\zeta = \int_{\Omega} \left(\frac{\partial^2 \alpha}{\partial x^2} + \frac{\partial^2 \alpha}{\partial y^2} \right) \beta dx + \int_{\Omega} \left(\frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x} + \frac{\partial \alpha}{\partial y} \frac{\partial \beta}{\partial y} \right) dx$$

In virtue of $\frac{\partial \alpha}{\partial x} n_1 + \frac{\partial \alpha}{\partial y} n_2 = \frac{\partial \alpha}{\partial n}$, $\frac{\partial^2 \alpha}{\partial x^2} + \frac{\partial^2 \alpha}{\partial y^2} = \Delta \alpha$

and $\frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x} + \frac{\partial \alpha}{\partial y} \frac{\partial \beta}{\partial y} = \nabla \alpha \cdot \nabla \beta$

Eq. (1.34) becomes:

$$\int_{\Gamma} \frac{\partial \alpha}{\partial n} \beta d\zeta = \int_{\Omega} \Delta \alpha \beta dx + \int_{\Omega} \nabla \alpha \cdot \nabla \beta dx \quad (1.35)$$

Rearranging Eq. (1.35) yields:

$$\int_{\Omega} \Delta \alpha \beta dx = \int_{\Gamma} \frac{\partial \alpha}{\partial n} \beta d\zeta - \int_{\Omega} \nabla \alpha \cdot \nabla \beta dx \quad (1.36)$$

Define a bilinear form on $C \times C$ as follows:

$$\sigma(u, \alpha) = \int_{\Omega} \nabla u \cdot \nabla \alpha \, dx$$

and

$$(f, \alpha) = \int_{\Omega} f \alpha \, dx$$

Also, we define the functional $F: C \rightarrow \mathbb{R}$ by

$$F(\alpha) = \frac{1}{2} \sigma(u, \alpha) - (f, \alpha)$$

Now, multiply both sides of Eq. (1.29) by α and then integrating over Ω :

$$- \int_{\Omega} \Delta u \alpha \, dx = \int_{\Omega} f \alpha \, dx \quad (1.37)$$

Note that if we substitute the right hand side of Eq. (1.37) into Eq. (1.36)

with $\alpha = 0$ on the boundary Γ , then we get:

$$\int_{\Omega} \Delta u \alpha \, dx = 0 - \int_{\Omega} \nabla u \cdot \nabla \alpha \, dx$$

Eq. (1.37) becomes [23]:

$$\int_{\Omega} \nabla u \cdot \nabla \alpha \, dx = \int_{\Omega} f \alpha \, dx$$

Now, suppose the domain Ω is divided into finite number of elements

(triangles) $T_i, \forall i = 1, 2, \dots, m$ such that:

$$\bar{\Omega} = \cup_{i=1}^m \bar{T}_i$$

where $\bar{\Omega}$ is the domain and its boundary, i.e. $\bar{\Omega} = \Omega \cup \Gamma$. The same definition for the triangular elements T_i .

Let T_h be a partition of Ω . Take any triangle $T \in T_h$ where:

$\text{diam}(T)$ = the longest edge of T and $h = \max_{T \in T_h} \text{diam}(T)$.

Now, we can define the finite element space as follows:

$C_h = \{ \alpha(x,y) \mid \alpha \text{ is a continuous function on } \Omega \text{ and it is linear on each triangle } T \in T_h, \text{ and } \alpha = 0 \text{ on } \Gamma \}$.

Each triangle $T_i \forall i = 1, 2, \dots, m$ has three vertices denoted by v_1, v_2, v_3 . We define the basis function ϕ_i as follows:

$$\phi_i(v_j) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$$

$$\forall i = 1, 2, \dots, m \text{ and } j = 1, 2, 3.$$

Let v be a set of vertices where $\phi_i(v) \neq 0$ and let m be the number of interior vertices in T_h , any function $\alpha \in C_h$ has a unique representation written as:

$$\alpha(v) = \sum_{i=1}^m \alpha_i \phi_i(v), \quad v \in \Omega$$

Where $\alpha_i = \alpha(v_i)$.

A linear system is appeared in the matrix form written as:

$$A\mathbf{u} = \mathbf{f}$$

where:

$$A = (a_{ij}), \quad a_{ij} = \sigma(\phi_i, \phi_j)$$

$$\mathbf{u} = (u_j), \quad (\text{unknowns})$$

$$\mathbf{f} = (f_j), \quad f_j = (f, \phi_j)$$

$$\forall i, j = 1, 2, \dots, m.$$

[6],[13] and [18].

Chapter Two

Iterative Methods for Solving Linear systems

In chapter 1, linear systems were generated using the finite difference method and the finite element method to describe the partial differential equations that should be solved by iterative techniques. In this chapter, we will solve such linear systems by iterative methods and discuss the convergence for each of them and make a comparison between these iterative methods to conclude the best.

For solving an $n \times n$ linear system

$$A \mathbf{x} = \mathbf{b}$$

We start with an initial approximation $\mathbf{x}^{(0)}$ to the solution \mathbf{x} and then generate a sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ that converges to \mathbf{x} .

Most iterative techniques involve a process of converting the system $A \mathbf{x} = \mathbf{b}$ into an equivalent system:

$$\mathbf{x} = T \mathbf{x} + C$$

where :

1. T is an $n \times n$ iteration matrix.
2. C is a column vector of dimension n .

After selecting an initial approximation $\mathbf{x}^{(0)}$, we generate a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ defined as:

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + C, \quad k \geq 1$$

The iterative methods are:

1. Jacobi Method.
2. Gauss-Seidel Method.
3. Successive over Relaxation (SOR) Method.
4. Conjugate Gradient Method.

2.1 Jacobi Method

The Jacobi method is the simplest iterative method for solving a (square) linear system $A\mathbf{x} = \mathbf{b}$.

The General Formula of Jacobi Method

To derive a general formulation of this method, consider the $n \times n$ (square) linear system:

$$\left. \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \end{array} \right\} \quad (2.1)$$

where :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \text{and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

We can simply write this system in matrix form as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

Now, we start by converting (2.1) into the form [3]:

$$\mathbf{x} = T \mathbf{x} + C$$

that is:

$$x_1^{(k)} = -\frac{a_{12}}{a_{11}}x_2^{(k-1)} - \frac{a_{13}}{a_{11}}x_3^{(k-1)} - \cdots - \frac{a_{1n}}{a_{11}}x_n^{(k-1)} + \frac{b_1}{a_{11}}, a_{11} \neq 0$$

$$x_2^{(k)} = -\frac{a_{21}}{a_{22}}x_1^{(k-1)} - \frac{a_{23}}{a_{22}}x_3^{(k-1)} - \cdots - \frac{a_{2n}}{a_{22}}x_n^{(k-1)} + \frac{b_2}{a_{22}}, a_{22} \neq 0$$

⋮

$$x_n^{(k)} = -\frac{a_{n1}}{a_{nn}}x_1^{(k-1)} - \frac{a_{n2}}{a_{nn}}x_2^{(k-1)} - \cdots - \frac{a_{nn-1}}{a_{nn}}x_{n-1}^{(k-1)} + \frac{b_n}{a_{nn}}, a_{nn} \neq 0$$

By writing this system in matrix form, we get:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & -\frac{a_{nn-1}}{a_{nn}} & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix}$$

So,

$$\mathbf{x} = T \mathbf{x} + C$$

Given initial approximation $\mathbf{x}^{(0)}$, we generate the sequence of vectors

$\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ by computing:

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + C, \quad k \geq 1$$

In general, the Jacobi iterative method is given by the sequence:

$$\mathbf{x}_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{j=1}^n -a_{ij}\mathbf{x}_j^{(k-1)} + b_i \right], i \neq j, a_{ii} \neq 0, i = 1, 2, 3, \dots, n \quad (2.2)$$

$k \in \mathbb{N}^*$ [3] and [16].

We can derive formula (2.2) by splitting matrix A into its diagonal and off-diagonal parts.

Let D be the diagonal matrix where entries are those of matrix A , let $-L$ be the strictly lower triangular part of matrix A and let $-U$ be the strictly upper triangular part of matrix A . With this notation matrix A is split into:

$$A = D - L - U$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Then,

$$A\mathbf{x} = \mathbf{b}$$

By substituting $A = D - L - U$, we get:

$$(D - L - U)\mathbf{x} = \mathbf{b}$$

The above equation can be written as:

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b}$$

If D^{-1} exists, then:

$$\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b}$$

This result is the matrix form of the Jacobi scheme:

$$\mathbf{x}^{(k)} = D^{-1}(L + U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}$$

Using $T_j = D^{-1}(L + U)$ and $C = D^{-1}\mathbf{b}$, we obtain the Jacobi technique of the form:

$$\mathbf{x}^{(k)} = T_j\mathbf{x}^{(k-1)} + C, \quad k \geq 1$$

So,

$$\mathbf{x}_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{j=1}^n -a_{ij}\mathbf{x}_j^{(k-1)} + b_i \right], i \neq j, a_{ii} \neq 0, i = 1, 2, 3, \dots, n.$$

Conclusion: to find $\mathbf{x}^{(k)}$ approximation we must know $\mathbf{x}^{(k-1)}$ approximation for any $k \geq 1$ where $k \in \mathbb{N}$. Continuing this procedure, we obtain a sequence of approximations [3] and [15].

2.2 Gauss-Seidel Method

This iterative method is used for solving a (square) linear system $A\mathbf{x} = \mathbf{b}$.

The General Formula of Gauss-Seidel Method

Consider the $n \times n$ (square) linear system:

$$\left. \begin{array}{l} \text{Eq. 1 : } a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ \text{Eq. 2 : } a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ \text{Eq. n : } a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \end{array} \right\}$$

Where:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}, \text{ and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

We can simply write this system in matrix form as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$x_1^{(k)} = -\frac{a_{12}}{a_{11}}x_2^{(k-1)} - \frac{a_{13}}{a_{11}}x_3^{(k-1)} - \cdots - \frac{a_{1n}}{a_{11}}x_n^{(k-1)} + \frac{b_1}{a_{11}}, \quad a_{11} \neq 0$$

$$x_2^{(k)} = -\frac{a_{21}}{a_{22}}x_1^{(k)} - \frac{a_{23}}{a_{22}}x_3^{(k-1)} - \cdots - \frac{a_{2n}}{a_{22}}x_n^{(k-1)} + \frac{b_2}{a_{22}}, \quad a_{22} \neq 0$$

⋮

$$x_n^{(k)} = -\frac{a_{n1}}{a_{nn}}x_1^{(k)} - \frac{a_{n2}}{a_{nn}}x_2^{(k)} - \cdots - \frac{a_{nn-1}}{a_{nn}}x_{n-1}^{(k)} + \frac{b_n}{a_{nn}}, \quad a_{nn} \neq 0$$

Given initial approximation $\mathbf{x}^{(0)}$, we generate the sequence of vectors

$\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ by computing:

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + C, \quad k \geq 1$$

In general, the Gauss-Seidel iterative method is given by the sequence :

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[- \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} + b_i \right], a_{ii} \neq 0, \quad (2.3)$$

$$i = 1, 2, 3, \dots, n.$$

We can derive formula (2.3) by splitting matrix A into its diagonal and off-diagonal parts [3].

Let D be the diagonal matrix where entries are those of matrix A , let $-L$ be the strictly lower triangular part of matrix A , and let $-U$ be the strictly upper triangular part of matrix A . With this notation matrix A is split into:

$$A = D - L - U$$

Then,

$$A \mathbf{x} = \mathbf{b}$$

By substituting $A = D - L - U$, we get:

$$(D - L - U)\mathbf{x} = \mathbf{b}$$

The above equation can be written as:

$$(D - L)\mathbf{x} = U\mathbf{x} + \mathbf{b}$$

If $(D - L)^{-1}$ exists, then:

$$\mathbf{x} = (D - L)^{-1} U\mathbf{x} + (D - L)^{-1} \mathbf{b}$$

This result is the matrix form of the Gauss-Seidel iterative method:

$$\mathbf{x}^{(k)} = (D - L)^{-1} U \mathbf{x}^{(k-1)} + (D - L)^{-1} \mathbf{b}$$

Using $T_g = (D - L)^{-1} U$ and $C = (D - L)^{-1} \mathbf{b}$, we obtain the Gauss-Seidel iterative method of the form:

$$\mathbf{x}^{(k)} = T_g \mathbf{x}^{(k-1)} + C, \quad k \geq 1$$

see references [3],[5] and [16].

2.3 Successive over Relaxation Method (SOR Method)

The main constraint to using this method is that the coefficient matrix A of the linear system $A \mathbf{x} = \mathbf{b}$ must be symmetric and positive definite. For any positive real number called the relaxation parameter (factor) $\omega \in (0,2)$, when $0 < \omega < 1$, the method is called Successive under Relaxation and can be used to achieve convergence for systems that are not convergent by the Gauss-Seidel method. However, if $1 < \omega < 2$, then the method is called Successive over Relaxation method and can be used to accelerate convergence of linear systems that are already convergent by the Gauss-Seidel method. If $\omega = 1$, then we get Gauss-Seidel method [10].

The General Formula of SOR Method

The derivation of the general formula of SOR method depends on Gauss-Seidel formula. Consider Gauss-Seidel formula that is (2.3):

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[- \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} + b_i \right], a_{ii} \neq 0$$

Defining the difference:

$$\Delta x_i = x_i^{(k)} - x_i^{(k-1)}$$

This can be written as:

$$x_i^{(k)} = x_i^{(k-1)} + \Delta x_i$$

Now, multiplying the relaxation parameter ω by Δx_i in the last expression, we get:

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)} + \omega \Delta x_i \\ &= x_i^{(k-1)} + \omega(x_i^{(k)} - x_i^{(k-1)}) \\ &= (1 - \omega) x_i^{(k-1)} + \omega x_i^{(k)} \end{aligned}$$

Substituting the Gauss-Seidel formula (2.3) into the last expression, we get:

$$x_i^{(k)} = (1 - \omega) x_i^{(k-1)} + \omega \frac{1}{a_{ii}} \left[- \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} + b_i \right] \quad (2.4)$$

$$a_{ii} \neq 0, \quad i = 1, 2, 3, \dots, n.$$

Formula (2.4) is called the SOR iterative method [3].

We can write Eq. (2.4) in matrix form as follows:

Since $a_{ii} \neq 0$, then we can multiply Eq. (2.4) by a_{ii} to get:

$$a_{ii} x_i^{(k)} = a_{ii} (1 - \omega) x_i^{(k-1)} + \omega \left[- \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} + b_i \right]$$

Simplifying the last equation, we get

$$a_{ii} x_i^{(k)} = (1 - \omega) a_{ii} x_i^{(k-1)} - \omega \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \omega \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} + \omega b$$

By rearranging the above equation, we get

$$a_{ii}x_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} = (1 - \omega)a_{ii}x_i^{(k-1)} - \omega \sum_{j=i+1}^n a_{ij}x_j^{(k-1)} + \omega \mathbf{b}$$

So,

$$(D - \omega L)\mathbf{x}^{(k)} = ((1 - \omega)D + \omega U)\mathbf{x}^{(k-1)} + \omega \mathbf{b}$$

Now, if $(D - \omega L)^{-1}$ exists, then we have:

$$\mathbf{x}^{(k)} = (D - \omega L)^{-1} ((1 - \omega)D + \omega U)\mathbf{x}^{(k-1)} + \omega (D - \omega L)^{-1} \mathbf{b}$$

Then, we get the matrix form of SOR method as:

$$\mathbf{x}^{(k)} = T_\omega \mathbf{x}^{(k-1)} + C_\omega,$$

Where:

$$T_\omega = (D - \omega L)^{-1} ((1 - \omega)D + \omega U) \text{ and } C_\omega = \omega (D - \omega L)^{-1} \mathbf{b}$$

[3] and [16].

2.4 Conjugate Gradient Method

The conjugate gradient method is a numerical iterative method used to approximate the exact solution of particular linear system $A \mathbf{x} = \mathbf{b}$ where the coefficient matrix A must be symmetric and positive definite.

General Formulas Needed to Compute Conjugate Gradient Method Algorithm

Suppose we want to solve the following $n \times n$ linear system:

$$A \mathbf{x} = \mathbf{b}$$

Where A is symmetric and positive definite matrix, \mathbf{x} and \mathbf{b} are column vectors ($n \times 1$ –matrices).

The solution of $A\mathbf{x} = \mathbf{b}$ uniquely minimizes the following quadratic form:

$$f(x) = \frac{1}{2}x^tAx - \mathbf{b}^tx$$

Suppose that p is a basis of \mathbb{R}^n where:

$p = \{\mathbf{p}_k \mid \mathbf{p}_i \cdot \mathbf{p}_k = 0 \text{ with respect to the matrix } A, \forall i \neq k \text{ where } 1 \leq i, k \leq n\}$ is a set of n mutually conjugate (orthogonal) directions [14].

We will write the conjugate gradient iterative method algorithm as follows:

Step 1:

Start with initial guess \mathbf{x}_0 that may be considered $\mathbf{0}$ if otherwise is given.

Step 2:

Calculate the residual vector \mathbf{r}_0 as follows:

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$$

Step 3:

Let the initial direction vector $\mathbf{p}_0 = \mathbf{r}_0$, that is, the negative of the gradient of the quadratic function:

$$f(x) = \frac{1}{2}x^tAx - \mathbf{b}^tx$$

at $x = x_0$.

Note that \mathbf{p}_k will change in each iteration.

Step 4:

Compute the scalars α_k 's using the formula:

$$\alpha_k = \frac{\mathbf{r}_k^t \mathbf{r}_k}{\mathbf{p}_k^t A \mathbf{p}_k}, \quad \forall k = 0, 1, 2, \dots, n-1.$$

Step 5:

Compute the first iteration x_1 using the formula:

$$x_1 = x_0 + \alpha_0 \mathbf{p}_0$$

Step 6:

Compute the residual vectors \mathbf{r}_k 's using the formula:

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{p}_k, \quad \forall k = 0, 1, 2, \dots, n-1.$$

Step 7:

Compute the scalars β_k 's using the formula:

$$\beta_k = \frac{\mathbf{r}_{k+1}^t \mathbf{r}_{k+1}}{\mathbf{r}_k^t \mathbf{r}_k}, \quad \forall k = 0, 1, 2, \dots, n-1.$$

Step 8:

Compute the direction vectors \mathbf{p}_k 's using the formula:

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k, \quad \forall k = 0, 1, 2, \dots, n-1.$$

Step 9:

Compute the iterations x_k using the formula [14],[16] and [20] :

$$x_{k+1} = x_k + \alpha_k \mathbf{p}_k, \quad \forall k = 1, 2, \dots, n-1.$$

2.5 Convergence of Iterative Methods

In this section, the general goal is to study the convergence for each previous iterative methods and then make a comparison between them. After that, we will conclude the fastest method. In any computational problem, we get high accuracy if the error becomes very small. In our iterative methods problem, the actual error e is the difference between the exact solution \mathbf{x} and the approximate solution $\mathbf{x}^{(k)}$. But we cannot compute its value since we do not know the exact solution. Instead of that, we will deal with the estimate error which is equal the difference between the approximate solution $\mathbf{x}^{(k)}$ and the next approximate solution $\mathbf{x}^{(k+1)}$. Therefore, we can compute more iterations with less errors and hence we get high level of accuracy.

Suppose \mathbf{x} is the exact solution of the following linear system:

$$A\mathbf{x} = \mathbf{b} \tag{2.5}$$

This can be written in equivalent form as:

$$\mathbf{x} = T\mathbf{x} + C, \tag{2.6}$$

where:

1. T is an $n \times n$ matrix.
2. C is a column vector.

The idea of the iterative methods is to generate a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ that converges to the exact solution \mathbf{x} of the linear system (2.5). (Note that each vector in the sequence is an approximation to the exact solution). To begin the study of convergence, we depend on some definitions and theorems.

Definition 2.5.1 [16]

Suppose $M = \{A: A \text{ is any } n \times n \text{ matrix}\}$. A matrix norm, $\|\cdot\|$, is a real-valued function defined on M . This function satisfies the following properties for any $A, B \in M$ and $\mu \in \mathbb{R}$:

1. $\|A\| \geq 0$,
2. $\|A\| = 0$ if and only if A is the zeromatrix 0 ,
3. $\|\mu A\| = |\mu| \|A\|$,
4. $\|A + B\| \leq \|A\| + \|B\|$,
5. $\|A B\| \leq \|A\| \|B\|$.

Definition 2.5.2 [3]

The spectral radius of any $n \times n$ (square) matrix A is:

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$$

where λ_i 's are the eigenvalues of a matrix A .

(Note: λ_i may be real or complex eigenvalue, then $|\lambda_i|$ is the absolute value or the magnitude of the eigenvalue.)

Definition 2.5.3 [5]

For any $n \times n$ matrix A :

1. $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$ "called the l_1 norm"
2. $\|A\|_2 = \sqrt{\rho(A^T A)}$ "called the l_2 norm"

where A^T is the transpose of the matrix A .

3. $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$ "called the l_∞ norm"

Definition 2.5.4 [16]

We call the $n \times n$ (square) matrix A strictly diagonally dominant if:

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

holds $\forall i = 1, 2, \dots, n$.

Definition 2.5.5 [3]

We call the $n \times n$ (square) matrix A positive definite if A is a symmetric matrix and $\mathbf{c}^t A \mathbf{c} > 0$ for any nonzero n -dimensional column vector \mathbf{c} .

Definition 2.5.6 [3]

An $n \times n$ (square) matrix A is said to be convergent if:

$$\lim_{k \rightarrow \infty} (A^k)_{i,j} = 0 \quad \forall i = 1, 2, \dots, n \text{ and } \forall j = 1, 2, \dots, n.$$

Theorem 2.5.7 [3]

When the linear system $A\mathbf{x} = \mathbf{b}$ converting into equivalent system $\mathbf{x} = T\mathbf{x} + C$ where T is an $n \times n$ iteration matrix. Then the following statements are equivalent:

1. A is convergent matrix.
2. $\rho(T) < 1$.

Theorem 2.5.8 [3]

If the coefficient matrix A for the linear system $A\mathbf{x} = \mathbf{b}$ is strictly diagonally dominant, then the sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ generated by the Jacobi method converges to the unique solution of that system.

Theorem 2.5.9 [3]

For any initial approximation, a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ converges to the exact solution \mathbf{x} if and only if the spectral radius of the square matrix T $\rho(T) < 1$. (T is the iteration matrix).

To see the proof, see reference [3], page 406.

Theorem 2.5.10 [3]

If $\|T\| < 1$, then the sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ converges to a vector $\mathbf{x} \in \mathbb{R}^n$ for any initial approximation vector $\mathbf{x}^{(0)} \in \mathbb{R}^n$.

For more details, see reference [3].

Theorem 2.5.11

Theorem 2.5.8 holds for Gauss-Seidel Method.

Theorem 2.5.12

Theorem 2.5.9 holds for Gauss-Seidel Method.

Theorem 2.5.13

Theorem 2.5.10 holds for Gauss-Seidel Method.

For more details, see reference [3].

Theorem 2.5.14 "Ostrowski-Reich" [3]

If the coefficient matrix A of the linear system $A\mathbf{x} = \mathbf{b}$ is a positive definite matrix and the relaxation parameter (factor) $\omega \in (0,2)$, then the SOR method converges for any choice of initial approximation vector $\mathbf{x}^{(0)}$.

Theorem 2.5.15

Theorem 2.5.8 holds for SOR method.

Theorem 2.5.16

If $\omega > 2$, then the SOR method diverges.

Theorem 2.5.17 [14]

The sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ generated by the Conjugate Gradient algorithm converges to the solution \mathbf{x} of the square linear system $A\mathbf{x} = \mathbf{b}$ of n variables in at most n steps for any choice of initial approximation vector $\mathbf{x}^{(0)}$.

Proof [14]: suppose \mathbf{x} is the exact solution and $\mathbf{x}^{(0)}$ is the initial solution.

The set of direction vectors are orthogonal so they are linearly independent.

Therefore, they span the space \mathbb{R}^n . Hence, we can write:

$$\mathbf{x} - \mathbf{x}^{(0)} = a_0 \mathbf{p}_0 + a_1 \mathbf{p}_1 + a_2 \mathbf{p}_2 + \dots + a_{n-1} \mathbf{p}_{n-1}, \text{ where } a_i \text{'s } \in \mathbb{R}.$$

Multiplying both sides of the last expression by $\mathbf{p}_j^t A$, we obtain

$$\mathbf{p}_j^t A (\mathbf{x} - \mathbf{x}^{(0)}) = \mathbf{p}_j^t A (a_0 \mathbf{p}_0 + a_1 \mathbf{p}_1 + a_2 \mathbf{p}_2 + \dots + a_{n-1} \mathbf{p}_{n-1})$$

Simplifying the above expression, we get:

$$\mathbf{p}_j^t A \mathbf{x} - \mathbf{p}_j^t A \mathbf{x}^{(0)} = a_0 \mathbf{p}_j^t A \mathbf{p}_0 + a_1 \mathbf{p}_j^t A \mathbf{p}_1 + a_2 \mathbf{p}_j^t A \mathbf{p}_2 + \dots + a_{n-1} \mathbf{p}_j^t A \mathbf{p}_{n-1}$$

but $\mathbf{b} = A\mathbf{x}$, $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}^{(0)}$ and $\mathbf{p}_j^t A \mathbf{p}_i = 0$, $\forall i \neq j$

So, it becomes:

$$\mathbf{p}_j^t \mathbf{r}_0 = a_j \mathbf{p}_j^t A \mathbf{p}_j$$

Thus,

$$a_j = \frac{\mathbf{p}_j^t \mathbf{r}_0}{\mathbf{p}_j^t A \mathbf{p}_j} \quad (*)$$

Now, we want to show that $a_j = \alpha_j$, where:

$$\alpha_j = \frac{\mathbf{r}_j^t \mathbf{r}_j}{\mathbf{p}_j^t A \mathbf{p}_j}, \quad \forall j = 0, 1, 2, \dots, n-1.$$

$$\mathbf{x}_j = \mathbf{x}_0 + a_0 \mathbf{p}_0 + a_1 \mathbf{p}_1 + a_2 \mathbf{p}_2 + \dots + a_{j-1} \mathbf{p}_{j-1}$$

Multiply both sides of the last equation by $\mathbf{p}_j^t A$:

$$\begin{aligned} \mathbf{p}_j^t A \mathbf{x}_j &= \mathbf{p}_j^t A (\mathbf{x}_0 + a_0 \mathbf{p}_0 + a_1 \mathbf{p}_1 + a_2 \mathbf{p}_2 + \dots + a_{j-1} \mathbf{p}_{j-1}) \\ &= \mathbf{p}_j^t A \mathbf{x}_0 + \mathbf{p}_j^t A (a_0 \mathbf{p}_0 + a_1 \mathbf{p}_1 + a_2 \mathbf{p}_2 + \dots + a_{j-1} \mathbf{p}_{j-1}) \end{aligned}$$

$$= \mathbf{p}_j^t A x_0 + 0$$

The above can be written as:

$$\mathbf{p}_j^t A x_j - \mathbf{p}_j^t A x_0 = 0$$

or

$$\mathbf{p}_j^t A (x_j - x_0) = 0$$

Therefore,

$$\begin{aligned} \mathbf{p}_j^t \mathbf{r}_0 &= \mathbf{p}_j^t (A\mathbf{x} - A\mathbf{x}^{(0)}) \\ &= \mathbf{p}_j^t A (\mathbf{x} - x_j + x_j - \mathbf{x}^{(0)}) \\ &= \mathbf{p}_j^t A (\mathbf{x} - x_j) \\ &= \mathbf{p}_j^t (A\mathbf{x} - A x_j) \\ &= \mathbf{p}_j^t (\mathbf{b} - A x_j) \end{aligned}$$

$$\mathbf{p}_j^t \mathbf{r}_0 = \mathbf{p}_j^t \mathbf{r}_j$$

Now, put $\mathbf{p}_j^t \mathbf{r}_0 = \mathbf{p}_j^t \mathbf{r}_j$ in (*), then we get:

$$a_j = \frac{\mathbf{p}_j^t \mathbf{r}_0}{\mathbf{p}_j^t A \mathbf{p}_j} = \frac{\mathbf{p}_j^t \mathbf{r}_j}{\mathbf{p}_j^t A \mathbf{p}_j} = \alpha_j$$

This completes the proof [14].

Chapter Three

Numerical Results

In this chapter, we will deal with Laplace equation and Poisson equation as model problems with Dirichlet and Neumann boundary conditions using the Finite Difference Method. Similarly, we will use the Finite Element Method.

Example 3.1

Consider the following Laplace equation

$$u_{xx} + u_{yy} = 0$$

with square domain $R = \{(x,y) / a = 0 < x < b = 1, c = 0 < y < d = 1\}$ subject to Dirichlet boundary conditions given on the boundaries:

$$u(x,1) = x, u(1,y) = y, \text{ and } u(0,y) = u(x,0) = 0$$

as shown in figure 3.1 .

We want to approximate the solution u by using Finite Difference Algorithm:

Step 1: Choose integers $n = m = 3$.

Step 2: Define $h = \frac{b-a}{n} = \frac{1-0}{3} = \frac{1}{3}$

$$\text{and } k = \frac{d-c}{m} = \frac{1-0}{3} = \frac{1}{3}$$

On X – axis, the interval $[0,1]$ is divided into $n = 3$ equal parts of width

$h = \frac{1}{3}$ also the interval $[0,1]$ is divided, on Y – axis, into $m = 3$ equal parts of width $k = \frac{1}{3}$.

Step 3: Define the (horizontal) grid lines as

$$x_i = a + ih, \quad i = 0, 1, 2, n=3.$$

$$\text{for } i=0: \quad x_0 = 0 + (0)\left(\frac{1}{3}\right) = 0 = a$$

$$\text{for } i=1: \quad x_1 = 0 + (1)\left(\frac{1}{3}\right) = \frac{1}{3}$$

$$\text{for } i=2: \quad x_2 = 0 + (2)\left(\frac{1}{3}\right) = \frac{2}{3}$$

$$\text{for } i=3: \quad x_3 = 0 + (3)\left(\frac{1}{3}\right) = \frac{3}{3} = 1 = b$$

In the same manner we can find the vertical grid lines $y_j = c + jk$,

$$j = 0, 1, 2, m=3 \text{ where } y_0 = 0 = c, y_1 = \frac{1}{3}, y_2 = \frac{2}{3}, \text{ and } y_3 = 1 = d.$$

The grid given in the following figure.

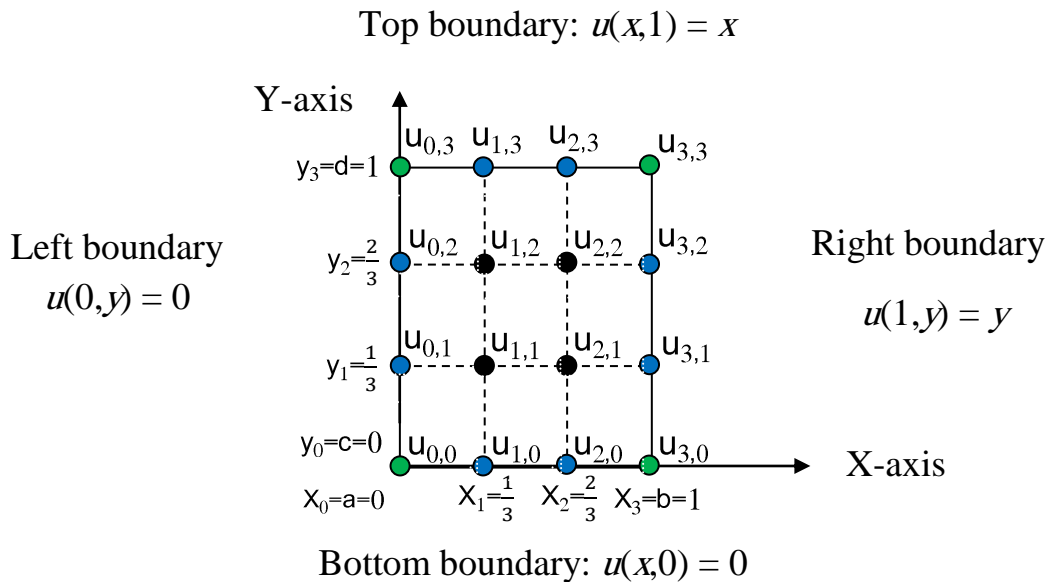


Figure 3.1

The blue points are known boundary points and the green points are corner points that are easy to be calculated by Eq.(1.18). However, the black

(interior) points are not known which are to be approximate.

Now, we use the difference equation (1.17) to approximate the interior (black points) mesh points as follows:

$$u_{i,j} = \frac{1}{4} [u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1}]$$

For $i=1$ and $j=1$

$$\begin{aligned} u_{1,1} &= \frac{1}{4} [u_{1+1,1} + u_{1,1+1} + u_{1-1,1} + u_{1,1-1}] \\ &= \frac{1}{4} [u_{2,1} + u_{1,2} + u_{0,1} + u_{1,0}] \end{aligned} \quad (3.1)$$

but both $u_{0,1}$ and $u_{1,0}$ are known boundary points whereas $u_{2,1}$ and $u_{1,2}$ are not known. So the value of $u_{0,1}$ and $u_{1,0}$ are

$$u_{0,1} = 0 \text{ (on left boundary) and } u_{1,0} = 0 \text{ (on bottom boundary)}$$

So the difference equation (3.1) becomes

$$\begin{aligned} u_{1,1} &= \frac{1}{4} [u_{2,1} + u_{1,2} + 0 + 0] \\ 4u_{1,1} - u_{2,1} - u_{1,2} &= 0 \end{aligned} \quad (3.2)$$

We can label these mesh points as follows:

$$u_{1,1} = u_3, u_{2,1} = u_4, u_{1,2} = u_1 \text{ and } u_{2,2} = u_2$$

$$\text{where } l = i + (m - 1 - j)(n - 1), \forall i = 1, 2. \text{ and } \forall j = 1, 2.$$

after labeling the interior mesh points, then Eq.(3.2) becomes:

$$-u_1 + 4u_3 - u_4 = 0 \quad (3.3)$$

In similar manner, we get the following difference equations

$$\text{For } i=2 \text{ and } j=1: \quad -u_2 - u_3 + 4u_4 = \frac{1}{3} \quad (3.4)$$

$$\text{For } i=1 \text{ and } j=2: \quad 4u_1 - u_2 - u_3 = \frac{1}{3} \quad (3.5)$$

$$\text{For } i=2 \text{ and } j=2: \quad -u_1 + 4u_2 - u_4 = \frac{4}{3} \quad (3.6)$$

rearrange the equations (3.3),(3.4),(3.5), and (3.6) then we get

$$4u_1 - u_2 - u_3 = \frac{1}{3}$$

$$-u_1 + 4u_2 - u_4 = \frac{4}{3}$$

$$-u_1 + 4u_3 - u_4 = 0$$

$$-u_2 - u_3 + 4u_4 = \frac{1}{3}$$

this linear system could be written in matrix form as

$A\mathbf{u} = \mathbf{b}$, where

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \text{ and } \mathbf{b} = \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ 0 \\ \frac{1}{3} \end{bmatrix}$$

If we apply Gaussian elimination to this linear system, then we get the following exact solution:

$$\mathbf{u} = (0.222222, 0.444444, 0.111111, 0.222222)^T$$

We can solve this linear system by any iterative methods like Jacobi method, Gauss-Seidel method, Successive over Relaxation (SOR) method and Conjugate Gradient method.

Jacobi method

It is given by the sequence (2.2):

$$u_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{j=1}^n -a_{ij} u_j^{(k-1)} + b_i \right], i \neq j, a_{ii} \neq 0, i = 1, 2, 3, n = 4.$$

where $k \in \mathbb{N}^*$ and n is the number of the unknown variables.

$$u_1^{(k)} = \frac{1}{4} u_2^{(k-1)} + \frac{1}{4} u_3^{(k-1)} + \frac{1}{12}$$

$$u_2^{(k)} = \frac{1}{4} u_1^{(k-1)} + \frac{1}{4} u_4^{(k-1)} + \frac{1}{3}$$

$$u_3^{(k)} = \frac{1}{4} u_1^{(k-1)} + \frac{1}{4} u_4^{(k-1)}$$

$$u_4^{(k)} = \frac{1}{4} u_2^{(k-1)} + \frac{1}{4} u_3^{(k-1)} + \frac{1}{12}$$

Consider the initial solution is $\mathbf{u}^{(0)} = (u_1^{(0)}, u_2^{(0)}, u_3^{(0)}, u_4^{(0)})^T = (0, 0, 0, 0)^T$.

For $k = 1$ (the first iteration)

$$u_1^{(1)} = \frac{1}{4} u_2^{(1-1)} + \frac{1}{4} u_3^{(1-1)} + \frac{1}{12}$$

$$= \frac{1}{4} u_2^{(0)} + \frac{1}{4} u_3^{(0)} + \frac{1}{12}$$

$$= \frac{1}{12}$$

$$u_2^{(1)} = \frac{1}{4} u_1^{(1-1)} + \frac{1}{4} u_4^{(1-1)} + \frac{1}{3}$$

$$\begin{aligned}
&= \frac{1}{4}u_1^{(0)} + \frac{1}{4}u_4^{(0)} + \frac{1}{3} \\
&= \frac{1}{3} \\
u_3^{(1)} &= \frac{1}{4}u_1^{(1-1)} + \frac{1}{4}u_4^{(1-1)} \\
&= \frac{1}{4}u_1^{(0)} + \frac{1}{4}u_4^{(0)} \\
&= 0 \\
u_4^{(1)} &= \frac{1}{4}u_2^{(1-1)} + \frac{1}{4}u_3^{(1-1)} + \frac{1}{12} \\
u_4^{(1)} &= \frac{1}{4}u_2^{(0)} + \frac{1}{4}u_3^{(0)} + \frac{1}{12} \\
&= \frac{1}{12}
\end{aligned}$$

So the first approximation is $\mathbf{u}^{(1)} = (u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, u_4^{(1)})^T = (\frac{1}{12}, \frac{1}{3}, 0, \frac{1}{12})^T$.

In the same manner, we can find $\mathbf{u}^{(k)}$ approximation if we know $\mathbf{u}^{(k-1)}$ approximation for any $k \geq 1$ where $k \in \mathbb{N}$. Continuing this procedure, we obtain a sequence of approximations.

The following approximate solution is found by Matlab program for the first sixteen iterations:

$$\mathbf{u} = (0.222219, 0.444440, 0.111107, 0.222219)^T$$

To see Matlab code for Jacobi iterative method back to appendix A.

Gauss-Seidel method

It is given by the sequence (2.3):

$$u_i^{(k)} = \frac{1}{a_{ii}} \left[- \sum_{j=1}^{i-1} a_{ij} u_j^{(k)} - \sum_{j=i+1}^n a_{ij} u_j^{(k-1)} + b_i \right],$$

$$a_{ii} \neq 0, i = 1, 2, 3, n=4.$$

where $k \in \mathbb{N}^*$ and n is the number of the unknown variables.

$$u_1^{(k)} = \frac{1}{4} u_2^{(k-1)} + \frac{1}{4} u_3^{(k-1)} + \frac{1}{12}$$

$$u_2^{(k)} = \frac{1}{4} u_1^{(k)} + \frac{1}{4} u_4^{(k-1)} + \frac{1}{3}$$

$$u_3^{(k)} = \frac{1}{4} u_1^{(k)} + \frac{1}{4} u_4^{(k-1)}$$

$$u_4^{(k)} = \frac{1}{4} u_2^{(k)} + \frac{1}{4} u_3^{(k)} + \frac{1}{12}$$

For $k = 1$ (the first iteration)

$$u_1^{(1)} = \frac{1}{4} u_2^{(1-1)} + \frac{1}{4} u_3^{(1-1)} + \frac{1}{12}$$

$$\begin{aligned} u_1^{(1)} &= \frac{1}{4} u_2^{(0)} + \frac{1}{4} u_3^{(0)} + \frac{1}{12} \\ &= \frac{1}{12} \end{aligned}$$

$$\begin{aligned} u_2^{(1)} &= \frac{1}{4} u_1^{(1)} + \frac{1}{4} u_4^{(1-1)} + \frac{1}{3} \\ &= \frac{1}{4} \times \frac{1}{12} + \frac{1}{4} \times 0 + \frac{1}{3} \\ &= \frac{17}{48} \end{aligned}$$

$$\begin{aligned}
 u_3^{(1)} &= \frac{1}{4}u_1^{(1)} + \frac{1}{4}u_4^{(1-1)} \\
 &= \frac{1}{4} \times \frac{1}{12} \\
 &= \frac{1}{48}
 \end{aligned}$$

$$\begin{aligned}
 u_4^{(1)} &= \frac{1}{4}u_2^{(1)} + \frac{1}{4}u_3^{(1)} + \frac{1}{12} \\
 &= \frac{1}{4} \times \frac{17}{48} + \frac{1}{4} \times \frac{1}{48} + \frac{1}{12} \\
 &= \frac{33}{192}
 \end{aligned}$$

So the first approximation is

$$\mathbf{u}^{(1)} = (u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, u_4^{(1)})^T = \left(\frac{1}{12}, \frac{17}{48}, \frac{1}{48}, \frac{33}{192}\right)^T.$$

The following approximate solution is found by Matlab program for the first nine iterations:

$$\mathbf{u} = (0.222219, 0.444443, 0.111110, 0.222222)^T$$

To see Matlab code for Gauss-Seidel iterative method back to appendix B.

SOR Method

The SOR method is given by the sequence (2.4):

$$u_i^{(k)} = (1 - \omega) u_i^{(k-1)} + \omega \frac{1}{a_{ii}} \left[- \sum_{j=1}^{i-1} a_{ij} u_j^{(k)} - \sum_{j=i+1}^n a_{ij} u_j^{(k-1)} + b_i \right]$$

$$a_{ii} \neq 0, \quad i = 1, 2, 3, n = 4.$$

suppose the relaxation factor $\omega = 1.3$

first, write the Gauss-Seidel equations

$$u_1^{(k)} = \frac{1}{4}u_2^{(k-1)} + \frac{1}{4}u_3^{(k-1)} + \frac{1}{12}$$

$$u_2^{(k)} = \frac{1}{4}u_1^{(k)} + \frac{1}{4}u_4^{(k-1)} + \frac{1}{3}$$

$$u_3^{(k)} = \frac{1}{4}u_1^{(k)} + \frac{1}{4}u_4^{(k-1)}$$

$$u_4^{(k)} = \frac{1}{4}u_2^{(k)} + \frac{1}{4}u_3^{(k)} + \frac{1}{12}$$

Now, the SOR equations with $\omega = 1.3$ are:

$$u_1^{(k)} = (1 - 1.3)u_1^{(k-1)} + 1.3\left[\frac{1}{4}u_2^{(k-1)} + \frac{1}{4}u_3^{(k-1)} + \frac{1}{12}\right]$$

$$u_2^{(k)} = (1 - 1.3)u_2^{(k-1)} + 1.3\left[\frac{1}{4}u_1^{(k)} + \frac{1}{4}u_4^{(k-1)} + \frac{1}{3}\right]$$

$$u_3^{(k)} = (1 - 1.3)u_3^{(k-1)} + 1.3\left[\frac{1}{4}u_1^{(k)} + \frac{1}{4}u_4^{(k-1)}\right]$$

$$u_4^{(k)} = (1 - 1.3)u_4^{(k-1)} + 1.3\left[\frac{1}{4}u_2^{(k)} + \frac{1}{4}u_3^{(k)} + \frac{1}{12}\right]$$

where $k \in \mathbb{N}^*$.

For $k = 1$ (the first iteration)

$$u_1^{(1)} = (1 - 1.3)u_1^{(1-1)} + 1.3\left[\frac{1}{4}u_2^{(1-1)} + \frac{1}{4}u_3^{(1-1)} + \frac{1}{12}\right]$$

$$= (-0.3)(0) + 1.3\left[0 + 0 + \frac{1}{12}\right]$$

$$= \frac{13}{120}$$

$$u_2^{(1)} = (1 - 1.3)u_2^{(1-1)} + 1.3\left[\frac{1}{4}u_1^{(1)} + \frac{1}{4}u_4^{(1-1)} + \frac{1}{3}\right]$$

$$\begin{aligned}
&= (-0.3)(0) + 1.3 \left[\frac{1}{4} \times \frac{13}{120} + 0 + \frac{1}{3} \right] \\
&= \frac{2249}{4800} \\
u_3^{(1)} &= (1 - 1.3)u_3^{(1-1)} + 1.3 \left[\frac{1}{4}u_1^{(1)} + \frac{1}{4}u_4^{(1-1)} \right] \\
&= (-0.3)(0) + 1.3 \left[\frac{1}{4} \times \frac{13}{120} + 0 \right] \\
&= \frac{169}{4800} \\
u_4^{(1)} &= (1 - 1.3)u_4^{(1-1)} + 1.3 \left[\frac{1}{4}u_2^{(1)} + \frac{1}{4}u_3^{(1)} + \frac{1}{12} \right] \\
&= (-0.3)(0) + 1.3 \left[\frac{1}{4} \times \frac{2249}{4800} + \frac{1}{4} \times \frac{169}{4800} + \frac{1}{12} \right] \\
&= \frac{52234}{192000}
\end{aligned}$$

The following approximate solution is found by Matlab program for the first nine iterations:

$$\mathbf{u} = (0.222186, 0.444439, 0.111128, 0.222230)^T$$

To see Matlab code for SOR iterative method back to appendix C.

Conjugate Gradient method

It is given by the following algorithm

Step 1: Start with initial guess \mathbf{u}_0 , say

$$\mathbf{u}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Step 2: Calculate the residual vector \mathbf{r}_0 as follows

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$$

$$= \begin{bmatrix} 1 \\ 3 \\ 4 \\ 3 \\ 0 \\ 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

so

$$\mathbf{r}_0 = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 3 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

Step 3: Let the initial direction vector $\mathbf{p}_0 = \mathbf{r}_0$. So

$$\mathbf{p}_0 = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 3 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

Step 4: Compute the scalars α_k 's by the formula

$$\alpha_k = \frac{\mathbf{r}_k^t \mathbf{r}_k}{\mathbf{p}_k^t A \mathbf{p}_k}, \quad \forall k = 0, 1, 2, \dots, n-1.$$

For $k=0$

$$\alpha_0 = \frac{\mathbf{r}_0^t \mathbf{r}_0}{\mathbf{p}_0^t A \mathbf{p}_0}$$

$$\begin{aligned}
& \left[\frac{1}{3} \quad \frac{4}{3} \quad 0 \quad \frac{1}{3} \right] \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ 0 \\ \frac{1}{3} \end{bmatrix} \\
= & \frac{\left[\frac{1}{3} \quad \frac{4}{3} \quad 0 \quad \frac{1}{3} \right] \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ 0 \\ \frac{1}{3} \end{bmatrix}}{6.219} \\
= & \frac{2}{6.219} \\
= & 0.3216
\end{aligned}$$

Step 5: Compute the first iteration \mathbf{u}_1 by the formula

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{u}_0 + \alpha_0 \mathbf{p}_0 \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 0.3216 \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ 0 \\ \frac{1}{3} \end{bmatrix} \\
&= \begin{bmatrix} 0.1071 \\ 0.4286 \\ 0.0000 \\ 0.1071 \end{bmatrix}
\end{aligned}$$

The approximate solution for the first three iterations is given by Matlab code:

$$\mathbf{u} = (0.222222, 0.444444, 0.111111, 0.222222)^T$$

To see Matlab code for conjugate gradient iterative method back to appendix D.

Comparison between the iterative methods

The generated linear system in example 3.1 that should be solved by some iterative techniques, namely: Jacobi, Gauss-Seidel, Successive over Relaxation (SOR), and the Conjugate Gradient methods with $\varepsilon = 10^{-5}$. Table 3.1 shows numerical results for these iterative techniques. Each of them obtains the approximate solution in different number of iterations. However, more iterations give less errors and leads to accurate solutions.

Table 3.1

The exact solution is $\mathbf{u} = (0.222222, 0.444444, 0.111111, 0.222222)^T$					
Method	u_1	u_2	u_3	u_4	number of iterations
Jacobi solution	0.222219	0.444440	0.111107	0.222219	16
Gauss-Seidel solution	0.222219	0.444443	0.111110	0.222222	9
(SOR) solution	0.222186	0.444439	0.111128	0.222230	8
Conjugate Gradient method solution	0.222222	0.444444	0.111111	0.222222	3

Example 3.2

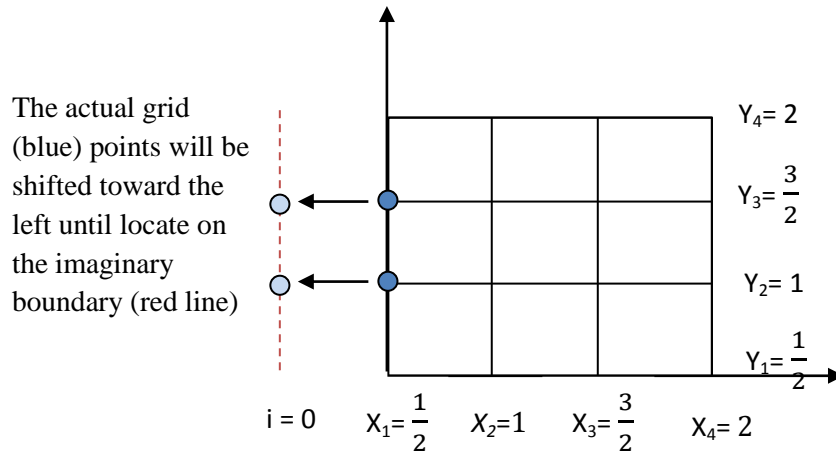
Consider the following Poisson equation

$$u_{xx} + u_{yy} = xy$$

with square domain $R = \{(x,y) / a = 0 < x < b = 2, c = 0 < y < d = 2\}$ with Neumann boundary condition $\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} = g(y) = y$ given on the left

boundary and Dirichlet boundary conditions $u = 1$ on the remaining boundaries. We will use the finite difference method to approximate the solution of Poisson equation.

The mesh size $h = \frac{1}{2}$ as shown in figure 3.2.



we want to put the grid points $(1,j)$ that is the blue points outside the domain toward the left. Let $m = n = 4$, the following are known as boundary conditions for $2 \leq i \leq 4-1$ and $n = 4$

$$u(2,4) = 1, u(3,4) = 1,$$

$$\text{and } 2 \leq j \leq 4-1 \text{ and } m = 4$$

$$u(4,2) = 1, u(4,3) = 1,$$

$$\text{and } 2 \leq i \leq 4-1 \text{ and } j=1$$

$$u(2,1) = 1, \text{ and } u(3,1) = 1.$$

Now, we use Eq.(1.26) to approximate the values of boundary points on left boundary :

$$u(1,j) = \frac{1}{4} [2u(2,j) + 2h g_{1,j} + u(1,j+1) + u(1,j-1)] - \frac{1}{16} G_{i,j}$$

for $2 \leq j \leq 4-1$, $g_{1,j} = g(x_1, y_j) = g(y_j) = y_j$ and $G_{i,j} = x_1 y_j$

$$u(1,2) = \frac{1}{4} [2u(2,2) + 2 \times \frac{1}{2} g_{1,2} + u(1,2+1) + u(1,2-1)] - \frac{1}{16} x_1 y_2$$

$$u(1,2) = \frac{1}{4} [2u(2,2) + u(1,3) + u(1,1) + 1] - \frac{1}{16} \times \frac{1}{2} \times 1$$

but $u(1,1)$ is a corner point which we can evaluate its value by Eq.(1.18)

$$\text{so, } u(1,2) = \frac{1}{4} [2u(2,2) + u(1,3) + \frac{1}{2} + \frac{1}{2}u(1,2) + 1] - \frac{1}{32}$$

rearrange this equation, then we get:

$$\frac{7}{2} u(1,2) - 2u(2,2) - u(1,3) = \frac{11}{8} \quad (3.1)$$

$$\text{now, } u(1,3) = \frac{1}{4} [2u(2,3) + 2 \times \frac{1}{2} g(1,3) + u(1,3+1) + u(1,3-1)] - \frac{1}{16} x_1 y_3$$

$$u(1,3) = \frac{1}{4} [2u(2,3) + u(1,4) + u(1,2) + \frac{3}{2}] - \frac{1}{16} \times \frac{1}{2} \times \frac{3}{2}$$

but $u(1,4)$ is a corner point which we can evaluate its value by Eq.(1.18)

$$\text{so, } u(1,3) = \frac{1}{4} [2u(2,3) + \frac{1}{2}u(1,3) + \frac{1}{2} + u(1,2) + \frac{3}{2}] - \frac{3}{64}$$

rearrange this equation, then we get:

$$\frac{7}{2} u(1,3) - 2u(2,3) - u(1,2) = \frac{29}{16} \quad (3.2)$$

Now, for $i = 2,3$ and $j = 2,3$, we use Eq.(1.20)

$$u(2,2) = \frac{1}{4} [u(3,2) + u(1,2) + u(2,3) + u(2,1)] - \frac{1}{16} \times 1 \times 1$$

but $u(2,1)$ is a known boundary point which is equal to 1

so, substitute its value and then rearrange the equation, then we get

$$4u(2,2) - u(3,2) - u(1,2) - u(2,3) = \frac{3}{4} \quad (3.3)$$

$$u(2,3) = \frac{1}{4} [u(3,3) + u(1,3) + u(2,4) + u(2,2)] - \frac{1}{16} \times 1 \times \frac{3}{2}$$

but $u(2,4)$ is a known boundary point which is equal to 1

so, substitute its value and then rearrange the equation, then we get

$$4u(2,3) - u(3,3) - u(1,3) - u(2,2) = \frac{5}{8} \quad (3.4)$$

$$u(3,2) = \frac{1}{4} [u(4,2) + u(2,2) + u(3,3) + u(3,1)] - \frac{1}{16} \times \frac{3}{2} \times 1$$

but $u(4,2)$ and $u(3,1)$ are known boundary points which are equal to 1

so, substitute their values and then rearrange the equation, then we get

$$4u(3,2) - u(2,2) - u(3,3) = \frac{13}{8} \quad (3.5)$$

$$u(3,3) = \frac{1}{4} [u(4,3) + u(2,3) + u(3,4) + u(3,2)] - \frac{1}{16} \times \frac{3}{2} \times \frac{3}{2}$$

but $u(4,3)$ and $u(3,4)$ are known boundary points which are equal to 1

so, substitute their values and then rearrange the equation, then we get

$$4u(3,3) - u(2,3) - u(3,2) = \frac{23}{16} \quad (3.6)$$

Now, we have six equations in six variables:

$$\frac{7}{2} u(1,2) - 2u(2,2) - u(1,3) = \frac{11}{8}$$

$$\frac{7}{2} u(1,3) - 2u(2,3) - u(1,2) = \frac{29}{16}$$

$$4u(2,2) - u(3,2) - u(1,2) - u(2,3) = \frac{3}{4}$$

$$4u(2,3) - u(3,3) - u(1,3) - u(2,2) = \frac{5}{8}$$

$$4u(3,2) - u(2,2) - u(3,3) = \frac{13}{8}$$

$$4u(3,3) - u(2,3) - u(3,2) = \frac{23}{16}$$

Label the variables as follow

$$u(1,3) = u_1, u(2,3) = u_2, u(3,3) = u_3, u(1,2) = u_4, u(2,2) = u_5, \text{ and } u(3,2) = u_6$$

So, the linear system can be written as:

$$\frac{7}{2}u_1 - 2u_2 - u_4 = \frac{29}{16}$$

$$-u_1 + 4u_2 - u_3 - u_5 = \frac{5}{8}$$

$$-u_2 + 4u_3 - u_6 = \frac{23}{16}$$

$$-u_1 + \frac{7}{2}u_4 - 2u_5 = \frac{11}{8}$$

$$-u_2 - u_4 + 4u_5 - u_6 = \frac{3}{4}$$

$$-u_3 - u_5 + 4u_6 = \frac{13}{8}$$

This linear system should be written in matrix form as follows:

$$\begin{bmatrix} \frac{7}{2} & -2 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & \frac{7}{2} & -2 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} \frac{29}{16} \\ \frac{5}{8} \\ \frac{23}{16} \\ \frac{11}{8} \\ \frac{3}{4} \\ \frac{13}{8} \end{bmatrix}$$

If we apply Gaussian elimination to this linear system, then we get the following exact solution:

$$\mathbf{u} = (1.4694, 0.9758, 0.8179, 1.3788, 0.9908, 0.8584)^T$$

Now, we can solve it by iterative method begin with Jacobi method and end with conjugate gradient method.

Jacobi method

It is given by the sequence (2.2). The using of Matlab program gives approximate solutions in table 3.2:

Table 3.2

Iteration #	u_1	u_2	u_3	u_4	u_5	u_6
1	0.5179	0.1563	0.3594	0.3929	0.1875	0.4063
2	0.7194	0.4224	0.5000	0.6480	0.4263	0.5430
3	0.9444	0.5677	0.6007	0.8420	0.5908	0.6378
4	1.0828	0.6902	0.6608	1.0003	0.6994	0.7041
5	1.1981	0.7670	0.7080	1.1019	0.7862	0.7463
6	1.2710	0.8293	0.7377	1.1844	0.8413	0.7798

The following approximate solution is found by Matlab program for the twenty eight iterations:

$$\mathbf{u} = (1.4693, 0.9757, 0.8179, 1.3787, 0.9907, 0.8584)^T$$

To see Matlab code for Jacobi iterative method back to appendix E.

Gauss-Seidel method

It is given by the sequence (2.3). The first six iterations by Matlab are given in table 3.3:

Table 3.3

Iteration#	u_1	u_2	u_3	u_4	u_5	u_6
1	0.5179	0.2857	0.4308	0.5408	0.3941	0.6125
2	0.8356	0.5714	0.6553	0.8568	0.6977	0.7445
3	1.0892	0.7668	0.7372	1.1027	0.8410	0.8008
4	1.2711	0.8686	0.7767	1.2366	0.9140	0.8289
5	1.3675	0.9208	0.7968	1.3059	0.9514	0.8433
6	1.4171	0.9476	0.8071	1.3414	0.9706	0.8507

To see Matlab code for Gauss-Seidel iterative method back to appendix F.

SOR Method

The SOR method is given by the sequence (2.4). The approximate solution is given by Matlab code after ten iterations:

$$\mathbf{u} = (1.4694, 0.9758, 0.8179, 1.3788, 0.9908, 0.8584)^T$$

To see Matlab code for SOR iterative method back to appendix G.

Conjugate Gradient method

The Conjugate Gradient method is given by the algorithm in section (2.4).

The approximate solutions in table 3.4 are given by Matlab code after six iterations:

Table 3.4

Iteration #	u_1	u_2	u_3	u_4	u_5	u_6
1	1.0105	0.3485	0.8015	0.7666	0.4181	0.9060
2	1.2050	1.0141	0.8372	1.1984	0.9641	0.8698
3	1.4845	0.9950	0.8348	1.3637	1.0463	0.8479
4	1.4962	0.9957	0.8247	1.4041	1.0123	0.8661
5	1.4825	0.9833	0.8241	1.3982	0.9924	0.8665
6	1.4669	0.9695	0.8208	1.3775	0.9832	0.8595

The approximate solution is given by Matlab code after ten iterations:

$$\mathbf{u} = (1.4688, 0.9775, 0.8150, 1.3794, 0.9922, 0.8572)^T$$

To see Matlab code for Conjugate Gradient iterative method back to appendix H.

Comparison between the iterative methods

The generated linear system in example 3.2 that should be solved by some iterative techniques, namely: Jacobi, Gauss-Seidel, Successive over Relaxation (SOR), and the Conjugate Gradient methods with $\varepsilon = 10^{-5}$. Table 3.5 shows numerical results for these iterative techniques. However, more iterations give less errors and leads to accurate solutions.

Table 3.5

The exact solution of the linear system is $\mathbf{u} = (1.4694, 0.9758, 0.8179, 1.3788, 0.9908, 0.8584)^T$							
Method	u_1	u_2	u_3	u_4	u_5	u_6	number of iterations
Jacobi solution	1.4693	0.9757	0.8179	1.3787	0.9907	0.8584	28
Gauss-Seidel solution	1.4693	0.957	0.8179	1.3787	0.9907	0.8584	15
(SOR) solution	1.4694	0.9758	0.8179	1.3788	0.9908	0.8584	10
Conjugate Gradient method solution	1.4688	0.9775	0.8150	1.3794	0.9922	0.8572	10

Example 3.3

Consider the following Laplace equation

$$u_{xx} + u_{yy} = 0$$

with square domain $R = \{(x,y) / a = 0 < x < b = 1, c = 0 < y < d = 1\}$ with Dirichlet boundary conditions given on the boundaries in figure 3.3 . We will use the finite element method to approximate the solution of Laplace equation.

$$u(0,y) = 0 , u(1,y) = y , u(x,0) = 0 , \text{ and } u(x,1) = x.$$

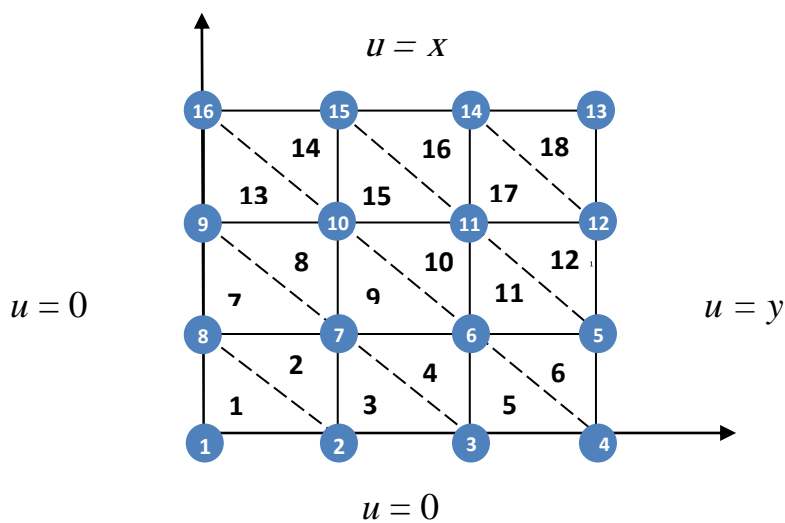


Figure 3.3

The region is divided into 18 equal triangular elements which are identified by encircled numbers 1 through 18 as indicated in figure 3.3. In this discretization there are 16 global nodes (blue points) numbered 1 through 16 as indicated in the figure.

Note that the bottom boundary is partitioned into 3 portions which are from node 1 to node 2, from node 2 to node 3 and from node 3 to node 4. Also, the left boundary is partitioned into 3 portions which are from node 1 to node 8, from node 8 to node 9 and from node 9 to node 16.

So, each portion has $\frac{1}{3} \times 1 = \frac{1}{3}$ length. Now, we will write the coordinates

for each node:

node 1 : (0 , 0), node 2 : ($\frac{1}{3}$, 0), node 3 : ($\frac{2}{3}$, 0), node 4 : (1 , 0)

node 5 : (1 , $\frac{1}{3}$), node 6 : ($\frac{2}{3}$, $\frac{1}{3}$), node 7 : ($\frac{1}{3}$, $\frac{1}{3}$), node 8 : (0 , $\frac{1}{3}$)

node 9 : (0 , $\frac{2}{3}$), node 10 : ($\frac{1}{3}$, $\frac{2}{3}$), node 11 : ($\frac{2}{3}$, $\frac{2}{3}$), node 12 : (1 , $\frac{2}{3}$)

node 13 : (1 , 1), node 14 : ($\frac{2}{3}$, 1), node 15 : ($\frac{1}{3}$, 1), node 16 : (0 , 1)

For each element e , we will label the local node numbers 1, 2, and 3 of element e in a counterclockwise sense.

Table 3.6 shows that for each element we write its global nodes and their local node numbers and coordinates.

Table 3.6

Element #	Its global nodes	local node numbers 1,2, and 3 in a counterclockwise sense	The coordinates of each global node
element 1	1	1	$(x_1, y_1) = (0, 0)$
	2	2	$(x_2, y_2) = (\frac{1}{3}, 0)$
	8	3	$(x_3, y_3) = (0, \frac{1}{3})$
element 2	2	1	$(x_1, y_1) = (\frac{1}{3}, 0)$
	7	2	$(x_2, y_2) = (\frac{1}{3}, \frac{1}{3})$
	8	3	$(x_3, y_3) = (0, \frac{1}{3})$
element 3	2	1	$(x_1, y_1) = (\frac{1}{3}, 0)$
	3	2	$(x_2, y_2) = (\frac{2}{3}, 0)$
	7	3	$(x_3, y_3) = (\frac{1}{3}, \frac{1}{3})$
⋮	⋮	⋮	⋮
element 17	11	1	$(x_1, y_1) = (\frac{2}{3}, \frac{2}{3})$
	12	2	$(x_2, y_2) = (1, \frac{2}{3})$
	14	3	$(x_3, y_3) = (\frac{2}{3}, 1)$
element 18	12	1	$(x_1, y_1) = (1, \frac{2}{3})$
	13	2	$(x_2, y_2) = (1, 1)$
	14	3	$(x_3, y_3) = (\frac{2}{3}, 1)$

Now, for each element e , the following must be computed:

For element 1:

$$P_1 = y_2 - y_3 = 0 - \frac{1}{3} = -\frac{1}{3}$$

$$Q_1 = x_3 - x_2 = 0 - \frac{1}{3} = -\frac{1}{3}$$

$$P_2 = y_3 - y_1 = \frac{1}{3} - 0 = \frac{1}{3}$$

$$Q_2 = x_1 - x_3 = 0 - 0 = 0$$

$$P_3 = y_I - y_2 = 0 - 0 = 0$$

$$Q_3 = x_2 - x_I = \frac{1}{3} - 0 = \frac{1}{3}$$

In the same manner, we compute P_i 's and Q_i 's for each remaining elements where $i = 1, 2, 3$.

Now, we use Eq.(1.27) to write the entries of the 3×3 element coefficient matrix, for example for element 1:

$$C_{ij}^{(e)} = \frac{1}{4A} [P_i P_j + Q_i Q_j], \quad i, j = 1, 2, 3. \quad \text{where}$$

$$\begin{aligned} A &= \frac{1}{2} [P_2 Q_3 - P_3 Q_2] \\ &= \frac{1}{2} \left[\frac{1}{3} \times \frac{1}{3} - 0 \times 0 \right] \\ &= \frac{1}{18} \end{aligned}$$

$$\begin{aligned} C_{11}^{(1)} &= \frac{1}{4 \times \frac{1}{18}} [P_1 P_1 + Q_1 Q_1] \\ &= \frac{18}{4} \left[-\frac{1}{3} \times -\frac{1}{3} + -\frac{1}{3} \times -\frac{1}{3} \right] \\ &= 1 \end{aligned}$$

$$\begin{aligned} C_{12}^{(1)} &= \frac{18}{4} [P_1 P_2 + Q_1 Q_2] \\ &= \frac{18}{4} \left[-\frac{1}{3} \times \frac{1}{3} + -\frac{1}{3} \times 0 \right] \\ &= -\frac{1}{2} \end{aligned}$$

$$C_{13}^{(1)} = \frac{18}{4} [P_1 P_3 + Q_1 Q_3]$$

$$C_{13}^{(1)} = \frac{18}{4} \left[-\frac{1}{3} \times 0 + -\frac{1}{3} \times \frac{1}{3} \right]$$

$$= -\frac{1}{2}$$

$$\begin{aligned} C_{21}^{(1)} &= \frac{18}{4} [P_2 P_1 + Q_2 Q_1] \\ &= \frac{18}{4} \left[\frac{1}{3} \times -\frac{1}{3} + 0 \times -\frac{1}{3} \right] \\ &= -\frac{1}{2} \end{aligned}$$

$$\begin{aligned} C_{22}^{(1)} &= \frac{18}{4} [P_2 P_2 + Q_2 Q_2] \\ &= \frac{18}{4} \left[\frac{1}{3} \times \frac{1}{3} + 0 \times 0 \right] \\ &= \frac{1}{2} \end{aligned}$$

$$\begin{aligned} C_{23}^{(1)} &= \frac{18}{4} [P_2 P_3 + Q_2 Q_3] \\ &= \frac{18}{4} \left[\frac{1}{3} \times 0 + 0 \times \frac{1}{3} \right] \\ &= 0 \end{aligned}$$

$$\begin{aligned} C_{31}^{(1)} &= \frac{18}{4} [P_3 P_1 + Q_3 Q_1] \\ &= \frac{18}{4} \left[0 \times -\frac{1}{3} + \frac{1}{3} \times -\frac{1}{3} \right] \\ &= -\frac{1}{2} \end{aligned}$$

$$\begin{aligned} C_{32}^{(1)} &= \frac{18}{4} [P_3 P_2 + Q_3 Q_2] \\ &= \frac{18}{4} \left[0 \times \frac{1}{3} + \frac{1}{3} \times 0 \right] \\ &= 0 \end{aligned}$$

$$C_{33}^{(1)} = \frac{18}{4} [P_3 P_3 + Q_3 Q_3]$$

$$= \frac{18}{4} \left[0 \times 0 + \frac{1}{3} \times \frac{1}{3} \right]$$

$$= \frac{1}{2}$$

Thus, the 3×3 element coefficient matrix for element 1 is:

$$C^{(1)} = \begin{bmatrix} C_{11}^{(1)} & C_{12}^{(1)} & C_{13}^{(1)} \\ C_{21}^{(1)} & C_{22}^{(1)} & C_{23}^{(1)} \\ C_{31}^{(1)} & C_{32}^{(1)} & C_{33}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

In same manner, we find the 3×3 element coefficient matrix for element 2,3,4,...,18.

$$C^{(2)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(3)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$C^{(4)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(5)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$C^{(6)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(7)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$C^{(8)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(9)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$\begin{aligned}
C^{(10)} &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(11)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \\
C^{(12)} &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(13)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \\
C^{(14)} &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(15)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \\
C^{(16)} &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, C^{(17)} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \\
C^{(18)} &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}
\end{aligned}$$

Now, the global coefficient matrix C is then assembled from the element coefficient matrices. Since there are 16 nodes, the global coefficient matrix will be a 16×16 matrix. The one diagonal entries can be computed as follows:

Take for example $C_{l,l}$:

The entry $C_{l,l}$ in the global coefficient matrix C corresponds to node 1 which belongs to element 1 but node 1 is assigned local node number 1 in

element 1, thus the entry $C_{1,1}$ equals to

$$C_{1,1} = C_{11}^{(1)} = 1$$

Also the entry $C_{2,2}$ in the global coefficient matrix corresponds to node 2 which belongs to elements 1, 2 and 3 but node 2 is assigned local node number 1 in elements 2 and 3 and local number 2 in element 1, thus the entry $C_{2,2}$ equals to

$$C_{2,2} = C_{22}^{(1)} + C_{11}^{(2)} + C_{11}^{(3)} = 0.5 + 0.5 + 1 = 2$$

In the same manner, we can find the remaining entries:

$$C_{3,3} = C_{22}^{(3)} + C_{11}^{(4)} + C_{11}^{(5)} = 0.5 + 0.5 + 1 = 2$$

$$C_{4,4} = C_{22}^{(5)} + C_{11}^{(6)} = 0.4 + 0.7 = 1$$

$$C_{5,5} = C_{22}^{(6)} + C_{22}^{(11)} + C_{11}^{(12)} = 1 + 0.4 + 0.7 = 2$$

$$C_{6,6} = C_{22}^{(4)} + C_{33}^{(5)} + C_{33}^{(6)} + C_{11}^{(11)} + C_{11}^{(10)} + C_{22}^{(9)} = 4$$

$$C_{7,7} = C_{22}^{(2)} + C_{33}^{(3)} + C_{33}^{(4)} + C_{11}^{(9)} + C_{11}^{(8)} + C_{22}^{(7)} = 4$$

$$C_{8,8} = C_{33}^{(1)} + C_{33}^{(2)} + C_{11}^{(7)} = 2$$

$$C_{9,9} = C_{33}^{(7)} + C_{33}^{(8)} + C_{11}^{(13)} = 2$$

$$C_{10,10} = C_{22}^{(8)} + C_{33}^{(9)} + C_{33}^{(10)} + C_{11}^{(15)} + C_{11}^{(14)} + C_{22}^{(13)} = 4$$

$$C_{11,11} = C_{22}^{(10)} + C_{33}^{(11)} + C_{33}^{(12)} + C_{11}^{(17)} + C_{11}^{(16)} + C_{22}^{(15)} = 4$$

$$C_{12,12} = C_{22}^{(12)} + C_{22}^{(17)} + C_{11}^{(18)} = 2$$

$$C_{13,13} = C_{22}^{(8)} = 1$$

$$C_{14,14} = C_{22}^{(16)} + C_{33}^{(17)} + C_{33}^{(18)} = 2$$

$$C_{15,15} = C_{22}^{(14)} + C_{33}^{(15)} + C_{33}^{(16)} = 2$$

$$C_{16,16} = C_{33}^{(13)} + C_{33}^{(14)} = 1$$

Now, the one off-diagonal entries can be computed as follows:

For the off-diagonal entry $C_{7,10}$, for example, the global link 7–10 corresponds to local link 1–2 of element 8 and local link 1–3 of element 9 as shown in figure 3.3 and hence

$$C_{7,10} = C_{12}^{(8)} + C_{13}^{(9)} = -0.5 + -0.5 = -1$$

We can compute the value of other off-diagonal entries in the same manner.

The global coefficient matrix C is then assembled from the element coefficient matrices. Since there are 16 nodes, the global coefficient matrix will be a 16×16 matrix.

$$\begin{bmatrix}
 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 -\frac{1}{2} & 2 & -\frac{1}{2} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & -\frac{1}{2} & 2 & -\frac{1}{2} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & -\frac{1}{2} & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 \\
 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -\frac{1}{2} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -\frac{1}{2} & 2 & -\frac{1}{2} & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -\frac{1}{2} & 2 & -\frac{1}{2} \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1
 \end{bmatrix}$$

The global coefficient matrix C . Red numbers are the entries of matrix C_{yn} while blue numbers are the entries of matrix C_{vy} both are discussed later.

Now, defining the vector \mathbf{u}_v to be vector of unknowns (interior nodes) and vector \mathbf{u}_n to be vector of prescribed boundary values (nodes that are located on the boundaries) as shown in table 3.7.

Table 3.7: represents vector of prescribed boundary values (nodes that are located on the boundaries).

Global Node (Boundary Node)	Boundary condition	The value of Global Node \mathbf{u}_n
1 (corner node)	Depend on bottom and left boundaries $u = 0$ and $u = 0$ respectively.	The average of its boundary values $\frac{0+0}{2} = 0$
2	Depend on bottom boundary only $u = 0$.	0
3	Depend on bottom boundary only $u = 0$.	0
4 (corner node)	Depend on bottom and right boundaries $u = 0$ and $u(1,y) = y$ respectively.	The coordinate of node 4 is (1,0) so its value under right condition is $u(1,0) = 0$. So, The average of its boundary values is $\frac{0+0}{2} = 0$
5	Depend on right boundary only $u(1,y) = y$.	The coordinate of node 5 is $(1, \frac{1}{3})$ so its value under right condition is $u(1, \frac{1}{3}) = \frac{1}{3}$. So, the value of node 5 is $\frac{1}{3}$.
8	Depend on left boundary only $u = 0$.	0
9	Depend on left boundary only $u = 0$.	0

12	Depend on right boundary only $u(1,y) = y$.	The coordinate of node 5 is $(1, \frac{2}{3})$ so its value under right condition is $u(1, \frac{2}{3}) = \frac{2}{3}$. So, the value of node 5 is $\frac{2}{3}$.
13 (corner node)	Depend on top and right boundaries $u(x,1) = x$ and $u(1,y) = y$ respectively.	The coordinate of node 13 is (1,1) so its value under top condition is $u(1,1) = 1$ and its value under right condition is $u(1,1) = 1$. So, the average value of node 13 is $\frac{1+1}{2} = 1$.
Global Node (Boundary Node)	Boundary condition	The value of Global Node
14	Depend on top boundary only $u(x,1) = x$.	The coordinate of node 14 is $(\frac{2}{3}, 1)$ so its value under top condition is $u(\frac{2}{3}, 1) = \frac{2}{3}$. So, the value of node 5 is $\frac{2}{3}$.
15	Depend on top boundary only $u(x,1) = x$.	The coordinate of node 15 is $(\frac{1}{3}, 1)$ so its value under top condition is $u(\frac{1}{3}, 1) = \frac{1}{3}$. So, the value of node 5 is $\frac{1}{3}$.
16 (corner node)	Depend on left and top boundaries $u = 0$ and $u(x,1) = x$ respectively.	The coordinate of node 16 is (0,1) so its value under top condition is $u(0,1) = 0$. So, the average value of node 16 is $\frac{0+0}{2} = 0$.

So,

$$\mathbf{u}_n = (0, 0, 0, 0, \frac{1}{3}, 0, 0, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0)^T$$

Now, defining the matrix C_{vv} to be the matrix of unknown nodes (interior nodes) as in table 3.8 and the matrix C_{vn} to be the matrix of unknown nodes and prescribed boundary values as in table 3.9. Both matrices C_{vv} and C_{vn} obtained from global coefficient matrix C .

Table 3.8

C_{vv}	6	7	10	11
6	4	-1	0	-1
7	-1	4	-1	0
10	0	-1	4	-1
11	-1	0	-1	4

Table 3.9

C_{vn}	1	2	3	4	5	8	9	12	13	14	15	16
6	0	0	-1	0	-1	0	0	0	0	0	0	0
7	0	-1	0	0	0	-1	0	0	0	0	0	0
10	0	0	0	0	0	0	-1	0	0	0	-1	0
11	0	0	0	0	0	0	0	-1	0	-1	0	0

Now, the inverse of matrix C_{vv} is

$$C_{vv}^{-1} = \begin{bmatrix} 0.291667 & 0.083333 & 0.041667 & 0.083333 \\ 0.083333 & 0.291667 & 0.083333 & 0.041667 \\ 0.041667 & 0.083333 & 0.291667 & 0.083333 \\ 0.083333 & 0.041667 & 0.083333 & 0.291667 \end{bmatrix}$$

The vector \mathbf{u}_v of unknowns nodes can be found by using Eq.(1.28):

$$\mathbf{u}_v = -C_{vv}^{-1}C_{vn}\mathbf{u}_n$$

$$= - \begin{bmatrix} 0.291667 & 0.083333 & 0.041667 & 0.083333 \\ 0.083333 & 0.291667 & 0.083333 & 0.041667 \\ 0.041667 & 0.083333 & 0.291667 & 0.083333 \\ 0.083333 & 0.041667 & 0.083333 & 0.291667 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \frac{1}{3} \\ \frac{2}{3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.222222 \\ 0.111111 \\ 0.222222 \\ 0.444445 \end{bmatrix}$$

Therefore, the approximate values of unknown nodes (interior nodes) are:

$$\mathbf{u}_v = \begin{bmatrix} \text{node 6} \\ \text{node 7} \\ \text{node 10} \\ \text{node 11} \end{bmatrix} = \begin{bmatrix} 0.222222 \\ 0.111111 \\ 0.222222 \\ 0.444445 \end{bmatrix}$$

Example 3.4

A very simple form of the steady state heat conduction in the rectangular domain shown in the following:

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

for $x \in [0, a], y \in [0, b]$, with $a = 4$, $b = 2$.

where $u(x, y)$ is the steady state temperature distribution in the domain.

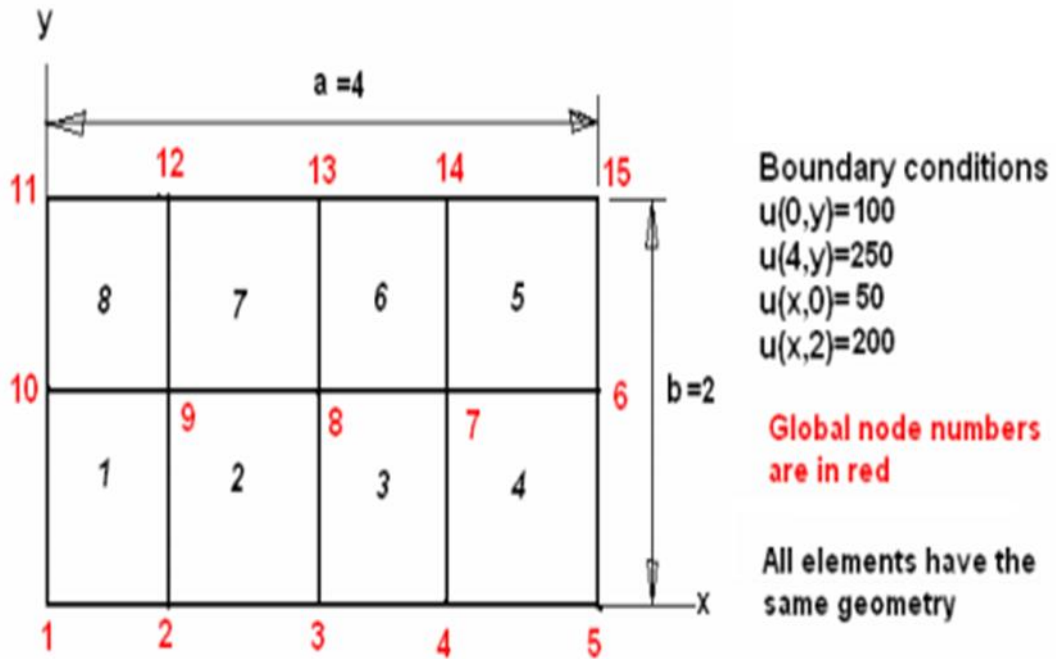
The boundary conditions are:

$u(0, y) = U_L = 100$, imposed temperatures on the left boundary.

$u(a, y) = U_R = 250$, imposed temperatures on the right boundary.

$u(x, 0) = U_B = 50$, imposed temperatures on the bottom boundary.

$u(x, b) = U_T = 200$, imposed temperatures on the top boundary.



The solution along the line $y = 1.5$ (listed in Table 1, and shown in Figure 1) was also computed at the locations $x = 0.0, 0.5, 1.0, 1.5, 2.0, 2.5, \dots, 4.0$ for comparison with the Finite Difference solution. Better agreement should be obtained between the two results by using a finer grid for the FD solution, and by using higher level h -meshing for the FE solution.

Table 1: Comparison between FE and FD solutions

x	u(x,1.5) FE solution	u(x,1.5) FD solution
0	100	100
0.50	142.3	141.7
1.00	159.6	156.1
1.50	161.8	161.7
2.00	164.1	165.8
2.50	171.2	172.0
3.00	178.3	183.8
3.50	207.9	207.0
4.00	250	250

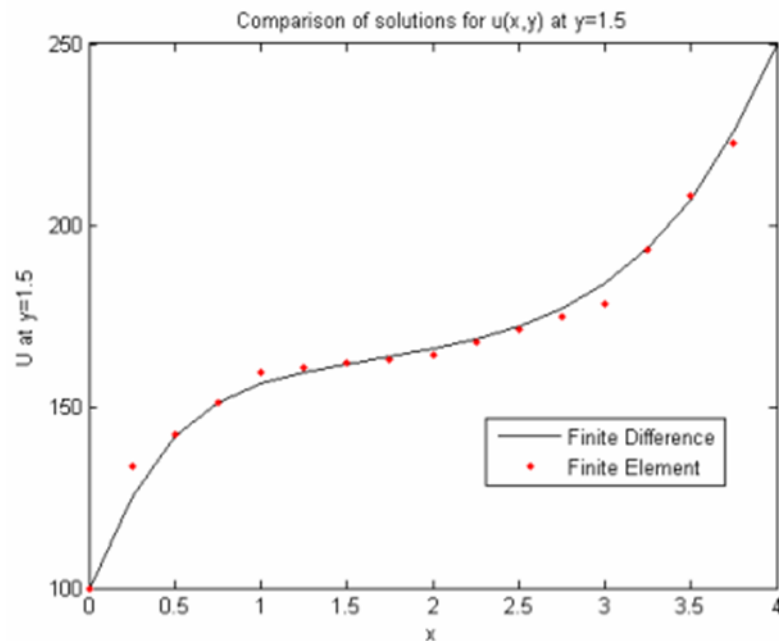


Figure 1

These results are taken from reference [1].

Example 3.5

Another example arising in electrostatics. Consider the charge-free region depicted in Figure 2. The region has prescribed potentials along its boundaries.

The potential $V = V(x, y)$ at an interior point (x, y) within the region is governed by the two-dimensional Laplace's equation:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

The triangular region is divided into a rectangular grid of nodes, with the numbering of free nodes as indicated in the figure.

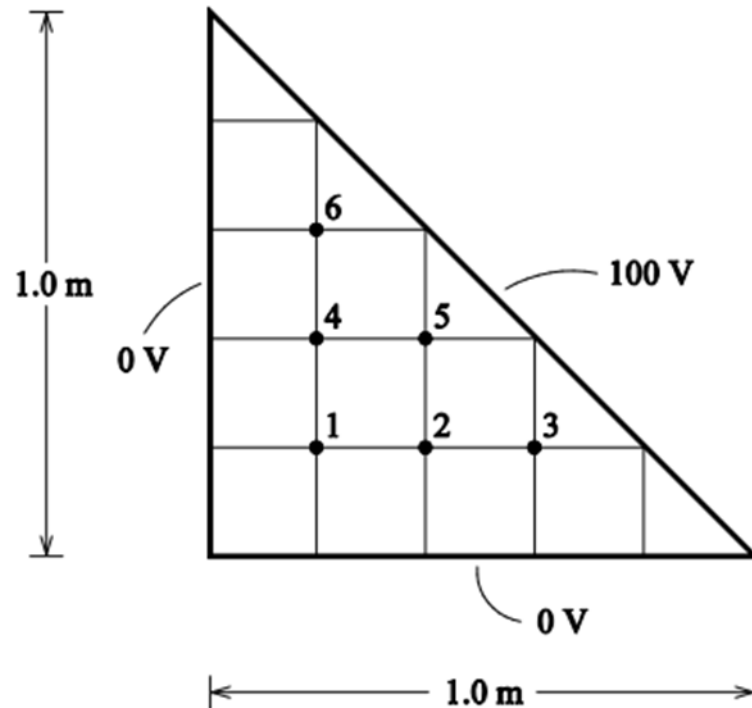


Figure 2: Charge-free region showing prescribed potentials at the boundaries and rectangular grid of free nodes to illustrate the finite difference method.

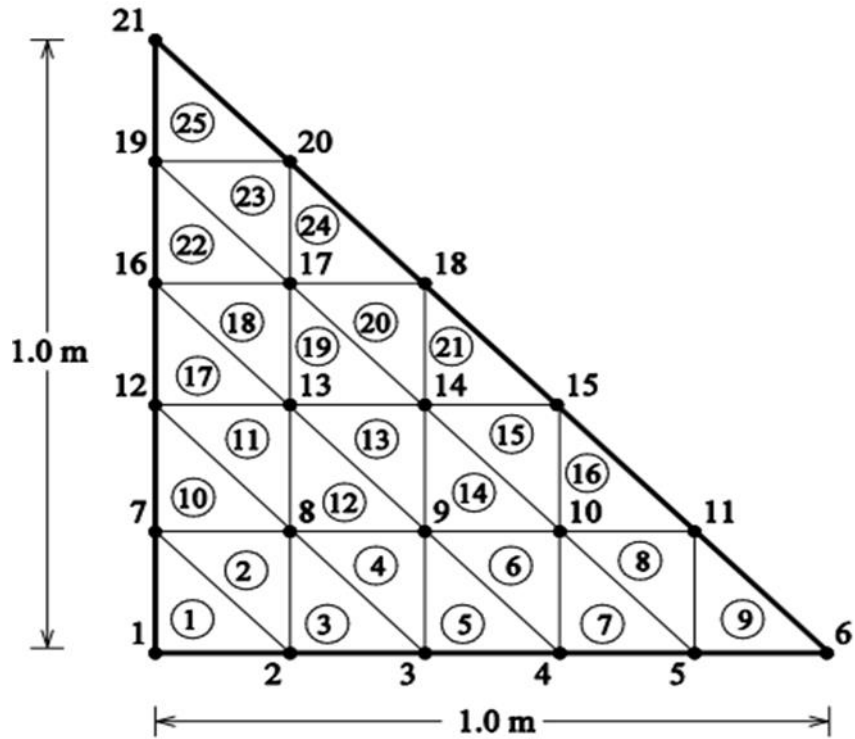


Figure 3: Finite element arrangement for electrostatic problem.

Table 2: Comparison of results obtained by FDM and FEM.

Finite difference		Finite element	
Node	Potential (V)	Node	Potential (V)
1	18.180	8	18.182
2	36.363	9	36.364
3	59.091	10	59.091
4	36.363	13	36.364
5	68.181	14	68.182
6	59.091	17	59.091

These results are taken from reference [11].

3.1 Comparison between results for finite difference method and finite element method:

A simple comparison between the results in example 3.1 and example 3.3 as in table 3.10 with $\varepsilon = 10^{-5}$:

Table 3.10

Finite difference method (using SOR iterative method)		Finite element method	
$u_1 = u_{1,2}$	0.222186	Node 10	0.222222
$u_2 = u_{2,2}$	0.444439	Node 11	0.444445
$u_3 = u_{1,1}$	0.111128	Node 7	0.111111
$u_4 = u_{2,1}$	0.222230	Node 6	0.222222

A better approximation can be obtained if more iterations of SOR method are performed.

We also see from example 3.4 the difference between the FDM and the FEM as in table 1:

Table 1: Comparison between FE and FD solutions

x	u(x,1.5) FE solution	u(x,1.5) FD solution
0	100	100
0.50	142.3	141.7
1.00	159.6	156.1
1.50	161.8	161.7
2.00	164.1	165.8
2.50	171.2	172.0
3.00	178.3	183.8
3.50	207.9	207.0
4.00	250	250

Example 3.5 gives simple comparison of results obtained by FDM and FEM in table 2:

Table 2: Comparison of results obtained by FDM and FEM.

Finite difference		Finite element	
Node	Potential (V)	Node	Potential (V)
1	18.180	8	18.182
2	36.363	9	36.364
3	59.091	10	59.091
4	36.363	13	36.364
5	68.181	14	68.182
6	59.091	17	59.091

3.2 Conclusions

In this thesis we have used the two numerical techniques, namely: the finite difference method and the finite element method to solve boundary value problems involving the Laplace equation and the Poisson equation. The discretization procedure transfers the BVP into a linear system of n -algebraic equations.

This linear system has been solved iteratively by various iterative schemes. These are: Jacobi, the Gauss-Seidel, Successive over Relaxation (SOR), and the Conjugate Gradient methods.

We observe that the finite difference method is very simple and efficient method for approximating the solution of the BVP when the domain has regular shape. On the other hand the finite element method is more efficient for complex and irregular domains. Moreover, we see clearly that the SOR iterative scheme is the most efficient method among the other iterative schemes for approximating the solution of the BVP.

References

1. L. Agbezuge, *Finite Element Solution of the Poisson equation with Dirichlet Boundary Conditions in a rectangular domain*. Rochester Institute of Technology, Rochester, NY, 2006.
2. V. Bokil and N. Gibson, *Finite Difference, Finite Element and Finite Volume Methods for the Numerical Solution of PDEs*, DOE Multiscale Summer School, June, 30, 2007.
3. L. Burden & J. Faires, **NUMERICAL ANALYSIS**, PWS- KENT Publishing Company, Fourth Edition, 1989 Edition.
4. D. Causon and C. Mingham, *Introductory Finite Difference Methods PDEs*, 2010, Ventus Publishing ApS ISBN 978-87-7681-642-1.
5. S. Chapra, *Applied Numerical Methods with MATLAB for Engineers*, Published by the McGraw-Hill Companies 2012, Third Edition.
6. Z. Chen, *Finite Element Methods and Their Applications*, Springer-Verlag Berlin Heidelberg, 2005.
7. R. Clough, *The Finite Element Method in Plane Stress Analysis*, American Society of Civil Engineers, 1960.
8. M. Davis, *Numerical Methods and Modeling for Chemical Engineers*, 1984 by John Wiley & Sons, Inc.
9. C. Johson, **Numerical Solution of Partial Differential Equations by the Finite Element Method**. Royal Institute of Technology, Stockholm. Copyright © 1987,2009 by Claes Johnson.

10. I. Kalambi, *A Comparison of three Iterative Methods for the Solution of Linear Equations*, JASEM ISSN 1119-8362, J. Appl. Sci. Environ. Manage, 2008.
11. M. Lau and S. Kuruganty, *Spreadsheet Implementations for Solving Boundary-Value Problems in Electromagnetics*, Spreadsheets in Education (eJSiE): Vol. 4: Iss. 1, Article 1, Sastry P. 2010.
12. R. LeVeque, *Finite Difference Methods for Differential Equations*, Draft Version for use in the course A Math 585–586 University of Washington, R. J. LeVeque, 1998–2005.
13. G. Micula and S. Micula, *Mathematics and Its Applications*, Springer Science+Business Media Dordrecht, Originally published by Kluwer Academic Publishers in 1999, 1st edition.
14. J. Nocedal and S. Wright, *Numerical Optimization*, 1999 Springer-Verlag New York, Inc.
15. W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes in C*, second edition, Published by the Press Syndicate of the University of Cambridge, 1992.
16. Y. Saad, *Iterative Methods for Sparse Linear Systems*, Second Edition, the Society for Industrial and Applied Mathematics, 2003.
17. R. Sekhar, *Numerical methods of Ordinary and Partial Differential Equations*, Indian Institute of Technology, Jul 22, 2013.
18. P. Seshu, *Textbook of Finite Element Analysis*, 2012 by PHI Learning Private Limited, New Delhi.

19. V. Thomee, *From finite differences to finite elements A short history of numerical analysis of partial differential equations*, Elsevier Science B.V., 2001.
20. C. Vuik, *Iterative solution methods*, Research School for Fluid Mechanics, Delft Institute of Applied Mathematics, 2015.
21. R. Wannan, *Multigrid Methods for Elliptic Partial Differential Equations*, MSc. Thesis, An-Najah National University, 2010.
22. S. Yip, *Handbook of Materials Modeling*, Springer 2005, Volume 1, P. 1–32.
23. O.C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, Vol. 1. Butterworth-Heinemann, Oxford, Fifth Edition, 2000.

Appendix A

Matlab code for Jacobi iterative method

```
% Iterative Solutions of linear equations: Jacobi Method
```

```
% Linear system:  $A u = B$ 
```

```
% Coefficient matrix A, right-hand side vector B
```

```
A=[4 -1 -1 0; -1 4 0 -1; -1 0 4 -1; 0 -1 -1 4];
```

```
B= [1/3;4/3;0;1/3];
```

```
% Set initial value of u to zero column vector
```

```
u0=zeros(1,4);
```

```
% Set Maximum iteration number k_max
```

```
k_max=6;
```

```
% Set the convergence control parameter erp
```

```
erp=0.0001;
```

```
% Show the q matrix
```

```
% loop for iterations
```

```
for k=1:k_max
```

```
    for i=1:4
```

```
        s=0.0;
```

```
        for j=1:4
```

```
            if j==i
```

```
                continue
```

```
            else
```

```
                s=s+A(i,j)*u0(j);
```

```
            end
```



```
end
u1(i)=(B(i)-s)/A(i,i);
end
if norm(u1-u0)<erp
    break
else
    u0=u1;
end
end
% show the final solution
u=u1
% show the total iteration number
n_iteration=k
```

Appendix B

Matlab code for Gauss-Seidel iterative method

```
clear;clc
```

```
format compact
```

```
%% Read or Input any square Matrix
```

```
A = [4 -1 -1 0;
```

```
    -1 4 0 -1;
```

```
    -1 0 4 -1;
```

```
    0 -1 -1 4];% coefficients matrix
```

```
C = [1/3;4/3;0;1/3];% constants vector
```

```
n = length(C);
```

```
X = zeros(n,1);
```

```
Error_eval = ones(n,1);
```

```
%% Check if the matrix A is diagonally dominant
```

```
for i = 1:n
```

```
    j = 1:n;
```

```
    j(i) = [];
```

```
    B = abs(A(i,j));
```

```
    Check(i) = abs(A(i,i)) - sum(B); % Is the diagonal value greater than the
remaining row values combined?
```

```
    if Check(i) < 0
```

```
        fprintf('The matrix is not strictly diagonally dominant at row
%2i\n\n',i)
```

```
    end
```

```

end

%% Start the Iterative method

iteration = 0;

while max(Error_eval) > 0.001

    iteration = iteration + 1;

    Z = X; % save current values to calculate error later

    for i = 1:n

        j = 1:n; % define an array of the coefficients' elements

        j(i) = []; % eliminate the unknown's coefficient from the remaining
coefficients

        Xtemp = X; % copy the unknowns to a new variable

        Xtemp(i) = []; % eliminate the unknown under question from the set
of values

        X(i) = (C(i) - sum(A(i,j) * Xtemp)) / A(i,i);

    end

    Xsolution(:,iteration) = X;

    Error_eval = sqrt((X - Z).^2);

end

%% Display Results

GaussSeidelTable = [1:iteration;Xsolution]'

MaTrIx = [A X C]

```

Appendix C

Matlab code for SOR method.

```
clc
```

```
clear all
```

```
A = [4 -1 -1 0 ; -1 4 0 -1; -1 0 4 -1; 0 -1 -1 4];
```

```
b = [1/3; 4/3; 0; 1/3];
```

```
% error tolerance
```

```
tol = 0.0001;
```

```
%initial guess:
```

```
x0 = zeros(4,1);
```

```
% Jacobi method
```

```
%-----
```

```
xnew=x0;
```

```
error=1;
```

```
while error>tol
```

```
    xold=xnew;
```

```
    for i=1:length(xnew)
```

```
        off_diag = [1:i-1 i+1:length(xnew)];
```

```
        xnew(i) = 1/A(i,i)*( b(i)-sum(A(i,off_diag)*xold(off_diag)) );
```

```
    end
```

```
    error=norm(xnew-xold)/norm(xnew);
```

```
end
```

```
x_jacobian=xnew
```

```
%Gauss?Seidel:
```

```

%-----
maxiter=6;
lambda=1;
n=length(x0);
x=x0;
error=1;
iter = 0;
while (error>tol & iter<maxiter)
    xold=x;
    for i=1:n
        I = [1:i-1 i+1:n];
        x(i) = (1-lambda)*x(i)+lambda/A(i,i)*( b(i)-A(i,I)*x(I) );
    end
    error = norm(x-xold)/norm(x);
    iter = iter+1;
end
x_siedal=x
%SOR
%-----
lambda=1.3;
n=length(x0);
x=x0;
error=1;
iter = 0;

```

```
while (error>tol & iter<maxiter)
    xold=x;
    for i=1:n
        I = [1:i-1 i+1:n];
        x(i) = (1-lambda)*x(i)+lambda/A(i,i)*( b(i)-A(i,I)*x(I) );
    end
    error = norm(x-xold)/norm(x);
    iter = iter+1;
end
x_SOR=x
```

Appendix D

Matlab code for conjugate gradient method.

```
function [u, niter, flag] = solveCG(A, f, s, tol, maxiter)
% SOLVECG  Conjugate Gradients method.
%
%   Input parameters:
%       A : Symmetric, positive definite NxN matrix
%       f : Right-hand side Nx1 column vector
%       s : Nx1 start vector (the initial guess)
%       tol : relative residual error tolerance for break
%           condition
%       maxiter : Maximum number of iterations to perform
%
%   Output parameters:
%       u : Nx1 solution vector
%       niter : Number of iterations performed
%       flag : 1 if convergence criteria specified by TOL could
%           not be fulfilled within the specified maximum
%           number of iterations, 0 otherwise (= iteration
%           successful).
A=[4 -1 -1 0; -1 4 0 -1; -1 0 4 -1; 0 -1 -1 4]
f=[1/3;4/3;0;1/3]
s=[0;0;0;0]
maxiter = 6
```

```

u = s;      % Set u_0 to the start vector s
r = f - A*s; % Compute first residuum
p = r;
rho = r'*r;
niter = 0;  % Init counter for number of iterations
flag = 0;   % Init break flag
% Compute norm of right-hand side to take relative residuum as
% break condition.
normf = norm(f);
if normf < eps % if the norm is very close to zero, take the
               % absolute residuum instead as break condition
               % ( norm(r) > tol ), since the relative
               % residuum will not work (division by zero).
warning(['norm(f) is very close to zero, taking absolute residuum' ...
        ' as break condition.']);
        normf = 1;
end
while (norm(r)/normf > 0.00001) % Test break condition
    a = A*p;
    alpha = rho/(a'*p);
    u = u + alpha*p;
    r = r - alpha*a;
    rho_new = r'*r;
    p = r + rho_new/rho * p;

```



```
rho = rho_new;
niter = niter + 1;
if (niter == maxiter)    % if max. number of iterations
    flag = 1;           % is reached, break.
    break
end
end
end
```

Appendix E

Matlab code for Jacobi method.

```
% Iterative Solutions of linear equations: Jacobi Method
```

```
% Linear system: A x = B
```

```
% Coefficient matrix A, right-hand side vector B
```

```
A=[7/2 -2 0 -1 0 0; -1 4 -1 0 -1 0; 0 -1 4 0 0 -1; -1 0 0 7/2 -2 0; 0 -1 0 -1 4 -1; 0 0 -1 0 -1 4];
```

```
B= [29/16; 5/8; 23/16; 11/8; 3/4; 13/8];
```

```
% Set initial value of x to zero column vector
```

```
x0=zeros(1,6);
```

```
% Set Maximum iteration number k_max
```

```
k_max=28;
```

```
% Set the convergence control parameter erp
```

```
erp=0.0001;
```

```
% Show the q matrix
```

```
% loop for iterations
```

```
for k=1:k_max
```

```
    for i=1:6
```

```
        s=0.0;
```

```
        for j=1:6
```

```
            if j==i
```

```
                continue
```

```
            else
```

```
                s=s+A(i,j)*x0(j);
```

```
    end
end
x1(i)=(B(i)-s)/A(i,i);
end
if norm(x1-x0)<erp
    break
else
    x0=x1;
end
end
% show the final solution
x=x1
% show the total iteration number
n_iteration=k
```

Appendix F

Matlab code for Gauss-Seidel method.

```
clear;clc
```

```
format compact
```

```
%% Read or Input any square Matrix
```

```
A = [7/2 -2 0 -1 0 0; -1 4 -1 0 -1 0; 0 -1 4 0 0 -1; -1 0 0 7/2 -2 0; 0 -1 0 -1 4  
-1; 0 0 -1 0 -1 4]; % coefficients matrix
```

```
C = [29/16; 5/8; 23/16; 11/8; 3/4; 13/8];% constants vector
```

```
n = length(C);
```

```
X = zeros(n,1);
```

```
Error_eval = ones(n,1);
```

```
%% Check if the matrix A is diagonally dominant
```

```
for i = 1:n
```

```
    j = 1:n;
```

```
    j(i) = [];
```

```
    B = abs(A(i,j));
```

```
    Check(i) = abs(A(i,i)) - sum(B); % Is the diagonal value greater than the  
remaining row values combined?
```

```
    if Check(i) < 0
```

```
        fprintf('The matrix is not strictly diagonally dominant at row  
%2i\n\n',i)
```

```
    end
```

```
end
```

```

%% Start the Iterative method

iteration = 0;
while max(Error_eval) > 0.001
    iteration = iteration + 1;
    Z = X; % save current values to calculate error later
    for i = 1:n
        j = 1:n; % define an array of the coefficients' elements
        j(i) = []; % eliminate the unknown's coefficient from the remaining
coefficients
        Xtemp = X; % copy the unknowns to a new variable
        Xtemp(i) = []; % eliminate the unknown under question from the set
of values
        X(i) = (C(i) - sum(A(i,j) * Xtemp)) / A(i,i);
    end
    Xsolution(:,iteration) = X;
    Error_eval = sqrt((X - Z).^2);
end

%% Display Results

GaussSeidelTable = [1:iteration;Xsolution]'
MaTrIx = [A X C]

```

Appendix G

Matlab code for SOR method.

```
clc
```

```
clear all
```

```
A = [7/2 -2 0 -1 0 0; -1 4 -1 0 -1 0; 0 -1 4 0 0 -1; -1 0 0 7/2 -2 0; 0 -1 0 -1 4  
-1; 0 0 -1 0 -1 4]; % coefficients matrix
```

```
b = [29/16; 5/8; 23/16; 11/8; 3/4; 13/8]
```

```
% error tolerance
```

```
tol = 0.0001;
```

```
%initial guess:
```

```
x0 = zeros(6,1);
```

```
% Jacobi method
```

```
%-----
```

```
xnew=x0;
```

```
error=1;
```

```
while error>tol
```

```
    xold=xnew;
```

```
    for i=1:length(xnew)
```

```

off_diag = [1:i-1 i+1:length(xnew)];

xnew(i) = 1/A(i,i)*( b(i)-sum(A(i,off_diag)*xold(off_diag)) );

end

error=norm(xnew-xold)/norm(xnew);

end

x_jacobian=xnew

%Gauss?Seidel:

%-----

maxiter=10;

lambda=1;

n=length(x0);

x=x0;

error=1;

iter = 0;

while (error>tol & iter<maxiter)

    xold=x;

    for i=1:n

        I = [1:i-1 i+1:n];

```

```

x(i) = (1-lambda)*x(i)+lambda/A(i,i)*( b(i)-A(i,I)*x(I) );

end

error = norm(x-xold)/norm(x);

iter = iter+1;

end

x_siedal=x

%SOR

%-----

lambda=1.3;

n=length(x0);

x=x0;

error=1;

iter = 0;

while (error>tol & iter<maxiter)

    xold=x;

    for i=1:n

        I = [1:i-1 i+1:n];

        x(i) = (1-lambda)*x(i)+lambda/A(i,i)*( b(i)-A(i,I)*x(I) );

```


end

error = norm(x-xold)/norm(x);

iter = iter+1;

end

x_SOR=x

Appendix H

Matlab code for Conjugate Gradient method.

```
function [u, niter, flag] = solveCG(A, f, s, tol, maxiter)
```

```
% SOLVECG  Conjugate Gradients method.
```

```
%
```

```
%  Input parameters:
```

```
%      A : Symmetric, positive definite NxN matrix
```

```
%      f : Right-hand side Nx1 column vector
```

```
%      s : Nx1 start vector (the initial guess)
```

```
%      tol : relative residual error tolerance for break
```

```
%      condition
```

```
%      maxiter : Maximum number of iterations to perform
```

```
%
```

```
%  Output parameters:
```

```
%      u : Nx1 solution vector
```

```
%      niter : Number of iterations performed
```

```
%      flag : 1 if convergence criteria specified by TOL could
```

```
%      not be fulfilled within the specified maximum
```

```
%      number of iterations, 0 otherwise (= iteration
```

```
%      successful).
```

```
A=[7/2 -2 0 -1 0 0; -1 4 -1 0 -1 0; 0 -1 4 0 0 -1; -1 0 0 7/2 -2 0; 0 -1 0 -1 4 -1; 0 0 -1 0 -1 4]
```

```
f=[29/16; 5/8; 23/16; 11/8; 3/4; 13/8];
```

```
s=[0;0;0;0;0;0]
```

```

maxiter = 6

u = s;      % Set u_0 to the start vector s

r = f - A*s; % Compute first residuum

p = r;

rho = r'*r;

niter = 0;  % Init counter for number of iterations

flag = 0;  % Init break flag

% Compute norm of right-hand side to take relative residuum as
% break condition.

normf = norm(f);

if normf < eps % if the norm is very close to zero, take the
% absolute residuum instead as break condition
% ( norm(r) > tol ), since the relative
% residuum will not work (division by zero).

warning(['norm(f) is very close to zero, taking absolute residuum' ...
' as break condition.']);

normf = 1;

end

while (norm(r)/normf > 0.00001) % Test break condition

a = A*p;

alpha = rho/(a'*p);

u = u + alpha*p;

r = r - alpha*a;

rho_new = r'*r;

```

```
p = r + rho_new/rho * p;  
rho = rho_new;  
niter = niter + 1;  
if (niter == maxiter)      % if max. number of iterations  
flag = 1;                  % is reached, break.  
break  
end  
end
```

جامعة النجاح الوطنية
كلية الدراسات العليا

طريقة الفروق المحدودة والعناصر المحدودة لحل المعادلات التفاضلية الجزئية الناقصة

اعداد

مالك أبو الرب

اشراف

أ.د ناجي قطناني

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الرياضيات بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس - فلسطين.

2016

ب

طريقة الفروق المحدودة والعناصر المحدودة لحل المعادلات التفاضلية الجزئية الناقصة

إعداد

مالك أبو الرب

إشراف

أ.د. ناجي قطناني

المخلص

كثيراً من الظواهر الفيزيائية والهندسية الطبيعية لا تظهر إلا على شكل أنظمة رياضية وتحديداً تظهر كمعادلات تفاضلية جزئية تصف طبيعة هذه الظواهر. في هذه الرسالة، استخدمنا المعادلات التفاضلية الجزئية الخطية الناقصة من الرتبة الثانية بحيث تم التركيز على معادلة بواسون ومعادلة لابلاس في البعد الثاني كنموذج لوصف تلك الظواهر. باستخدام طريقة الفروق المحدودة والعناصر المحدودة لتقريب حل تلك المعادلات يتم تحويل المعادلة إلى شكل آخر بحيث يتم الحصول في النهاية على نظام خطي من المعادلات يمكن حله باستخدام طرق تكرارية مثل:

Jacobi method, Gauss-Seidel method, Successive over Relaxation (SOR) method and Conjugate Gradient method.

وجدنا من خلال هذا البحث أن طريقة الفروق المحدودة أفضل من طريقة العناصر المحدودة للحصول على حل مُقربٍ للمعادلة وبأقل خطأ ممكن في حالة كون المجال (منطقة الحل) لمعادلة لابلاس أو لمعادلة بواسون ذات أشكال هندسية منتظمة (مثلث، مستطيل، ...). ولحل النظام الخطي الناتج من تجزئة معادلة لابلاس أو معادلة بواسون، وجدنا أن طريقة SOR هي أفضل طريقة تكرارية من بين الطرق التكرارية الأخرى للحصول على حل مُقربٍ للحل الدقيق والتي تعطي أقل خطأ ممكن.