Systems of Linear Equations

Exercise 7

# Solving a Linear System with LU Decomposition

*Author*
Cesare De Cal

*Professor*
Annie Cuyt

*Assistant Professor*
Ferre Knaepkens

October 31, 2019

# 1 Introduction

In this exercise I am going to use solve a linear system $A\vec{x} = \vec{y}$ using LU decomposition. The goal is to verify that first element of the unknowns vector, $x_1$, will contain an approximation of $e - 2$.

As we've seen in class, there are multiple ways of solving a linear system $AX = B$. Assume $A$ is a $n \times n$ square matrix, $B$ is a "constant" term matrix $n \times h$, and $X$ is a $n \times h$ unknown matrix. To solve for $X$, we could compute the inverse of $A$ and find $x = A^{-1}y$. We've seen that this approach, however, requires more computations than necessary and returns a less accurate result.

On the other hand, the LU decomposition technique is a way to represent the matrix A in the form of simpler matrices, $L$ and $U$ (lower triangular and upper triangular matrices, respectively):

$$PA = LU$$

This method uses forward substitution (solving for $Y$ from $LY = B$) and backward substitution (solving for $X$ from $UX = Y$). I'll be specifically solving the system by using Gaussian Elimination with partial pivoting, which reduces round-offs errors compared to its naive implementation. I'll also be calculating the error and the condition number as the variable $n$ increases, and plot the results.

## 2 Tools

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

- C
- GSL (GNU Scientific Library)

The following double-precision 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_matrix_memcpy(matrixToCopyFrom, matrix)`
- `gsl_linalg_SV_decomp(A, V, S, workspaceVector)`
- `gsl_vector_minmax(vector, minInVector, maxInVector)`

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)`
- `gsl_permutation_alloc(size)`

# 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 uses forward and back-substitution to modify 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:

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

Then, I calculate the condition number of the matrix $A$ of order $n$ which will give me a better idea if this is a well-conditioned or an ill-conditioned linear system. In GSL there is no direct function that calculates the condition number, but it's possible to use the ratio of the largest singular value of matrix A, $\sigma_n(A)$, to the smallest $\sigma_1(A)$:

$$\kappa(A) := \frac{\sigma_n(A)}{\sigma_1(A)} = \frac{\|A\|}{\|A^{-1}\|^{-1}}$$

I proceed to factorize $A$ into its singular value decomposition $SVD$ using the `gsl_linalg_SV_decomp` method, and then use `gsl_vector_minmax` to extract the minimum and maximum singular values out of the vector $S$ that contains the diagonal elements of the singular value matrix.

For $n = 5$, the condition number is

$$\kappa(A) = \frac{\sigma_n(A)}{\sigma_1(A)} = \frac{4.2051006107}{1.1426432872} = 3.6801516779$$

For $n = 10$, the condition number is

$$\kappa(A) = \frac{\sigma_n(A)}{\sigma_1(A)} = \frac{6.2820508697}{1.1424251953} = 5.4988728328$$

I calculate the error by subtracting the computed solution $x_1^*$ from the exact mathematical solution $\widetilde{x}$ (which can be obtained by using the `M_E` GSL constant minus 2).

Finally, I am now going to insert the $x_1$ component, the error, and the condition number for $n$ from 1 to 50 in the next page. I chose $n = 50$ as the upper limit because it looks like the error doesn't get better after $n = 20$.

| $n$ | $\widetilde{x}_1$ | $x_1^* - \widetilde{x}_1$ | $\kappa(A_n)$ |
|---|---|---|---|
| 1 | 1.0000000000000000 | -0.2817181715409549 | 1.0000000000000000 |
| 2 | 0.6666666666666667 | 0.0516151617923784 | 1.7675918792439982 |
| 3 | 0.7500000000000000 | -0.0317181715409549 | 2.5615528128088298 |
| 4 | 0.7142857142857142 | 0.0039961141733309 | 2.2586960380558874 |
| 5 | 0.7187500000000000 | -0.0004681715409549 | 3.6801516778795333 |
| 6 | 0.7179487179487180 | 0.0003331105103271 | 3.9538640020225495 |
| 7 | 0.7183098591549295 | -0.0000280306958844 | 3.8476746091189153 |
| 8 | 0.7182795698924731 | 0.0000022585665720 | 5.3770375880477213 |
| 9 | 0.7182835820895522 | -0.0000017536305071 | 5.7275818392893347 |
| 10 | 0.7182817182817183 | 0.0000001101773268 | 5.4988728328025962 |
| 11 | 0.7182818352059925 | -0.0000000067469474 | 7.1003357703679963 |
| 12 | 0.7182818229439497 | 0.0000000055150954 | 7.5821646375997114 |
| 13 | 0.7182818287356958 | -0.0000000002766507 | 7.1955317021216594 |
| 14 | 0.7182818284454013 | 0.0000000000136438 | 8.8331498923754399 |
| 15 | 0.7182818284705836 | -0.0000000000115385 | 9.4880747300490409 |
| 16 | 0.7182818284585635 | 0.0000000000004816 | 8.9115586964085285 |
| 17 | 0.7182818284590651 | -0.0000000000000200 | 10.5715228483523767 |
| 18 | 0.7182818284590280 | 0.0000000000000171 | 11.4201824603811897 |
| 19 | 0.7182818284590458 | -0.0000000000000007 | 10.6381340746273825 |
| 20 | 0.7182818284590452 | -0.0000000000000001 | 12.3131996586543000 |
| 21 | 0.7182818284590453 | -0.0000000000000002 | 13.3688310431707524 |
| 22 | 0.7182818284590453 | -0.0000000000000002 | 12.3710782126140035 |
| 23 | 0.7182818284590453 | -0.0000000000000002 | 14.0570047940165566 |
| 24 | 0.7182818284590453 | -0.0000000000000002 | 15.3286298262578065 |
| 25 | 0.7182818284590453 | -0.0000000000000002 | 14.1081637697318296 |
| 26 | 0.7182818284590453 | -0.0000000000000002 | 15.8022624874376820 |
| 27 | 0.7182818284590453 | -0.0000000000000002 | 17.2963070591942554 |
| 28 | 0.7182818284590453 | -0.0000000000000002 | 15.8480934751923019 |
| 29 | 0.7182818284590453 | -0.0000000000000002 | 17.5485561661636069 |
| 30 | 0.7182818284590453 | -0.0000000000000002 | 19.2697572434997504 |
| 31 | 0.7182818284590453 | -0.0000000000000002 | 17.5900604343512654 |
| 32 | 0.7182818284590453 | -0.0000000000000002 | 19.2956148454130130 |
| 33 | 0.7182818284590453 | -0.0000000000000002 | 21.2475632526110978 |
| 34 | 0.7182818284590453 | -0.0000000000000002 | 19.3335364470429454 |
| 35 | 0.7182818284590453 | -0.0000000000000002 | 21.0432545577449801 |
| 36 | 0.7182818284590453 | -0.0000000000000002 | 23.2287362210777601 |
| 37 | 0.7182818284590453 | -0.0000000000000002 | 21.0781612793335533 |
| 38 | 0.7182818284590453 | -0.0000000000000002 | 22.7913459874268973 |
| 39 | 0.7182818284590453 | -0.0000000000000002 | 25.2125651999996911 |
| 40 | 0.7182818284590453 | -0.0000000000000002 | 22.8236808378170366 |
| 41 | 0.7182818284590453 | -0.0000000000000002 | 24.5397955641461145 |
| 42 | 0.7182818284590453 | -0.0000000000000002 | 27.1985260013623389 |
| 43 | 0.7182818284590453 | -0.0000000000000002 | 24.5699107748932342 |
| 44 | 0.7182818284590453 | -0.0000000000000002 | 26.2885339032351872 |
| 45 | 0.7182818284590453 | -0.0000000000000002 | 29.1862237020870907 |
| 46 | 0.7182818284590453 | -0.0000000000000002 | 26.3167141046176667 |
| 47 | 0.7182818284590453 | -0.0000000000000002 | 28.0375084620521520 |
| 48 | 0.7182818284590453 | -0.0000000000000002 | 31.1753551463410439 |
| 49 | 0.7182818284590453 | -0.0000000000000002 | 28.0639869168957361 |
| 50 | 0.7182818284590453 | -0.0000000000000002 | 29.7866787202101015 |

# 4 Observations

It can be observed that as $n$ increases, the $\widetilde{x}_1$ component gets incrementally closer to the actual $e-2$ value. The condition number also gets incrementally bigger. It can be noticed, however, that a large condition number doesn't necessarily mean that the error will be large in all cases, just that it is possible to have a large error.

The linear system appears to be ill-conditioned for most $n$.