

Assignment #3, Module: MA5633

Gustavo Ramirez

January 18, 2017

In this document are described the algorithms employed for solving the different tasks of this assignment.

1 LU FACTORISATION WITH PARTIAL PIVOTING

When preparing a matrix A to solve a system of equations of the form $A\vec{x} = \vec{b}$, some entries in the matrix can have a much different scale than the rest of the values in that matrix, which may lead to "divisions by zero" (due to the fact that the value is indeed zero, or due to the resolution of values stored in the computer, which makes it zero). The more prone to happen this in an algorithm, the more *unstable* is that algorithm.

A way to make certain algorithm more stable, is adding to it an extra step: *pivoting*. There are three types of pivoting: partial, complete and scaling. We will focus here on partial pivoting.

The algorithm to be used here is *LU decomposition*:

if a system of equations of the form $A\vec{x} = \vec{b}$ is to be solved, then the matrix A can be decomposed in the following way: $A = LU$, such that: $A\vec{x} = (LU)\vec{x} = \vec{b}$, from which we can find \vec{x} by first solving $L\vec{y} = \vec{b}$ for \vec{y} , and then solving $U\vec{x} = \vec{y}$ for \vec{x}

When applying that LU decomposition algorithm naively, i.e. by using the following equations:

$$u_{11} = a_{11}, u_{12} = a_{12}, \dots, u_{1n} = a_{1n}, l_{ii} = 1 \quad (1.1)$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{kj} l_{ik}, i > 1 \quad (1.2)$$

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} u_{kj} l_{ik} \right) \quad (1.3)$$

If any of those u_{jj} is zero, the system becomes unstable; the solution to this is *partial pivoting*: rearrange the rows of A in a way that, in each column, the largest element is at the diagonal.

When pivoting, each time a permutation of rows is performed, this represents a permutation in the vector \vec{b} . Permutations can be written as a matrix acting over A : $A_p = PA$, where A_p is a matrix where the diagonal entries are the largest on their corresponding columns; this implies:

$$A_p = L(U\vec{x}) = (PA)\vec{x} = P(A\vec{x}) = P(\vec{b}) \quad (1.4)$$

i.e. P is obtained previous to the LU decomposition, the decomposition is applied to A_p , and then the solution \vec{x} is obtained in two steps:

$$L\vec{y} = P\vec{b} \Rightarrow U\vec{x} = \vec{y} \quad (1.5)$$

Once the LU decomposition has taken place, the solutions are obtained as:

$$y_i = \left((P\vec{b})_i - \sum_{j=0}^{i-1} l_{ij}y_j \right) \frac{1}{l_{ii}} \quad (1.6)$$

for i going from $i = 0$ to $i = n - 1$, and finally:

$$x_i = \left(y_i - \sum_{j=i+1}^{n-1} u_{ij}x_j \right) \frac{1}{u_{ii}} \quad (1.7)$$

for i going from $i = n - 1$ to $i = 0$.

2 CONDITION NUMBER ESTIMATOR

To study the condition number estimator, it's necessary to introduce the infinity norm of a matrix:

$$\|A\|_{\infty} = \max_{1 \leq i \leq n} \left(\sum_{j=1}^n |a_{ij}| \right) \quad (2.1)$$

and in the case of vectors:

$$\|x\|_{\infty} = \max_{1 \leq i \leq n} |x_i| \quad (2.2)$$

The condition number gives some limits for the variations of the solutions of the linear system of equations, in terms of variations on their data. The bigger the condition number, the more prone to fluctuate is the solution when small changes on the data of the system are made.

3 JACOBI AND GAUSS-SEIDEL ITERATIVE SOLVERS

3.1 JACOBI

The algorithm is very simple: from the system of equation of the form $A\vec{x} = \vec{b}$, from the matrix A is extracted only the diagonal: $D_{ii} = A_{ii}$, $D_{ij} = 0$, $i \neq j$, and then another matrix is created: $R = A - D$.

Then, the solution of the system is obtained through the use of the iterative formula:

$$x^{k+1} = D^{-1}(n - Rx^k) \quad (3.1)$$

starting with an initial guess x^0 , and then iterating.

3.2 GAUSS-SEIDEL

Gauss-Seidel method is similar to Jacobi's Method, both being iterative methods for solving systems of linear equations, but Gauss-Seidel converges somewhat quicker in serial applications.

The method consists in the following: if the matrix A is decomposed into L (this lower matrix containing the diagonal as well) and U matrices, and after that split, the idea is to apply the following iterative formula:

$$x^{k+1} = L^{-1}(b - Ux^k) \quad (3.2)$$

4 FORWARD EULER'S METHOD

5 TRAPEZOID RULE PREDICTOR-CORRECTOR METHOD

6 SECOND ORDER RUNGE-KUTTA

7 FOURTH ORDER RUNGE-KUTTA