DOI: 10.51984/SUCP.V4I1.3922

المؤتمر العلمي الأول للتطبيقات الهندسية (ICEA'2024)

The First Scientific Conference on Engineering Applications (ICEA'2024)

Conference homepage: www.icea.ly

Teaching the Principle Jacobi Method Using MATLAB

*Eman Daman[a] and Laila gerazi[b]

[a] General Department of Physics, College Of Engineering Technology, Janzour, Tripoli, Libya.
[b] General Department of Mathematics, College Of Engineering Technology, Janzour, Tripoli, Libya.

Keywords:

Iterativ methodse
Jacobi Method
linear system
MATLAB
The matrices

ABSTRACT

This paper presents a study of the effectiveness of the MATLAB program in solving linear equations using iteration methods for numerical analysis, as the numerical analysis of some linear equations requires many complex operations to analyse the approximate solutions, and this leads to many computational errors, and great effort to solve them. The importance of this study lies in shedding light on the MATLAB program and its strength in solving linear equation problems and helping students solve complex equations faster and more accurately. The aim of this study is to know one of the iterative methods represented by the Jacobi method, to solve systems of linear equations, using the MATLAB program to calculate the values, where the solution is done by guessing a specific solution, which is improved in successive steps until the results become close with the progress of the steps, which is the final solution for a set of linear equations, as we changed the error value by adding and subtracting 50%, and we noticed an increase in the number of iterations and a decrease in the error, which led to obtaining satisfactory results.

تعليم مبدأ جاكوبي بإستخدام الماتلاب

*إيمان دامان[1] و ليلى القرازي[2]

[1] القسم العام فيزياء، كلية التقنية الهندسية جنزور، طرابلس ليبيا.
[2] القسم العام رياضيات، كلية التقنية الهندسية جنزور، طرابلس، ليبيا.

الكلمات المفتاحية:

الطرق التكرارية
النظام الخطي
المصفوفات
الماتلاب
طريقة جاكوبي.

الملخص

تقدم هذه الورقة دراسة فعالية برنامج MATLAB المتمثلة في حل المعادلات الخطية بإستخدام طرق التكرار للتحليل العددي، حيث يتطلب التحليل العددي لبعض المعادلات الخطية الكثير من العمليات المعقدة لتحليل الحلول التقريبية، ويؤدي هذا الى العديد من الاخطاء الحسابية، وبذل جهد كبير لحلها، تكمن أهمية هذه الدراسة في القاء الضوء على برنامج الماتلاب وقوته في حل مسائل المعادلات الخطية ومساعدة الطلاب في حل المعادلات المعقدة بشكل أسرع واكثر دقة، والهدف من هذه الدراسة معرفة أحد الطرق التكرارية والمتمثلة في طريقة جاكوبي، لحل أنظمة من المعادلات الخطية، بإستخدام برنامج MATLAB لحساب القيم، حيث يتم الحل عن طريق تخمين حل معين، يتم تحسينه في خطوات متتالية حتى تصبح النتائج متقاربة مع تقدم الخطوات، وهو الحل النهائي لمجموعة من المعادلات الخطية، حيث قمنا بتغيير قيمة الخطأ بإضافة وطرح 50%، ولاحظنا زيادة عدد التكرارات وإنخفاض الخطأ، مما أدى إلى الحصول على نتائج مرضية.

1. Introduction

The system of linear equations arises in many scientific fields, whether applied or theoretical, and sometimes produces large systems that cannot be dealt with manually, so it was necessary to use the computer, as these systems can be written in the form of matrices, which made the matrix an important tool with a prominent role, as it is a common factor in many physical, mathematical and other applications [1], and the MATLAB program is considered one of the important multi-use application programs, as the word MATLAB is originally taken from the expression (matrix laboratory) because the basic data unit in it is the matrix, and one of its most important features is its ability to draw curves and three-dimensional spatial shapes, and the iterative methods (Jacobi method) are among the easiest methods of numerical analysis used in solving systems of linear equations [1],[2], the researcher relies on the descriptive method in formulating the axes and hypotheses, and searching for the opinions of researchers related to the research topic, and on the analytical method in choosing research hypotheses and achieving its objectives, in mathematical issues that were chosen to address the research problem in its applied framework, and to find approximate solutions for such systems, especially when direct methods become impractical and expensive Mathematically, most

*Corresponding author:
E-mail addresses: Eman.daman@gmail.com, (L. Gerazi) Laila.gerazi@gmail.com
Article History : Received 25 March 2024 - Received in revised form 17 September 2024 - Accepted 15 October 2024

numerical analysis methods and algorithms require a matrix in one Way or another, if we consider the linear system [3], [4]:

$$A\,X = B \tag{1}$$

This equation can be written in the form of matrices as follows:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2}$$

Where A is the coefficient matrix of the linear system, X is the matrix of unknowns, B is the matrix of absolute terms We divide the coefficient matrix A into three parts, the purpose of which is to facilitate the process of finding the solution: D is a diagonal matrix, L is a lower triangular matrix, and U is an upper triangular matrix, so that [5], [6]:

$$A = D + L + U \tag{3}$$

Where:

$$D = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

To find the approximate solution of the linear system which is presented by equation (1), as follows:

$$(D + L + U)X = B \tag{4}$$
$$DX = B - LX - UX$$

Here we give an estimated value to the vector X^k on the right side, in order to obtain a new value on the left side, so the equation (4) becomes in iterative form [6], [7] :

$$X^{(k+1)} = D^{-1}B = D^{-1}(L + U)X^k, \quad k = 0, 1, 2, \dots \tag{5}$$

Therefore, we obtain the Jacobi formula as follows:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{n} a_{ij}\, x_j^{(k)} \right]; \quad k = 1, 2, \dots \tag{6}$$

This process is repeated until the following condition is met [7], [8]:

$$max_i \left| x_i^{(k+1)} - x_i^k \right| \le \varepsilon$$

For each i, where ε is the given amount of error.

In this work, it will give two examples for Jacobi method which are in the following points:

A) Example: use Jacobi's method to solve the system of equations, where the zero vector is assumed as the initial approximation.

$$\begin{aligned} 3x + y + z &= 1 \\ 2x + 5y - z &= 2 \\ x + 2y + 6z &= 0 \end{aligned} \tag{7}$$

Solution:

Equation (7) can be written using Jacobi method as follows:

$$\begin{aligned} x &= \tfrac{1}{3}(1 - y - z) \\ y &= \tfrac{1}{5}(2 - 2x + z) \\ z &= \tfrac{1}{6}(-x - 2y) \end{aligned} \tag{8}$$

To solution equation (8), method program is written to gets values x, y and z, (see appendix A).

In this example, The error value was forced when $e =$ [0.1 0.2 0.3 0.4 0.5] , Then the error was changed by an increase or decrease of 50%, to obtain numerical results for the number of iterations. The comparison between the error and the number of iterations appears in Table I.

Table I: shows the repeatability and error values

0%		+ 50 %		- 50%	
e	*it*	*e*	*it*	*e*	*it*
0.1	*3*	*0.15*	*8*	*0.05*	*4*
0.2	*2*	*0.3*	*2*	*0.1*	*4*
0.3	*2*	*0.45*	*1*	*0.15*	*3*
0.4	*1*	*0.6*	*1*	*0.2*	*2*
0.5	*1*	*0.75*	*1*	*0.25*	*2*

The comparison between the number of iterations (IT) versus the error value (e), is shown in Figs 1,2 and 3. Starting from Fig. 1. it is noted that the greater the number of iterations, the lower the error value, when the error was 0.1, the number of iterations reached 3, but when the error was 0.4 and 0.5, the number of iterations was 1.

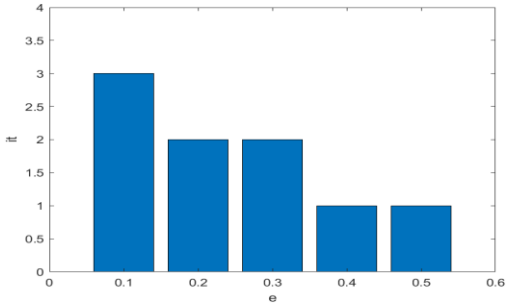

Fig .1: the shape of example A where e = [0.1 0.2 0.3 0.4 0.5].

Fig. 2. when the error increased by 50%, It is noticeable a difference in the number of iterations. When the error was 0.15, the number of iterations reached 8, while with an error of 0.45 and above, the number of iterations reached 1.

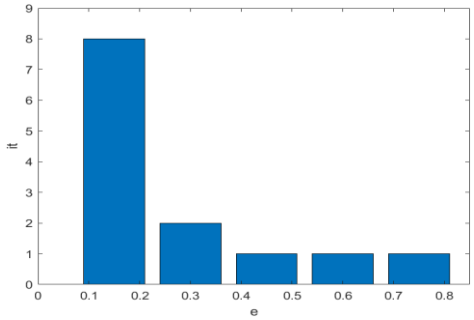

Fig .2: Increase error by 50%.

From Fig. 3. The number of iterations was equal when the error was 0.05 and 0.1, and it was the largest number of iterations, but when the error was 0.25, it gave the least number of iterations, meaning that the more the number of iterations, the smaller the error.

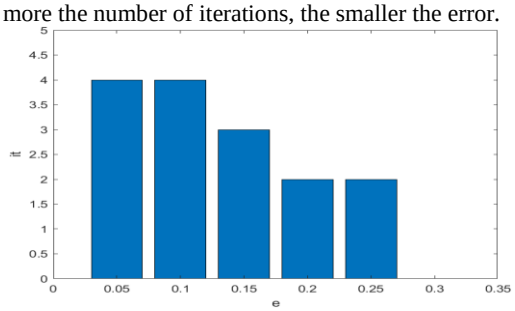

Fig.3: Reduce the error rate by 50%.

B) Example: use Jacobi's method to solve the system of equations. the zero vector is assumed as the initial approximation.

$$\begin{aligned} 4x + y - 2z &= 1 \\ x + 3y - z &= 8 \\ x - 7y + 10z &= 2 \end{aligned} \tag{9}$$

Solution:

According to Jacobi's method, the equations are as follows:

$$\begin{aligned} x &= \left(\frac{1}{4}\right)(1 - y - 2z) \\ y &= \left(\frac{1}{3}\right)(8 - x + z) \\ z &= \left(\frac{1}{10}\right)(2 - x + 7y) \end{aligned} \tag{10}$$

To see Matlab code for jacobi iterative method back to appendix B.

Perform the same previous steps in example A, From Table II, been noticed the numerical results of the archived simulation for the imposed error, with an increase and decrease of 50%.

Table II: shows the repeatability and error values

0 %		*+ 50 %*		*- 50 %*	
e	*it*	*e*	*it*	*e*	*It*
0.1	*8*	*0.15*	*8*	*0.05*	*12*
0.2	*6*	*0.3*	*6*	*0.1*	*9*
0.3	*6*	*0.45*	*5*	*0.15*	*8*
0.4	*6*	*0.6*	*5*	*0.2*	*7*

0.5	*5*	*0.75*	*4*	*0.25*	*7*

In Fig. 4, It is clear that the number of iterations is equal at when the error was 0.2, 0.3, and 0.4, then the number of iterations decreased as the error was larger.

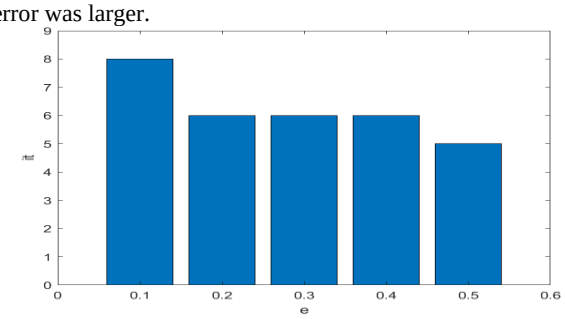

Fig .4: the shape of example (B) when e = [0.1 0.2 0.3 0.4 0.5].

As for Fig. 5. There is clearly a convergence between the number of iterations, and the smaller the error, the greater the number of iterations

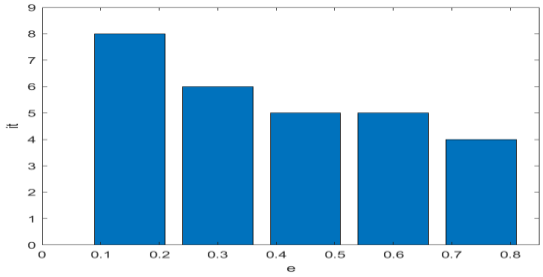

Fig .5: Increase error by 50%.

Finally: Fig. 6. The number of iterations reached 12 when the error was 0.05, and 7 when the error was 0.2, 0.25.

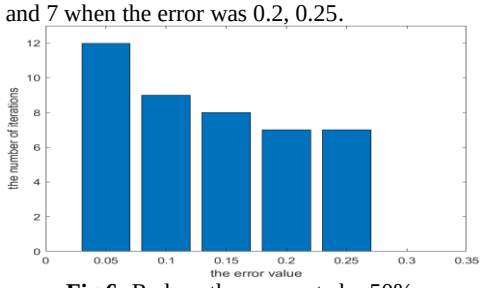

Fig.6: Reduce the error rate by 50%.

2. Conclusion:

Through our study of the Jacobi method using the MATLAB program, we noticed that this program combines theoretical understanding and practical application, as the Jacobi method is an effective tool for solving linear systems repeatedly, helping students gain a deeper insight into the principles of the MATLAB program to solve the problems of these systems. One of the advantages of the Jacobi method is that after each attempt, the error is calculated, and when the error reaches an appropriate value, the program stops, so it is possible to save a lot of time needed for the solution. One of its disadvantages is that it may not reach a solution, meaning that the solutions diverge when starting with a specific initial solution, and with another initial solution, the solutions will converge and the method will reach a final solution. In this research, we noticed that all the figures indicate an inverse relationship between the number of repetitions and the error value, meaning that the greater the number of repetitions, the lower the error rate and we reach a solution faster.

3. Appendix A

```
clear all
clc
x0=input('x0=');
y0=input('yo=');
z0=input('z0=');
format long
for i=1:100
   i
   x1=(1/3)*(1-y0+z0)
   y1=(1/5)*(2-2*x0+z0)
   z1=(1/6)*(-x0 -2*y0)
   if abs(x1-x0)<=0.001 & abs(y1-y0)<=0.001 & abs(z1-z0)<=0.001
        disp('the solution is x,y,z=')
        x1
        y1
        z1
         break
       else
              x0=x1;,y0=y1;,z0=z1;
       end
end
```

4. Appendix B

```
clear all
clc
x0=input('x0=');
y0=input('yo=');
z0=input('z0=');
format long
for i=1:100
   i
   x1=(1/4)*(1-y0+2*z0)
   y1=(1/3)*(8-x0+z0)
   z1=(1/10)*(2-x0 +7*y0)
   if abs(x1-x0)<=0.001 & abs(y1-y0)<=0.001 & abs(z1-z0)<=0.001
        disp('the solution is x,y,z=')
        x1
        y1
        z1
         break
       else
              x0=x1;,y0=y1;,z0=z1;
       end
end
```

5. REFERENCES:

[1]- Mohit Tekriwal, Andrew W. Appel, Ariel E. Kellison, David Bindel & Jean-Baptiste, “Verified Correctness, Accuracy and Convergence of Stationary Iterative Linear Solver: Jacobi Method”, International Conference on Intelligent Computer Mathematics, pp206-221, August 2023.

[2]- T. K. Enyew[1], G. Awgichew, E.haile G. D. Abie, “Second Refinement of Generalized Jacobi iterative method for solving linear system equations”, Journal of the Nigerian Mathematical Society, pp. 117-133, 2020.

[3]- Jeffrey R. chasov, “Numerical Method for Engineers”, Lecture Notes for Coursera, the hong kong university of science and technlogy, 2020.

[4]- Richard L. Burden, J. Douglas Faires, “Numerical Analysis”, Youngs state University, 2010.

[5]- H. Rutishauser, “ The Jacobi Method for Real symmetrices Numerische Mathematik”, 1-10, 1966.

[6]- Professor d. m. causon & Professor C. G. Mingham, “Introductory finite difference method for PDEs”, book 2010.

[7]- Mansour Rasoulzadeh Darabad, “ A fine-grained Approach to parallelise the Circuit simulation process on Highly parallel systems”, University of southampton, 2015.

[8]- Kevin Berwick, “Computational Physics using MATLAB”, west Lafayette, Indiana USA, 2012.