Systems of Linear Equations

Exercise 7

Systems of Linear Equations Report

Author
Cesare De Cal

Professor
Annie Cuyt

Assistant Professor
Ferre Knaepkens

October 28, 2019

1 Introduction

In this exercise I am going to use solve a linear system $Ax = y$ using LU decomposition. The first element of the unknowns vector x will contain an approximation of $e - 2$.

As we've seen in class, there are multiple ways of solving a linear system $Ax = y$. Assume A is a square matrix of order n, y is a column vector with n components, and x is the vector of the unknowns. Calculating the inverse A^{-1} to solve the system requires more computations than necessary, and returns a less accurate result. On the other hand, Gaussian Elimination is an algorithm that can be used to represent the coefficients matrix A in terms of simpler matrices:

$$LU = PA$$

I'll be solving the system by using the latter mentioned algorithm, and then provide observations.

2 Tools

The following programming language and libraries have been used in this exercise:

- C
- GSL (GNU Scientific Library)

The following GSL data types have been used in the exercise:

- `gsl_vector`
- `gsl_matrix`
- `gsl_permutation`

The following GSL methods have been used in the exercise:

- `gsl_matrix_alloc(size1, size2)`
- `gsl_matrix_set_zero(matrix)`
- `gsl_matrix_set(matrix, row, column, value)`
- `gsl_matrix_get(matrix, row, column)`
- `gsl_vector_alloc(size)`
- `gsl_vector_set_zero(vector)`
- `gsl_vector_set(vector, index, value)`
- `gsl_vector_get(vector, index)`
- `gsl_permutation_alloc(size)`
- `gsl_matrix_memcpy(matrixToCopyFrom, matrix)`

In order to factorize a matrix into the LU decomposition, and then solve the square system $Ax = y$ using the decomposition of A, I've used the following methods:

- `gsl_linalg_LU_decomp(A, permutation, signum)`
- `gsl_linalg_LU_solve(LU, permutation, b, x)`

3 Solving the linear system

In order to solve the system $Ax = y$, I first need to build the matrix A by understanding how it's build. The requirements are to build a tridiagonal matrix with the values -1 on the adjacent upper diagonal, the entries $+1$ on the adjacent lower diagonal, and on the main diagonal the values b_i, with $i = 1, \ldots, n$ given by

$$b_i = \frac{2(i+1)}{3}, \quad i+1 = 3, 6, 9, \ldots$$
$$b_i = 1, \quad i+1 = 2, 4, 5, 7, 8, \ldots$$

By looking closely at the first rule, we see that the $i+1$ are all multiples of 3 ($i+1 = 3*k$, for some k). Hence the i are of the form $i = 3*k-1$, for some k. For $n = 5$, for example, this is what the matrix looks like:

$$\begin{bmatrix} 1.0000000000 & -1.0000000000 & 0.0000000000 & 0.0000000000 & 0.0000000000 \\ 1.0000000000 & 2.0000000000 & -1.0000000000 & 0.0000000000 & 0.0000000000 \\ 0.0000000000 & 1.0000000000 & 1.0000000000 & -1.0000000000 & 0.0000000000 \\ 0.0000000000 & 0.0000000000 & 1.0000000000 & 1.0000000000 & -1.0000000000 \\ 0.0000000000 & 0.0000000000 & 0.0000000000 & 1.0000000000 & 4.0000000000 \end{bmatrix}$$

The coefficients matrix A is first allocated by using the `gsl_matrix_alloc` method, then I set all the elements to zero with `gsl_matrix_set_zero` and finally nested `for` loops fill the diagonal values by checking the indexes. The coefficients reported above on the diagonal have 5 significant digits for improve the readability of this report.

I used the `gsl_vector_alloc` method to create an instance of the vector. All of its elements were set to zero by using `gsl_vector_set_zero(vector)`. The exercise asks us to set the first element of the y vector to one, so I used `gsl_vector_set(vector, 0, 1)` to assign the value 1 to index 0. For $n = 5$, we have:

$$\vec{y} = \begin{bmatrix} 1.0000000000 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \end{bmatrix}$$

Given the $Ax = y$ system, my goal is now to find the vector of the unknowns x. To do so, I first factorize A into its LU decomposition by allocating a new matrix (so that the matrix which represents A doesn't get overridden) using `gsl_matrix_memcpy` and then by calling `gsl_linalg_LU_decomp`. This method utilizes Gaussian Elimination with partial pivoting to compute the decomposition. The following is the LU matrix for $n = 5$:

$$LU = \begin{bmatrix} 1.0000000000 & -1.0000000000 & 0.0000000000 & 0.0000000000 & 0.0000000000 \\ 1.0000000000 & 3.0000000000 & -1.0000000000 & 0.0000000000 & 0.0000000000 \\ 0.0000000000 & 0.3333333333 & 1.3333333333 & -1.0000000000 & 0.0000000000 \\ 0.0000000000 & 0.0000000000 & 0.7500000000 & 1.7500000000 & -1.0000000000 \\ 0.0000000000 & 0.0000000000 & 0.0000000000 & 0.5714285714 & 4.5714285714 \end{bmatrix}$$

I can now use the LU matrix to solve the system by passing LU, x, a permutation structure `gsl_permutation` and y to `gsl_linalg_LU_solve`. This method modifies the contents of the x vector given in input, which now looks like this (for $n = 5$):

$$\vec{x} = \begin{bmatrix} 0.7187500000 \\ -0.2812500000 \\ 0.1562500000 \\ -0.1250000000 \\ 0.0312500000 \end{bmatrix}$$

The first element looks contains an approximation of $e - 2$. By increasing the size of the matrix n, x_i becomes increasingly more precise. For $n = 10$, for example:

$$\vec{x} = \begin{bmatrix} 0.7182817183 \\ -0.2817182817 \\ 0.1548451548 \\ -0.1268731269 \\ 0.0279720280 \\ -0.0149850150 \\ 0.0129870130 \\ -0.0019980020 \\ 0.0009990010 \\ -0.0009990010 \end{bmatrix}$$

For $n = 20$:

$$\vec{y} = \begin{bmatrix} 0.7182818285 \\ -0.2817181715 \\ 0.1548454854 \\ -0.1268726862 \\ 0.0279727992 \\ -0.0149814893 \\ 0.0129913099 \\ -0.0019901794 \\ 0.0010502335 \\ -0.0009399460 \\ 0.0001102875 \\ -0.0000576459 \\ 0.0000526416 \\ -0.0000050043 \\ 0.0000025983 \\ -0.0000024061 \\ 0.0000001922 \\ -0.0000000994 \\ 0.0000000928 \\ -0.0000000066 \end{bmatrix}$$

4 Observations