Numerical solutions of systems of linear equations[1]

V. B. Yap[2], Q. Sheng[3]

1. Introduction

A vector is a collection of items. A set of vectors with certain properties, such as with the same number of items, forms a vector space. R^n is the vector space wherein the vectors have n real items each. If the items are complex numbers, C^n is used to denote the vector space. $R^n \subset C^n$, *i.e.*, R^n is a subspace of C^n. For $x, y \in C^n$, define $z = x \pm y$ as

$$z = \begin{bmatrix} x_1 \pm y_1 \\ \vdots \\ x_n \pm y_n \end{bmatrix} \text{ or } z_i = x_i \pm y_i, \quad i = 1, \ldots, n$$

and $z \in C^n$.

A matrix is a rectangular array of numbers. Let A be a matrix of size $n \times m$, *i.e.*, n rows by m columns, then

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \text{ or } [a_{ij}]_{i=1,2,\ldots,n;\ j=1,2,\ldots,m}$$

where i and j are known as the row and column number respectively. For a square matrix of size $n \times n$, $a_{11}, a_{22}, \ldots, a_{nn}$ are known as the diagonal entries. $R^{n \times n}$ is the matric space containing $n \times n$ matrices with real numbers and $C^{n \times n}$, complex numbers. For $A, B \in C^{n \times n}$, define $P = A + B$ and $Q = AB$ as

$$P = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} + b_{n1} & \cdots & a_{nn} + b_{nn} \end{bmatrix} \text{ or } p_{ij} = a_{ij} + b_{i,j}, \quad i, j = 1, 2, \ldots, n,$$

[1]Paper presented to the Science Research Congress 1991 as part of the 1991 SRP organised jointly by the National University of Singapore and the Ministry of Education.

[2]Raffles Junior College, Mount Sinai Road, Singapore 1027.

[3]Department of Mathematics, National University of Singapore, Kent Ridge, Singapore 0511.

$$Q = \begin{bmatrix} \sum_{k=1}^{n} a_{1k}b_{k1} & \cdots & \sum_{k=1}^{n} a_{1k}b_{kn} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{nk}b_{k1} & \cdots & \sum_{k=1}^{n} a_{nk}b_{kn} \end{bmatrix} \text{ or } q_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}, \quad i,j = 1,\ldots,n,$$

and $P, Q \in C^{n\times n}$. Generally $AB \neq BA$. For other matrices, AB is defined if and only if the number of columns of A equals that of rows of B. A square matrix A is singular if and only if $|A| = 0$. A vector is a special matrix with size $n \times 1$.

Consider the system of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + \cdots\cdots\cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\quad \ddots \qquad\quad \vdots \qquad\quad &\vdots \\ a_{n1}x_1 + \cdots\cdots\cdots + a_{nn}x_n &= b_n \end{aligned}$$

where a_{ij} and b_i are real constants and x_i are unknowns $(i,j = 1,2,\ldots,n)$. The above linear system can be written in an equivalent matrix form:

$$Ax = b, \quad A \in R^{n\times n}, \; x, b \in R^n,$$

where

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

Systems of linear equations have a wide range of applications in both theoritical and practical sciences. Our study attempts to give a brief introduction to the numerical solutions of the linear systems together with some important theorems in linear algebra. Several algorithms for solving linear systems are developed using Fortran 77. Partial pivoting is introduced to achieve accuracy and applicability. LU decomposition is studied and the operation count for Gaussian elimination is given and compared to other methods. The programs are then tested with an ill-conditioned system and a nuclear reactor problem. Errors in the discretizations are estimated.

2. Analysis and Algorithms

(a) *Preliminary and Analysis*

Let A be an $n \times m$ matrix. The transpose of A, denoted A^T, is obtained by making the ith row of A the ith column of A^T and is given by

$$a_{ij}^T = a_{ji}.$$

The conjugate transpose of A, denoted A^*, is obtained in a similar way and is given by

$$a_{ij}^* = \bar{a}_{ji}.$$

Thus A^T and A^* are $m \times n$ matrices. We may prove

Lemma 2.1. For $A, B \in C^{n \times n}$

(*a*) $A+B = B+A$, (*b*) $(A+B)+C = A+(B+C)$, (*c*) $A(B+C) = AB + AC$;

(*d*) $A(BC) = (AB)C$, (*e*) $(A+B)^T = A^T+B^T$, (*f*) $(AB)^T = B^T A^T$.

A is called symmetric if $A^T = A$, and Hermitian if $A^* = A$. The term symmetric is generally used only with real matrices. Let A be a Hermitian matrix. It is called positive definite if and only if $x^T Ax > 0$, $\forall x \in C^n, x \neq 0 = [0, 0, \ldots, 0]^T$. Positive definite matrices possess important properties and occur in a wide variety of applications.

A norm of a vector x is a function $f : R^n \mapsto R$ which gives the size of the vector and is denoted by $\|x\|$. A useful class of norms are the Hölder or p-norms defined by

$$\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}} \quad p \geq 1.$$

Of which $\|x\|_1$, $\|x\|_2$ and $\|x\|_\infty$ are the most important. Similar norms can be defined for matrices. We have

Theorem 2.1. If $A \in R^{n \times n}$, the following statements are equivalent.

(*a*) $A = MM^T$ with $M \in R^{n \times n}$ and $|M| \neq 0$.

(*b*) A is positive definite.

(*c*) There exists a lower triangular matrix L with $L \in R^{n \times n}$ such that $A = LL^T$.

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{bmatrix}$$

Proof. Let $M = \{m_{ij}\}, A = \{a_{ij}\}, \quad i,j = 1,2,\ldots,n$. Then $M^T = \{m_{ji}\}$.

$$a_{ij} = \sum_{k=1}^{n} m_{ik} m_{kj}^T = \sum_{k=1}^{n} m_{ik} m_{jk},$$

$$a_{ji} = \sum_{k=1}^{n} m_{jk} m_{ki}^T = \sum_{k=1}^{n} m_{jk} m_{ik}.$$

Therefore A is symmetric. Furthermore, we have

$$x^T A x = x^T M M^T x$$

$$= (M^T x)^T M^T x$$

$$= \|M^T x\|_2^2 > 0, \quad x \in R^n, x \neq 0.$$

Therefore A is positive definite. □

Corollary 2.1. If $A \in C^{n\times n}$ is positive definite, then $a_{ij} > 0$ for $i = 1,2,\ldots,n$.

(b) *Algorithm I: Gaussian Elimination*
Consider a real linear system

$$Ax = b, \quad A \in R^{n\times n}, \quad x,b \in R^n, \quad |A| \neq 0.$$

We wish to reduce A to an upper triangular matrix which is easier to solve. The process of reducing A and subsequently solving for x is known as Gaussian elimination. Denote A by $A^{(1)}$, we have

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

Step 1: Assuming $a_{11}^{(1)} \neq 0$, for $i = 1,2,\ldots,n$, define row multipliers

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}},$$

and define

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, \quad j = 2,3,\ldots,n,$$

$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}.$$

Leaving the fisrt row undisturbed and setting the first column below $a_{11}^{(1)}$ to zeroes, we obtain the system $A^{(2)}x = b^{(2)}$ which looks like

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ \vdots \\ b_n^{(1)} \end{bmatrix}.$$

Ignore the first row and column and repeat the process for the submatrices until step $(n-1)$ is finished. To show the general case, after $(k-1)$ steps, $k = 2, 3, \ldots, n$, we have $A^{(k)}x = b^{(k)}$,

$$\begin{bmatrix} a_{11}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & & \vdots & \\ \vdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_k^{(k)} \end{bmatrix}.$$

Step k: Assuming $a_{kk}^{(k)} \neq 0$, for $i = k+1, k+2, \ldots, n$, define

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \tag{2.1}$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad j = k+1, k+2, \ldots, n, \tag{2.2}$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}. \tag{2.3}$$

After $(n-1)$ steps, the sysem $A^{(n)}x = b^{(n)}$ looks like

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_{n-1}^{(n-1)} \\ b_n^{(n)} \end{bmatrix}$$

which is quite easy to solve by back substitution. First

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}, \tag{2.4}$$

then

$$x_k = \frac{1}{a_{kk}^{(k)}} \left(b_k^{(k)} - \sum_{i=k+1}^{n} a_{ki}^{(k)} x_i \right), \quad k = n-1, n-2, \ldots, 1. \tag{2.5}$$

This completes the algorithm. Equations (2.1) to (2.5) are used to write a Fortran program (Program 1) which performs the Gaussian elimination. It is to be noted that Gaussian elimination algorithms are not unique. There are other ways to reduce A to an upper triangular or even a lower triangular matrix. In that case, forward substitution is used. Regardless of the different methods used, the solutions should be identical theoritically. However, from the point of view of numerical analysis, different computers and programs may give different answers for the same problem. We will see this in the next section.

(c) *Algorithm II: LU Decomposition*

Construct the matrices L and U where

$$L = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ m_{21} & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & 1 & 0 \\ m_{n1} & \cdots & \cdots & m_{n,n-1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & \cdots & 0 & u_{nn} \end{bmatrix} = A^{(n)}.$$

This is called the LU decomposition of A which is very useful as L and U can be stored together as a matrix(ignoring the diagonal entries of L) in the computer. If A appears in other systems, then we need only to modify b using the row multipliers to obtain the solution.

Lemma 2.2. $A = LU$ and $|A| = u_{11}u_{22}\ldots u_{nn}$.

Proof. For $j \geq i$,

$$\begin{bmatrix} m_{i1}, \ldots, m_{i,i-1}, 1, 0, \ldots, 0 \end{bmatrix} \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sum_{k=1}^{i-1} m_{ik} u_{kj} + u_{ij}$$

$$= \sum_{k=1}^{i-1} m_{ik} a_{kj}^{(k)} + a_{ij}^{(i)} = \sum_{k=1}^{i-1} \left(a_{ij}^{(k)} - a_{ij}^{(k+1)} \right) + a_{ij}^{(i)}$$

$$= a_{ij}^{(1)} = a_{ij}.$$

For $i > j$,

$$\left[m_{i1}, \cdots, m_{i,i-1}, 1, 0, \cdots, 0 \right] \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sum_{k=1}^{j-1} m_{ik} u_{kj} + m_{ij} u_{jj}$$

$$= \sum_{k=1}^{j-1} m_{ik} a_{kj}^{(k)} + m_{ij} a_{jj}^{(j)} = \sum_{k=1}^{j-1} \left(a_{ij}^{(k)} - a_{ij}^{(k+1)} \right) + a_{ij}^{(j)}$$

$$= a_{ij}^{(1)} = a_{ij}.$$

Therefore $A = LU$. It follows that

$$|A| = |L||U|.$$

But $|L| = 1$, therefore

$$|A| = |U| = u_{11} u_{22} \ldots u_{nn}. \quad \square$$

(d) *Operation Count For Gaussian Elimination*

At STEP 1, $(n-1)$ divisions are used to evaluate the row multipliers. Then $(n-1)^2$ multiplications and $(n-1)^2$ subtractions are required to give $A^{(2)}$. Also $(n-1)$ multiplications and $(n-1)$ subtractions are needed to modify $b^{(1)}$ to $b^{(2)}$. Let $MD(\cdot)$ and $AS(\cdot)$ denote the number of multiplications and divisions and the number of additions and subtractions respectively, we can sum up the total number of operations. Since $(n-1)$ steps are needed, and in each successive step the side of the current submatrix shrinks by 1 unit, we have

$$AS(A) = \sum_{r=1}^{n-1} r^2 \quad MD(A) = \sum_{r=1}^{n-1} \left(r^2 + r \right);$$

$$AS(b) = \sum_{r=1}^{n-1} r \quad MD(b) = \sum_{r=1}^{n-1} r;$$

$$AS(A) + AS(b) = \sum_{r=1}^{n-1} r^2 + \sum_{r=1}^{n-1} r = \frac{2n^3 - 2n}{6} \simeq \frac{n^3}{3},$$

$$MD(A) + MD(b) = \sum_{r=1}^{n-1} r^2 + 2\sum_{r=1}^{n-1} r = \frac{2n^3 + 3n^2 - 5n}{6} \simeq \frac{n^3}{3}.$$

The number of additions is almost the same to that of multiplications and divisions, so we consider only the latter. Gaussian elimination is less troublesome than multiplying two $n \times n$ matrices which requires n^3 operation. It is also better than the Cramer's rule by which the operation count is $(n+1)!$ if the determinants are computed using expansion by minors or the Gauss-Jordan method which requires $\frac{1}{2}n^3$ operations.

(e) *Algorithm III: Partial Pivoting*

In the algorithm, we assume that the pivot element $a_{kk}^{(k)} \neq 0, k = 1, 2, \ldots, n$. This assumption can be removed by switching the rows such that the pivot element is always nonzero. But in computer operations, a zero may not be exactly zero due to errors, and if we happen to make this 'nonzero' the pivot, gross error will result. Partial pivoting is a technique designed to avert this danger.

Define

$$s_i = \max_{k \leq i \leq n} |a_{ik}^{(k)}|, \quad k = 1, 2, \ldots, n.$$

If $s_i \neq |a_{kk}^{(k)}|$, interchange the ith row and the kth row in A and B and it is certain that the pivot element will not be zero, for if $a_{ik}^{(k)} = 0, |A| = 0$, contrary to the condition of A. Note that by doing so, $|m_{ik} \leq 1|$ for $i = k+1, k+2, \ldots, n$, and this will prevent the elements from being magnified in subsequent elimination which will cause severe loss of significant figures in the floating point representation of numbers in the computer. In Program 2, partial pivoting is used.

3. Applications

(a) *An Ill-conditioned (Stiff) Problem*

Given the system

$$6x_1 + 2x_2 + 2x_3 = -2;$$

$$2x_1 + \frac{2}{3}x_2 + \frac{1}{3}x_3 = 1;$$

$$x_1 + 2x_2 - x_3 = 0.$$

Solving without partial pivoting using Program 1, the solution,

$$x_1 = -3.97682137 \times 10^7, \quad x_2 = 1.19304647 \times 10^8, \quad x_3 = -5.00000000,$$

is disastrous. But with partial pivoting (Program 2), the approximately correct solution is found:

$$x_1 = 2.60000006, \quad x_2 = -3.79999990, \quad x_3 = -4.99999988.$$

Using another version of the Gaussian elimination (Program 3) without partial pivoting, we get the wrong answers:

$$x_1 = 1.33333333, \quad x_2 = 0.00000000, \quad x_3 = -5.00000000.$$

With partial pivoting, the solution is even closer to the true one:

$$x_1 = 2.60000006, \quad x_2 = -3.80000000, \quad x_3 = -5.00000000.$$

This type of problem is called ill-conditioned or stiff. The slight difference in the two correct solutions is due to computer round-off error. It is stressed that by doing hand calculations, we get the exact solution with or without partial pivoting.

(b) *A Nuclear Reactor Problem*

In a unit square area Ω of a nuclear reactor, the pressure function U satisfies the partial differential equation

$$\Delta U - aU = 0 \tag{3.1}$$

where

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \text{ and } a = \sin^3(x+y),$$

with boundary functions given as follows:

$A : U = P = (1-y)^2 y^2$, if $x = 0, 0 \le y \le 1$;

$B : U = Q = \sin^3 4\pi x$, if $0 \le x \le 1, y = 1$;

$C : U = R = xy(1-y)$, if $x = 1, 0 \le y \le 1$;

$D : U = S = \sin 2\pi x$, if $0 \le x \le 1, y = 0$. We wish to find the numerical values of U for points in Ω. Let $N \ge 1 (N \in Z_+)$, define

$$h = \frac{1}{N+1}, \tag{3.2}$$

$$(x_j, y_k) = (jh, kh) \in \Omega, \quad j,k = 1,2,\ldots,N.$$

These are called grid points or mesh points and they are labelled in the order shown in the sketch. Let the value of U at point n ($n = 1,2,\ldots,N^2$) be U_n. We approximate equation (3.1) by means of the finite difference method. Using central differences, we have

$$\frac{\partial^2 U}{\partial x^2} \simeq U_{\hat{x}\hat{x}} = \frac{U(x+h,y) - 2U(x,y) + U(x-h,y)}{h^2},$$

$$\frac{\partial^2 U}{\partial y^2} \simeq U_{\hat{y}\hat{y}} = \frac{U(x,y+h) - 2U(x,y) + U(x,y-h)}{h^2}.$$

Substituting the above expressions into (3.1), we get the difference equation

$$U(x+h,y)+U(x-h,y)+U(x,y+h)+U(x,y-h)-(4+ah^2)U(x,y) = 0. \tag{3.3}$$

For the point $(x_j, y_k) \in \Omega$, from equation (3.3),

$$U(x_{j+1},y_k)+U(x_{j-1},y_k)+U(x_j,y_{k+1})+U(x_j,y_{k-1})-(4+ah^2)U(x_j,y_k) = 0. \tag{3.4}$$

This will be used to find the equation on each grid point. For example, at point 1, $j = k = 1$, equation (3.4) becomes

$$U(x_2,y_1) + U(x_0,y_1) + U(x_1,y_2) + U(x_1,y_0) - (4+ah^2)U(x_1,y_1) = 0$$

which gives

$$U_2 + P(0,h) + U_{N+1} + S(h,0) - (4+ah^2)U_1 = 0.$$

Bringing the constants to the right, we obtain the first linear equation. Similarly for the remaining grid points. Arranging the N^2 equations in N^2 unknowns into a system according to the order of grid points, we can solve it using Gaussian elimination. Let us call the system $AU = b$. A data file is built using Program 4 to generate the entries of A, then Program 1 or 2 is used to solve the system and the numerical solutions are satisfactory.

We observed that the matrix A has several interesting properties:

(a) A is called sparse because most of its elements are zeroes.

(b) Nonzeroes can only be found in five diagonal lines, so A is called a band matrix.

(c) It is called block tridiagonal because if A were partitioned into N^2 $N \times N$ matrices, most nonzeroes will be found in blocks along the three main diagonal lines.

(d) It is symmetric and positive definite and by lemma 2.4, a lower triangular matrix E exists such that $A = EE^T$.

(e) It is diagonally dominant, meaning the absolute values of the diagonal elements are numerically larger than others in their correponding rows and columns.

(f) The matrices L and U obtained by LU decomposition are also sparse.

This type of matrix has important applications in solving many practical problems.

(c) *Truncation Error In The Discretization*

Using Taylor's expansion, we know that the error in $U_{\hat{x}\hat{x}}$,

$$err_x = \left| \frac{\partial^2 U}{\partial x^2} - U_{\hat{x}\hat{x}} \right|$$

$$= \frac{2h^2}{4!}\frac{\partial^4 U}{\partial x^4} + \cdots + \frac{2h^{2n-2}}{(2n)!}\frac{\partial^{2n} U}{\partial x^{2n}} + \cdots.$$

Similarly error in $U_{\hat{y}\hat{y}}$,

$$err_y = \frac{2h^2}{4!}\frac{\partial^4 U}{\partial y^4} + \cdots + \frac{2h^{2n-2}}{(2n)!}\frac{\partial^{2n} U}{\partial y^{2n}} + \cdots.$$

Define

$$\Delta_n = \frac{\partial^n}{\partial x^n} + \frac{\partial^n}{\partial y^n}.$$

Total error

$$= err_x + err_y = \frac{2h^2}{4!}\left(\frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4}\right) + \cdots + \frac{2h^{2n-2}}{(2n)!}\left(\frac{\partial^{2n} U}{\partial x^{2n}} + \frac{\partial^{2n} U}{\partial y^{2n}}\right) + \cdots$$

$$= 2\sum_{k=2}^{\infty} \frac{h^{2k-2}}{(2k)!}\Delta_{2k} U.$$

As N becomes larger, h becomes smaller from equation (1), and the approximation becomes more accurate.

References

1. Kahaner, D.; Moler, C. & Nash, S. (1989): *Numerical Methods and Software,* Prentice-Hall International Editions.

2. Golub, G. H. (1983): *Matrix Computations.* The Johns Hopkins University Press.

3. Balfour, A. & Marwick D. H. (1983): *Programming in Standard Fortran 77,* Heinemann Educational Books.

4. Sheng, Q. (1989): "Solving Linear Partial Differential Equation by Exponential Splitting", *IMA J. of Numerical Analysis,* **9,** pp 199-212.

5. Sheng, Q. (1991): *Lecture Notes for Numerical Analysis.* NUS.