UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Civil, Arquitetura e Urbanismo

IC639: Métodos Numéricos para Engenharia Civil

List 3

Decomposition and Storage of Matrices

Student:

Carlos Henrique Chama Puga - 195416

Advisors:

Porf. Dr. Philippe Devloo

Dr. Giovane Avancini

Campinas

2024

Contents

1 Introduction **3**

2 The Full Matrix Class **6**

2.1 Class Implementation 6

2.2 Matrix LU Decomposition 6

2.3 Matrix LDLt Decomposition 6

3 The Sparse Matrix Class **6**

3.1 Class Implementation 6

3.2 Storage Structure and Memory Usage 6

3.3 Product Matrix Vector 6

4 Conclusions **6**

References **6**

A GitHub Repository **7**

1 Introduction

In mathematics, physics, and many engineering fields, the need to find the solution of a linear system of equations appears. Given a $N \times N$ linear system of the form of Eq. (1.1)

$$Ax = b, \tag{1.1}$$

in which the matrix A is the coefficient matrix, and the vector b is the equation's right-hand side, the solution vector x can be easily found by using the inverse of the matrix A as shown in Eq. (1.2)

$$x = A^{-1}b. \tag{1.2}$$

The task of inverting a matrix, however, is computationally expensive, and as the problem grows in size, the computational cost increases exponentially. In this context, the decomposition method arises to ease the computational burden of finding the inverse of a matrix.

This is justified by the fact that inverting the decomposed form $A = LU$ is computationally cheaper than inverting the original matrix A. The final solution is then obtained by solving two cheaper systems as shown in Eq. (1.3)

$$\begin{cases} Ax = LUx = b, \\ Ly = b, \\ Ux = y \end{cases} . \tag{1.3}$$

LU, LDU, LLt (or Cholesky), and LDLt decompositions are examples of this methodology, to cite a few.

This work will focus on the LU and LDLt methods, but the main idea behind

all of them is to decompose the original matrix A into two triangular matrices, L (lower triangular), and U (upper triangular). The D stands for a diagonal matrix and might appear in methods such as LDU and LDLt. Finally, for the Cholesky and LDLt decompositions, the upper matrix is the transpose of the lower matrix, $U = L^t$, which is achieved due to the symmetry of the problem.

To decompose a matrix, in the following methodology, the rank one update is employed. The rank one update is a method to update a matrix by adding the outer product between the vectors of the line and column of a given equation to the original submatrix formed by all lines and columns below the assessed equation. A simple example follows

$$A_{step1} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

The first equation is $1x_1 + 2x_2 + 3x_3 = b_1$. To update the matrix, the line and column are divided by the pivot (the term in the diagonal in this case 1), and the outer product between the vectors is calculated

$$cols \otimes rows = \begin{bmatrix} 8 & 12 \\ 14 & 21 \end{bmatrix},$$

and the result is subtracted from the original matrix

$$A_{step2} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -3 & -6 \\ 7 & -6 & -12 \end{bmatrix}.$$

This procedure is repeated until the last equation. Once it is over, the re-

sulting matrix is the decomposed form of the original one. The matrix L is the lower triangle of the decomposed matrix (not considering the diagonal) and the matrix U is the upper triangle of the decomposed matrix (including the diagonal). For the cases in which the diagonal matrix is required, the diagonal of the decomposed matrix is the matrix D and U does not include the diagonal.

One problem that might happen is when the pivot is zero. In these cases, the rank one update can not be executed since the division by zero is undetermined. To overcome this issue, the pivoting procedure is employed. It consists of swapping lines and/or columns to ensure that the pivot is not zero. The pivoting procedure is essential to guarantee the convergence of the decomposition method.

Pivoting can be done in two ways: partial pivoting, in which only lines or columns are swapped and full pivoting, in which both lines and columns are swapped. This work focuses on implementing the full pivoting procedure for both LU and LDLt decompositions.

This work also covers structures to storage matrices. Some forms of computational storage matrices include the full matrix, in which all elements are stored, a symmetric matrix, where only the upper or lower triangle is stored, band matrix, where only the diagonal and few elements close to it are stored (the band is more properly the matrix type rather than the storage method itself), and the sparse matrix, in which only the non-zero elements are stored.

Elements in a sparse matrix are stored in three vectors, usually called cols, values, and ptr. The cols vector stores which column each element is, the values vector stores the value of each element, and the ptr vector stores the cumulative number of elements per line in the matrix.

Sparse matrices are, allegedly, the most efficient way to store matrices, espe-

cially when the system is large and contains relatively low non-zero elements. For this reason, the sparse matrix is used in this work and compared to the full matrix storage method.

In this list, LU and LDLt decompositions are implemented and a sparse matrix class is developed to compare its storage performance. Finally, a method to multiply a matrix by a vector is implemented to test the sparse matrix class and results are compared. The literature used herein can be found in (1, 2, 3, 4)

2 The Full Matrix Class

2.1 Class Implementation

2.2 Matrix LU Decomposition

2.3 Matrix LDLt Decomposition

3 The Sparse Matrix Class

3.1 Class Implementation

3.2 Storage Structure and Memory Usage

3.3 Product Matrix Vector

4 Conclusions

References

1 CUNHA, M. C. de C. *Métodos numéricos*. [S.l.]: Editora da UNICAMP, 2000.

2 GOLUB, G. H.; LOAN, C. F. V. *Matrix computations*. [S.l.]: JHU press, 2013.

3 STRANG, G. *Introduction to linear algebra*. [S.l.]: SIAM, 2022.

4 JENNINGS, A. Matrix computation for engineers and scientists. *(No Title)*, 1977.

A GitHub Repository

The source code for this report and every code inhere mentioned can be found in the following GitHub repository: CarlosPuga14/MetodosNumericos_-2024S1.