Computer Practise 6

Computer Practise with MATLAB: Systems of lineal equations

6.1 Exercises.

1. The exercises can be done individually or in groups of at most two people. Exercises ''shared'' or copied from your companions or form previous years will be graded with 0 points.

2. Each group has to email to **virginia.muto@ehu.eus** a zip file with the name *group.zip* that has to contain the script files that solve the questions stated at the end of the page (a), (b) and (c). The files have to contain the explication of the used commands. Each group has one week to solve the exercises and late deliver will be penalized with half the mark.

3. The name(s) and surname(s) of the author(s) have to appear also in the first line of every Matlab file. Unnamed files will not be graded.

4. Run always the codes before submitting them.

5. Besides the functions used in the previous classes, there will be new MATLAB functions that we describe in the following.

6. In the first part of this practise, we explore the basic MATLAB tools for solving systems of linear equations.

   (a) MATLAB has the operator ''lu'' which allows to compute the LU factorization:

   - If we use it as [L,U,P]=lu(A), we will get the LU factorization of A with pivoting: P is the permutation matrix, L is the lower triangular one (with zeros above the main diagonal) and U is the upper triangular one (with zeros below the main diagonal): $PA = LU$.
   - If we use the operator ''lu'' as [L,U]=lu(A), we will get the LU factorization of A, where L appears with permuted rows, $L = P^T L : A = (P^T L)U$.

The instruction $x = A\backslash b$ is equivalent to

```
[L,U]=lu(A),    x=U\(L\b)
```

or also to

```
[L,U,P]=lu(A),    x=U\(L\(P*b)).
```

These last forms may be useful if one wants to solve more than one system with the same A matrix.

- In order to construct a $n \times n$ symmetric Toeplitz matrix in MATLAB we can use the instruction
  `toeplitz(1:n)`
- Build a 5×5 symmetric Toeplitz matrix A and compute its LU factorization with pivoting, writing the L, U and P factors.
- Using the previously obtained result, solve the system $Ax = b$ for b given by each of the following vectors:

$$\begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} 9 \\ 4 \\ 2 \\ 0 \\ -1 \end{pmatrix}, \qquad \begin{pmatrix} 10^{-3} \\ -0.33 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

  Check the results of each case.

(b) Also, MATLAB has the operator ‘‘chol’’ which allows to compute the Cholesky factorization $C^T C$:

- It is used as C=chol(A), therefore we will obtain the C matrix, with factorization $C^t C$ for A: C is the lower triangular matrix and C^T is the upper triangular one: $A = C^t C$. It is equivalent to C = chol(A,’upper’).
- If we type L = chol(A,’lower’), we will get the factorization LL^T of A, where L is the lower triangular matrix.
- In order to build $n \times n$ matrices to use for testing in MATLAB we can use the command *gallery* whose syntax is
  `[A,B,C,...] = gallery('matname',P1,P2,...)`
  ‘‘matname’’ is the name of a family of matrices and the numbers $P1, P2, \ldots.$ mean different things depending on the family. One has to look up for each case.
- Build a 5×5 symmetric and positive definite A, using the command ‘‘gallery’’ and the matrices’ family ‘‘moler’’. Here the meaning of $P1$ is the matrix dimension. Compute the factorization $C^t C$ of A.
- Build a 5×5 symmetric and positive semi definite A, using the command ‘‘gallery’’ and the matrices’ family ‘‘toeppd’’. Here the meaning of $P1$ is the matrix dimension. Compute the factorization $C^t C$ of A.

- Using the previously obtained results, solve the system $Ax = b$ for A, b indicated as follow:

  `A = [[6,16,21,33]; [2,20,18,16]; [3,18,19,21]; [1,2,3,4]],`

  with the vectors b given above in 6 (a) and eliminating the last row;

  `A = magic(5),`
  `A = hilb(5),`
  `A = pascal(5),`

  with the vectors b given above 6 (a); and finally with

  `A = rand(15,15),`
  `b = rand(15,1).`

7. In the second part of this computer practice, we want to implement the Doolittle factorization algorithm. You can find it in the book "Numerical Analysis" written by R.L. Burden and J. D. Faires (Brooks/Cole 2011).

   or following the algorithm presented on pages 144-145 of my notes

   http://www.ehu.eus/~mepmufov/html/Parte3.pdf

   and the suggestions about MATLAB that you can find in the Web of Prof. Palomares of University of Granada

   https://www.ugr.es/~anpalom/practica6.html.

   Crout method is a slight modification of Doolittle one.

   In this factorization method, given A matrix we want to build two matrices L and U such that $L * U = A$. L matrix represents a lower triangulaw matrix and U represents the upper triangular one.

   - First of all, let's see how we can implement the sums that appear in the Doolittle method as the scalar product of two vectors.

     In Doolittle method, in order to get $U(k, j)$ we have to add up the sum $L(k, r) * U(r, j)$ with $r = 1, 2, ..., k - 1$. Namely, we have to compute

     $$L(k, 1) * U(1, j) + L(k, 2) * U(2, j) + ... + L(k, k - 1) * U(k - 1, j).$$

     This sum can be written as the scalar product of the row vector $L(k, 1 : k - 1)$ times the column vector $U(1 : k - 1, j)$.

     In a similar manner, in order to obtain $L(i, k)$ we need the sum $L(i, r) * U(r, k)$ with $r = 1, 2, ..., k - 1$.

     This sum can be written as the scalar product $L(i, 1 : k - 1) * U(1 : k - 1, k)$.

   Let's give the details of the different steps of the algorithm that you have to write in MATLAB.

- In Doolittle method the L matrix contains ones in the main diagonal.

  We will start defining the A matrix and initializing the variables L and U as matrices $zero(n)$, with $n = length(A)$.
- Then one implements the case for $k = 1$, since in the algorithm appear sums for the index r from 1 to $k - 1$. When $k = 1$, these sum has not to be computed and therefore in Matlab this case ($k = 1$) has to be done separately.
- We compute the rest of elements of L and U following the order:

```
For 2<=k<=n
For k<=j<=n
U(k,j)
Next j
L(k,k)=1
For k+1<=i<=n
L(i,k)
Next i
Next k
```

- We will show the result and check the factorization $A = L * U$.

Finally, if $A = L * U$, then system $A * x = b$ can be written as $L * U * x = b$.

In order to obtain x we can introduce an auxiliary vector $z = U * x$, such that the system can be thought as $L * z = b$. Now in order to obtain z, we have only to solve a triangular system. Once we know z, in order to compute x is enough to solve $U * x = z$, that is again a triangular system.

Therefore, in order to solve a system of linear equations is reduced to find a factorization $L * U$ and then to solve two triangular systems. Both steps of factorization and solution of the triangular systems are done with scripts as the ones described above.

Shortly, the steps needed to solve a system using the $L * U$ factorization:

- Factorize A = L*U.
- Solve $Ly = b$, so we get y.
- Solce $Ux = y$, so we get x.

More details on page 147 of http://www.ehu.eus/˜mepmufov/html/Parte3.pdf.

- Using the results obtained in the previous points, solve systems $Ax = b$ for A and b given in the previous exercises 6 (a) and 6 (b).