v. 4, n. 6, 2024

Journal of
Engineering
Research

USING MATLAB TO STUDY MATRICES AND LINEAR

Camila Fredegoto Santos
Universidade Tecnológica Federal de Paraná
Campo Mourão – Paraná
http://lattes.cnpq.br/5269307111893849

Wellington José Corrêa
Universidade Tecnológica Federal de Paraná
Campo Mourão – Paraná
http://lattes.cnpq.br/1045931096324971

Abstract: Solving linear systems is a fundamental task in several areas of mathematics and engineering, playing a crucial role in solving real-world problems. MATLAB, a powerful numerical computing platform, offers a wide range of tools and resources to efficiently address this type of problem. For the purposes of this article, we will explore how MATLAB can be used to solve linear systems. Additionally, MATLAB offers the ability to work with matrices, making it particularly suitable for dealing with systems of linear equations represented in this form. Through practical examples, we will illustrate how to use MATLAB functionalities to solve linear systems of different sizes and complexities. Finally, we will highlight the advantages of using MATLAB to solve linear systems, including its computational efficiency, ability to deal with large and complex systems, and the ease of visualizing results, making it a great tool to help students in their studies and research.
Keywords: linear algebra; MATLAB; linear systems.

INTRODUCTION

Linear systems have a wide range of applications in mathematics, physics, engineering, economics, computer science and many other disciplines. They are used to model and solve real-world problems such as electrical circuits, structural analysis, production planning, and more.

A system of n linear equations is described as:

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

where a_{ij}, $b_i \in R$. Denoting the matrices: A = (a_{ij}) and B = (b_i) para $1 \leq i, j \leq n$, we can rewrite the system Equation (1) as:

$$A \,.\, X = B \quad (2)$$

Numerical methods for solving systems of linear equations are divided into two groups:

(i) Exact Methods: are those that lead to the exact solution, minus rounding errors introduced by the machine, after a finite number of steps (operations). Its objective is to transform the linear system into an upper or lower triangular system (depending on the method adopted), making the resolution of the system or the calculation of the inverse matrix simpler to obtain.

(ii) Iterative Methods: these are methods that, from an initial approximation $x0$, allow obtaining an approximation sequence: x_0, x_1,..., x_k,... which converges to the solution, considering an a priori defined precision.

With regard to the exact methods described in item (i), we studied the following methods:

- The *LU* (Lower-Upper) factorization. It decomposes the matrix into two components, namely: lower triangular *L* and upper triangular *U*. Its application simplifies many resolutions of linear systems.

- LDL (Lower-Diagonal-Lower) factorization is another decomposition technique that represents a matrix as the product of three matrices: diagonal *D*, lower triangular matrix *L* (com a_{ii}=1) and the transpose of the lower triangular matrix: L^T. This factorization is especially efficient for solving symmetric linear systems, avoiding the need to calculate square roots, as in Cholesky factorization.

- Cholesky factorization, on the other hand, is a method for decomposing symmetric and positive matrices, and is also defined in two other matrices: lower triangular *L* and transposed L^T.

- Finally, partial pivoting is an important technique in Gaussian elimination and solving linear systems. This involves selecting the row with the largest absolute value (axis) in the current column and replacing that row with the current row before removing it. This improves numerical stability, reduces rounding errors and improves solution accuracy.

Regarding iterative methods (item (ii)), we studied the Gauss-Jacobi method and the Gauss-Seidel method. In the Gauss-Jacobi method, all variables are updated simultaneously after each iteration, based on the values of the variables in the previous iteration. In the Gauss-Seidel method, variables are updated one by one as they are calculated, based on the updated values already available.

It is worth mentioning that we also study various types of matrices, including tridiagonal matrices, which play an essential role in mathematical calculations. Tridiagonal matrices have a special structure with nonzero elements only on the three diagonals: main, immediately above, and immediately below. This concept is very useful in interpolation and discretization of partial differential equations. As this work focuses on the use of MATLAB using these methods, to study the theory and application of the methods mentioned above, we recommend the reader consult the following works: (BURDEN, R. L.; FAIRES; BURDEN, A. M., 2016), (DORNELLES FILHO, 2016), (FRANCO, 2006) among others.

MATERIALS AND METHODS

As mentioned previously, the scope of this work is to use MATLAB to solve linear systems using numerical methods described in (i) and (ii). According to (DORNELLES FILHO, 2016), MATLAB (acronym for Matrix Laboratory) is software that allows the user to perform calculations via direct typing of commands and also to build programs that automate more complex calculation procedures. MATLAB is a widely used tool in both the academic world (teaching, research, etc.) and the professional world (product development, problem analysis, etc.). It has a simple and intuitive interface, and is an indispensable tool for students of exact sciences and engineering.

For each of the methods exposed above, a file with the extension (. *m*) was generated. Such a file contains MATLAB code, which can include commands to perform calculations, process data, create graphs, and perform a variety of numerical tasks. They are essentially scripts that can be run in the MATLAB environment to perform specific tasks. containing the sequence of commands necessary for compilation. Specifically, in our work, codes for the numerical methods mentioned in the introduction were created using the pseudo-codes given in (BURDEN, R. L.; FAIRES; BURDEN, A. M., 2016). In some codes, the user simply needs to insert the matrices A and B to obtain the solution X of the system Eq. (2). In other situations, when executing the code, the user can choose to insert a file in the format (.DTA) containing, for example, the augmented matrix of the system Eq. (2) in which the code returns the solution of the aforementioned linear system.

Figures 1 and 2 refer to the implemented code of Cholesky factorization and Gaussian elimination with partial pivoting, respectively:

RESULTS AND DISCUSSIONS

The algorithms used in our study provided, as presented results, the outputs of the MATLAB implementations of the studied methods. Figures 3 and 4 give us the outputs of the implemented codes presented in Figures 1 and 2, respectively:

Note that in Figure 3, the code presents, in

```
clear;
 [ret, name] = system('hostname');
if ret ~= 0,
   if ispc
      name = getenv('COMPUTERNAME');
   else
      name = getenv('HOSTNAME');
   end
end
disp('data:');
disp(datetime('now'));
disp('Nome do computador:');
disp(name);
A = input('Insira as entradas da matriz A: por exemplo,  [4 2 -4;2 10 4; -4 4
9]\n ');
B=input('Insira a matriz B: por exemplo, [0; 6; 5]\n');
G=chol(A);
disp('A Matriz Triangular inferior G é');
disp(G);
disp('Solução do sistema linear G*Y=B:');
Y = G'\B;
disp(Y);
disp('Solução do sistema linear G^t*X=Y:');
X=G\Y;
disp(X);
```

Figure 1 –– Cholesky method

Source: Own authorship (2023)

```
clear;
 [ret, name] = system('hostname');
if ret ~= 0,
   if ispc
      name = getenv('COMPUTERNAME');
   else
      name = getenv('HOSTNAME');
   end
end
disp('data:');
disp(datetime('now'));
disp('Nome do computador:');
disp(name);
A = input('Insira as entradas da matriz A: por exemplo, [1 2 3; 3 1 0;0 3
4]\n');
B=input('Insira a matriz B: por exemplo, [3; 4; 3]\n');
N = length(B);
X= zeros(N,1);
Aug = [A B];
disp('Matriz ampliada');
disp(Aug);
for j=1:N-1
    %partial pivoting
    [M,P] = max(abs(Aug(j:N,j)));
    C=Aug(j,:);
    Aug(j,:) = Aug(P+j-1,:);
    Aug(P+j-1,:) = C;
    %echlon form
    for i=j+1:N
        m= -Aug(i,j)/Aug(j,j);
        Aug(i,:) = Aug(i,:)+ m*Aug(j,:);

    end
   fprintf('%3s\n','Forma escalonada do sistema usando  eliminação de Gauss com
pivotamento parcial');

fprintf('%3s\n','Matriz ampliada:');
Aug;
disp(Aug);
end


x=zeros(N,1);
for i= N:-1:1
    x(i) = (Aug(i,N+1)-Aug(i,1:N)*x)/Aug(i,i);
end
disp('A solução do sistema linear A*X=B é');
disp(x);
```

Figure 2 – Gaussian elimination with partial pivoting

Source: Own authorship (2023)

```
Command Window
>> Cholesky
data:
   19-Sep-2023 10:33:00

Nome do computador:
LAPTOP-OR9KOMMI

Insira as entradas da matriz A: por exemplo,  [4 2 -4;2 10 4; -4 4 9]
 [4 2 -4;2 10 4; -4 4 9]
Insira a matriz B: por exemplo, [0; 6; 5]
[0; 6; 5]
A Matriz Triangular inferior G é
     2     1    -2
     0     3     2
     0     0     1

Solução do sistema linear G*Y=B:
     0
     2
     1

Solução do sistema linear G^t*Y=X:
     1
     0
     1

fx >>
```

Figure 3 – Cholesky method

Source: Own authorship (2023)

```
Command Window
Matriz ampliada
     1     2     3     3
     3     1     0     4
     0     3     4     3

Forma escalonada do sistema usando  eliminação de Gauss com pivotamento parcial
Matriz ampliada:
    3.0000    1.0000         0    4.0000
         0    1.6667    3.0000    1.6667
         0    3.0000    4.0000    3.0000

Forma escalonada do sistema usando  eliminação de Gauss com pivotamento parcial
Matriz ampliada:
    3.0000    1.0000         0    4.0000
         0    3.0000    4.0000    3.0000
         0         0    0.7778         0

A solução do sistema linear A*X=B é
     1
     1
     0
fx
```

Figure 4 – Eliminação de Gauss com pivotamento parcial

Source: Own authorship (2023)

addition to the solution of the linear system Eq. (2), the lower triangular matrix *G* and the solutions of the triangular systems G Y=B and G^T X=Y. On the other hand, in Figure 4, the code provides, in addition to the solution, all matrices expanded with partial pivoting. In such situations, in addition to providing the response of the proposed linear system using a given method, such codes offer in a didactic way the resolution of the exercise, in which the user and graduate in some discipline that requires the solution of a linear system, can check a step intermediate to the system solution.

CONCLUSION

MATLAB proved to be an important tool in obtaining the solution of linear systems given in Eq. (2). Additionally, MATLAB allows educators to teach complex mathematical and scientific concepts in a more accessible and visual way. It can be used to illustrate concepts in algebra, calculus, statistics, differential equations, among others, making learning these topics more concrete and practical.

THANKS

I would like to thank my advisor, professor Dr. Wellington José Corrêa, for his support and contributions during the research project. I would also like to thank CNPq for the financial support.

CODE AVAILABILITY

A implementação computacional desenvolvida está disponível no Google Drive, por meio do link: https://drive.google.com/drive/folders/1BhyffsosOb6wW9ilPXXfXNUk-giuxa8y?usp=sharing.

REFERENCES

BURDEN, Richard L; FAIRES, J Douglas; BURDEN, Annette M. **Análise numérica**. [S.l.]: Cengage Learning, 2016.

DORNELLES FILHO, Adalberto Ayjara. **Fundamentos de cálculo numérico**. [S.l.]: Bookman Editora, 2016.

FRANCO, Neide Bertoldi. **Cálculo numérico**. [S.l.]: Pearson, 2006.